MATHEMATICAL OPTIMIZATION METHODS FOR THE REMEDIATION OF GROUND WATER CONTAMINATIONS

Dissertation

zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften

Dem Fachbereich IV der Universität Trier
vorgelegt von

ASTRID BATTERMANN

Trier, im Dezember 2000

1. Gutachter: Prof. Dr. E. W. Sachs
2. Gutachter: Prof. Dr. C. T. Kelley

Tag der mündlichen Prüfung: 9. Februar 2001

TO H. AND G. BATTERMANN

Acknowledgments

I am indebted to many people for their help, guidance and generous support. In particular, I would like to thank my advisor Prof. Sachs for his numerous, direct and indirect, contributions to my mathematical education, and, specifically, to the making of this work. I am especially grateful also for the assistance and suggestions from the engineers of TGU GmbH while working out the specifics of their applications over the past three years.

Thanks are extended to the two students A. Bringer and A. Krahnke who have been working with me on the project. Financial support is acknowledged from the foundation Stiftung Rheinland–Pfalz für Innovation that made the work of the last three years possible.

Many others have contributed to make my mathematical life over the last years a rewarding experience. Among those, I would like to express my gratitude to my coreader Prof. Kelley from North Carolina State University for his interest in my work and for valuable comments.

Contents

Chapter 1

Introduction

This work is concerned with the numerical solution of optimization problems that arise in the context of ground water modeling. Ground water hydraulic and quality management is a field of wide interest that has not yet received much attention from the optimization community. The optimization problems considered in this work are discretized optimal control problems. As such, they exhibit a particular structure which is due to the partitioning of variables and to the infinite dimensional nature of the original problems. In these problems there are two sets of variables, state and control variables. This partitioning introduces structure that can be efficiently used in optimization algorithms. Originally, the variables are functions, and the constraint functions involve partial differential equations. For a numerical solution, these have to be discretized which is another source of structure. Our goal is to make efficient use of this structure.

But mathematical theory cannot always be exploited to full extent. Practicality may for instance enforce the usage of external, off–the–shelf software. This is the situation we encounter for one of our applications. In this case study, usage of external software excludes some problem formulations and, consequently, certain mathematical approaches to the problem. Moreover, structure is lost in the reduced, black–box, approach that we use. Additionally, errors are introduced into the objective function through the numerical treatment in the codes. One is well advised to refrain from taking higher derivatives, and special care has to be taken in using first–order information obtained through finite–differencing. Also, rather large error margins must be accounted for which are induced through simplifications in the modeling and through the data situation. Methods for noisy functions are appropriate. The treatment of our first practical application is laid out in Chapters 2, 3, and 4.

Two main examples are our guideline through this work. We deal with another application in ground water modeling in the subsequent chapters, Chapters 5, 6, and 7. It is approached from a different angle. A self–coded discretization allows to perform

all computations that are necessary for the so called full–system approach. A common method for nonlinearly constrained nonlinear optimization problems is sequential quadratic programming (SQP). In SQP methods a sequence of linearly constrained quadratic subproblems is solved. The solution of a subproblem can be done by assembling all relevant equations in one large system and solving them simultaneously. In this full–system approach, the solution of a subproblem requires the solution of a linear system. In order to build a practical algorithm, it is crucial to be able to solve the linear systems efficiently. We therefore concern ourselves with preconditioners to accelerate the iterative solution of the linear systems. These preconditioners are block preconditioners. Their block structure is similar to that of the original system in order to allow its exploitation.

Both applications were brought to our attention by the industrial partner in this project, Technologieberatung Grundwasser und Umwelt (TGU). TGU GmbH is a company of consulting engineers for ground water and water resources on whose experience we could draw in this project. The project with the working title “Mathematical Optimization Methods for the Remediation of Ground Water Contaminations” was financed by the foundation Stiftung Rheinland–Pfalz für Innovation.

1.1 Scope

This work is concerned with the numerical solution of optimization problems from the ground water management context. The guideline to this work is provided by two examples. Both examples are taken from the ground water modeling context, and both are problems of optimal control that are governed by partial differential equations. Despite these common grounds, the example problems require different solution techniques. One reason is their different formulation. The choice of problem formulation is for both problems motivated in the following. Also, the optimization approaches and main findings are delineated.

Problem formulations An optimization problem can usually be posed in various, mathematically equivalent, ways. These mathematical formulations differ in the type of constraints and optimization variables. Different formulations are obtained by using parts of the constraints to eliminate variables, or by exposing variables to the optimization and introducing coupling constraints. Such problem formulations can have a great impact on how the optimization problem can be solved efficiently. The coupling of variables and the structure corresponding to a given problem formulation can make some formulations more suitable for certain optimization methods. The formulations

together with the choice of particular solution algorithms also determine to which extent, and of which content, problem information has to be provided to the optimization algorithm. Problem formulation issues are addressed in various places for particular problems and in the general optimization literature. See e.g. [36, Ch. 7]. The two problem formulations that figure in this work are the all–at–once and black–box formulation, respectively. We frame our first application as a black–box problem, while the second is solved in its all–at–once formulation.

We present the formulations for the real, possibly nonlinear, equality–constrained optimization problem

$$\min_{(\phi,u)} \; f(\phi,u) \quad \text{s.t.} \quad c(\phi,u)=0. \tag{1.1.1}$$

The objective function of problem (1.1.1) is f, $f : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$. The constraints are c, $c : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}^m$ for appropriate integers m and k of which m is usually by far the larger in our context. In (1.1.1), $\phi \in \mathbb{R}^m$ represents the state variables, and $u \in \mathbb{R}^k$ represents the control variables for a problem of optimal control. The equation $c(\phi,u)=0$ stands for the state equation. The state equation is in our context typically a discretized partial differential equation whose solution requires considerable computational effort.

The formulation (1.1.1) is called the all–at–once formulation of the problem. Both states and controls are viewed as variables. Under appropriate differentiability assumptions on objective f and constraints c, the implicit function theorem can be used to obtain a reduced formulation. If there exist (ϕ,u) that satisfy the state equation, and if the partial derivative $c_\phi(\phi,u)$ of c with respect to ϕ is nonsingular, it is possible to eliminate the state equation from (1.1.1) by defining ϕ implicitly as a function $\phi(u)$ of u by the equation $c(\phi(u),u)=0$. Then one obtains the black–box formulation of the problem (1.1.1) as

$$\min_u \quad \widehat{f}(u) = f(\phi(u),u).$$

Obviously, the number and type of variables and constraints change with the problem. Measured in number of variables and constraints, the all–at–once formulation is typically by far the largest problem. The black–box formulation typically leads to the smallest problem. But the expense of a single function evaluation is typically significantly higher for the black–box formulation than for the all–at–once formulation. This is due to the fact that the objective function in the black–box case requires the evaluation of $\phi(u)$, the implicitly defined function. This evaluation often corresponds to the solution of nonlinear systems. The increase in cost of function evaluations when switching from the all–at–once to the black–box case also carries over to the derivatives. Availability and, in second line, ease of derivative computations can have considerable impact on the choice of problem formulation. So can the size of the quantities

involved. Sometimes the constraints $c(\phi, u) = 0$ are so large that they cannot be incorporated into a given implementation of an optimization algorithm. This is often the case when the constraints represent discretized partial differential equations, and software packages are brought into service to treat the constraints. It is also possible that, given software to treat the constraints, the software cannot solve the adjoint equation. However, as we see in the following paragraph, the adjoint solve is a prerequisite for the all–at–once approach. We are given external software for the equation solve in our first application. Its inability to perform the adjoint solve excludes the all–at–once approach. There is an additional point in favor of the black–box approach for applications with expensive constraints. Within an optimization algorithm based on the all–at–once formulation, progress towards optimality and progress towards feasibility with respect to the state equation have to be made simultaneously. Meanwhile, only feasible iterates with respect to the original constraints $c(\phi, u) = 0$ are considered in the course of optimization with the black–box approach. If the main computational expense is the equation solve, then, upon premature termination, one does not want to end up with an infeasible iterate.

General framework The two applications that we are concerned with in the present work are derived as infinite–dimensional problems. Numerically, we deal with their discretizations, of course. But since structure is inherited from the infinite–dimensional setting, we introduce some concepts in that setting. We refrain from rigorously formulating these concepts here. For an introduction to the infinite–dimensional problems and differentiability properties within a Hilbert space framework see e.g. [23], [46], [84].

Consider the general optimization problem

$$\min_{(\mathcal{X},\mathcal{Y})} \mathcal{F}(\mathcal{X}, \mathcal{Y}) \quad \text{s.t.} \quad \mathcal{C}(\mathcal{X}, \mathcal{Y}) = 0, \tag{1.1.2}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are functions in some Hilbert spaces X and Y, where $\mathcal{F}$ maps the functions to the real line, and where $\mathcal{C}$ stands for a differential equation with values in some Hilbert space D. We call $\mathcal{X}$ the state variable and $\mathcal{Y}$ the control variable. The state equation $\mathcal{C}(\mathcal{X}, \mathcal{Y}) = 0$ governs the optimization problem. Typically, the solution of the (discretized) state equation, the so called equation solve, dominates the cost of one step of an optimization method.

The linearization of the state equation is given by

$$\mathcal{C}_{\mathcal{X}}(\mathcal{X}, \mathcal{Y})(\delta\mathcal{X}) + \mathcal{C}_{\mathcal{Y}}(\mathcal{X}, \mathcal{Y})(\delta\mathcal{Y}) = 0,$$

$\delta\mathcal{X} \in X$, $\delta\mathcal{Y} \in Y$. Subscripts denote differentiation, in this instance in the Fréchet sense. The partial derivative $\mathcal{C}_{\mathcal{X}}(\mathcal{X}, \mathcal{Y})$, abbreviated $\mathcal{C}_{\mathcal{X}}$, is a linear function in the

increment $\delta\mathcal{X}$. We assume throughout this part that all differentiation that we perform is justified and that all quantities involved do exist.

Under appropriate differentiability assumptions on the state equation, a reduced formulation can be obtained for the optimization problem (1.1.2) via the implicit function theorem: If the partial Fréchet derivative of $\mathcal{C}$ with respect to $\mathcal{X}$ at a solution pair $(\mathcal{X}_0, \mathcal{Y}_0)$, $\mathcal{C}_\mathcal{X}(\mathcal{X}_0, \mathcal{Y}_0)(.)$, has a bounded inverse, then the state equation is locally uniquely solvable. A function $\mathcal{X}(\mathcal{Y})$ can locally be implicitly defined by this solution. The implicitly defined function $\mathcal{X}(\mathcal{Y})$ is continuously Fréchet differentiable and the derivative is

$$\mathcal{X}_\mathcal{Y}(\mathcal{Y}) = -\mathcal{C}_\mathcal{X}^{-1}(\mathcal{X}(\mathcal{Y}), \mathcal{Y})\, \mathcal{C}_\mathcal{Y}(\mathcal{X}(\mathcal{Y}), \mathcal{Y}). \tag{1.1.3}$$

This implicit definition allows the reduced formulation

$$\min_\mathcal{Y} \hat{\mathcal{F}}(\mathcal{Y}) = \mathcal{F}(\mathcal{X}(\mathcal{Y}), \mathcal{Y}) \tag{1.1.4}$$

which is equivalent to (1.1.2). Using the chain rule, the derivative of the reduced problem (1.1.4) is, using (1.1.3), computed as

$$\hat{\mathcal{F}}_\mathcal{Y}(\mathcal{Y}) = -\Big(\mathcal{C}_\mathcal{X}^{-1}(\mathcal{X}(\mathcal{Y}), \mathcal{Y})\, \mathcal{C}_\mathcal{Y}(\mathcal{X}(\mathcal{Y}), \mathcal{Y})\Big)^\times \mathcal{F}_\mathcal{X}(\mathcal{X}, \mathcal{Y}) + \mathcal{F}_\mathcal{Y}(\mathcal{X}, \mathcal{Y}). \tag{1.1.5}$$

Formula (1.1.5) is used in the sensitivity equation method to compute the gradient of the reduced problem (1.1.4). The superscript $^\times$ signifies the adjoint operator which is defined with respect to the considered inner product on the relevant Hilbert space. The null space of the linearized constraints has a representation via the linear operator $\mathcal{W}$,

$$\mathcal{W}(\mathcal{X}, \mathcal{Y}) = \begin{pmatrix} -\mathcal{C}_\mathcal{X}^{-1}(\mathcal{X}, \mathcal{Y})\, \mathcal{C}_\mathcal{Y}(\mathcal{X}, \mathcal{Y}) \\ \mathcal{I} \end{pmatrix}.$$

With this, the gradient $\hat{\mathcal{F}}_\mathcal{Y}(\mathcal{Y})$ of the reduced problem in (1.1.5), which is also called the reduced gradient of the original problem (1.1.2), can be seen to obey

$$\hat{\mathcal{F}}_\mathcal{Y}(\mathcal{Y}) = \mathcal{W}(\mathcal{X}, \mathcal{Y})^\times \begin{pmatrix} \mathcal{F}_\mathcal{X}(\mathcal{X}, \mathcal{Y}) \\ \mathcal{F}_\mathcal{Y}(\mathcal{X}, \mathcal{Y}) \end{pmatrix}.$$

In addition to the sensitivity equation method for the computation of the reduced gradient there is the adjoint equation method, usually derived by rewriting (1.1.5) as

$$\hat{\mathcal{F}}_\mathcal{Y}(\mathcal{Y}) = -\mathcal{C}_\mathcal{Y}(\mathcal{X}(\mathcal{Y}), \mathcal{Y}))^\times \Big((\mathcal{C}_\mathcal{X}(\mathcal{X}(\mathcal{Y}), \mathcal{Y})^\times\Big)^{-1} \mathcal{F}_\mathcal{X}(\mathcal{X}, \mathcal{Y}) + \mathcal{F}_\mathcal{Y}(\mathcal{X}, \mathcal{Y}).$$

The adjoint variables $\mathcal{P}(\mathcal{Y})$ are elements of $D^\times$, the dual of D, and defined as the solution of the adjoint equation

$$(\mathcal{C}_\mathcal{X}(\mathcal{X}(\mathcal{Y}), \mathcal{Y})^\times\, \mathcal{P}(\mathcal{Y}) = -\mathcal{F}_\mathcal{X}(\mathcal{X}, \mathcal{Y}). \tag{1.1.6}$$

In an all–at–once approach, all variables, state, control, and adjoint, are simultaneously solved for. Consequentially, the adjoint solve is a prerequisite for such an approach.

It follows directly from the representation (1.1.6) that the reduced gradient can be written as

$$\widehat{\mathcal{F}}_{\mathcal{Y}}(\mathcal{Y}) = \mathcal{C}_{\mathcal{Y}}(\mathcal{X}(\mathcal{Y}),\mathcal{Y})^{\times}\,\mathcal{P}(\mathcal{Y}) + \mathcal{F}_{\mathcal{Y}}(\mathcal{X},\mathcal{Y}).$$

The adjoint equation is linear in the adjoint variables $\mathcal{P}$ for given $\mathcal{Y}$. Despite the linearity, the adjoint equation method is not always the method of choice. The adjoint $\mathcal{C}_{\mathcal{Y}}(\mathcal{X}(\mathcal{Y}),\mathcal{Y})^{\times}$ may differ considerably from $\mathcal{C}_{\mathcal{Y}}(\mathcal{X}(\mathcal{Y}),\mathcal{Y})$, and while $\mathcal{C}_{\mathcal{Y}}$ is typically readily available to the user as part of the linearized state equation, the adjoint $\mathcal{C}_{\mathcal{Y}}^{\times}$ is typically not. In many cases, the computation of $\mathcal{C}_{\mathcal{Y}}^{\times}$ is difficult. Thus, not only the cost of solving the adjoint equation must be taken into account, but also the cost of implementing such solvers.

We are dealing with such a problem in Chapters 2, 3, and 4 of the present work. Our industrial partner relies on external software for the equation solve. This software can be used to furnish the evaluation of the state equation in dependence of the current control. The adjoint equation cannot be solved by the software package as is. Our goal in this application is not to newly code the equation solve and, additionally, the adjoint solve. Our goal is to provide optimization tools within the given setting.

Applicable optimization routines We use given software that solves the equations underlying the optimization. The matrix representing the discretization of the state equation is not explicitly stored. Instead, only equations are solved to furnish the desired state variables. Thus, it is not possible to extract the matrix from the code. This means that the discrete counterpart of $\mathcal{C}_{\mathcal{X}}(\mathcal{X},\mathcal{Y})$ is not available, but only the discrete version of $(\mathcal{C}_{\mathcal{X}}(\mathcal{X},\mathcal{Y}))^{-1}\mathcal{C}_{\mathcal{Y}}(\mathcal{X},\mathcal{Y})$. This piece of information could open up the possibility to build an approximation to the discrete version of the reduced Hessian of (1.1.2),

$$\mathcal{W}(\mathcal{X},\mathcal{Y})^{\times}\,\nabla^2_{\mathcal{X}\mathcal{X}}\mathcal{F}(\mathcal{X},\mathcal{Y})\;\mathcal{W}(\mathcal{X},\mathcal{Y}). \tag{1.1.7}$$

The exact (discretized) reduced Hessian

$$\mathcal{W}(\mathcal{X},\mathcal{Y})^{\times}\,\nabla^2_{\mathcal{X}\mathcal{X}}\Big(\mathcal{F}(\mathcal{X},\mathcal{Y}) + \mathcal{P}(\mathcal{X},\mathcal{Y})^{\times}\mathcal{C}(\mathcal{X},\mathcal{Y})\Big)\;\mathcal{W}(\mathcal{X},\mathcal{Y}) \tag{1.1.8}$$

is not available because it necessarily involves the adjoint variables. If $\mathcal{C}(\mathcal{X},\mathcal{Y})$ is linear in $\mathcal{X}$, the quantities (1.1.7) and (1.1.8) coincide. The reduced Hessian, or its approximation, could be used for a Newton approach. In our opinion, this is not advisable even if (1.1.7) and (1.1.8) coincide. Instead, we already handle first–order information, obtained via finite–differencing, with special care. A rather large error margin must be admitted for. This is due to simplifications in the modeling and to the

sparse data. The black–box type use of external software additionally introduces severe inaccuracies into the function. Nonsmoothness of the problem, in combination with several local minima that have large regions of attraction, see Section 4.3, easily traps conventional optimization algorithms. Such algorithms, e.g. globalized quasi–Newton methods, are usually based on the assumption of fully accurate functions, i.e., accuracy in the order of machine epsilon. They are not designed to deal with noisy functions. For this, the Implicit Filtering algorithm is appropriate. This method furnishes reliable results while investing only a small budget of function evaluations. We also employ the Nelder–Mead algorithm, a classical direct search method which typically also only requires a low number of function evaluations.

As was mentioned before, ground water hydraulic and quality management problems have not yet received much attention from the optimization community. Few approaches to such problems are reported in the literature. A review of several approaches to problems arising in this context, [24], [40], [41], [68], [83], is done in Section 2.3. To our knowledge, optimization methods for noisy functions have not yet been employed in the ground water modeling context. The treatment of our first application is developed and justified throughout Chapters 2, 3, and 4.

Outline of the two problems Our first application is a ground water quality management problem. Heated water is, after use for cooling purposes, reinjected into the ground. This leads to an increase in temperature at a set of drinking water wells which are located downstream relative to the injection well. German law, the Wasserhaushaltsgesetz, requires that anthropogenic changes of ground water properties be minimized. In this regulation is the requirement that drinking water be provided at the lowest temperature that is possible under undisturbed conditions. Our goal is to control the temperature at the drinking water wells. This is done via the optimization objective to minimize a quadratic function. The quadratic involves the pumping rates at a set of barrier wells as an approximate measure of cost. In addition, a linear combination of the pumping rates and the temperature at the drinking water wells is taken into account. This is a penalty for increased temperature. The temperature at the drinking water wells is obtained via the solution of the partial differential equations modeling ground water flow and heat transport in an aquifer.

Our second application is a ground water hydraulic management problem. Like the first problem, it is formulated as an optimal control problem governed by a partial differential equation. The boundary control problem occurs as a mixing problem when extracting ground water that originates from two different water bodies. A quadratic objective function models pumping cost and treatment cost for contaminated water. The constraint is the partial differential equation that models ground water flow in

an aquifer. The discretization of this problem is a quadratic programming problem with linear constraints. We approach this problem from a different angle than the first problem. A self–coded discretization enables us to pursue a full–system approach. For this approach, treated in Chapters 5, 6, and 7, we now provide the background.

KKT systems The optimization problems that we are concerned with are originally infinite–dimensional problems like (1.1.2). The variables are functions, and the constraint functions involve the solution of differential equations. The numerical solution of the problems requires a discretization of the variables. We then apply a finite–dimensional optimization method. A common choice for the numerical solution of the equality constrained problem

$$\min_{(\phi,u)} f(\phi,u) \quad \text{s.t.} \quad c(\phi,u)=0, \tag{1.1.9}$$

where

$$f : I\!R^{m+k} \to I\!R, \quad c : I\!R^{m+k} \to I\!R^m, \quad \phi \in I\!R^m, \quad u \in I\!R^k,$$

for appropriate $m,\ k \in I\!N$, are sequential quadratic programming methods (SQP). The basic idea of SQP methods is to model a nonlinear, nonlinearly constrained, problem at a given approximate solution by a quadratic, linearly constrained, subproblem. The solution to the subproblem is used to improve upon the current approximate solution. This process is iterated to create a sequence of approximations that, under certain assumptions, will converge to an optimizer of (1.1.9). Sequential quadratic programming, with an appropriate choice of quadratic subproblem and update strategy, can be viewed as the "extension of Newton's method and of quasi–Newton methods to the constrained optimization setting" [12]. Sequential quadratic programming is not so much a single method, but rather a conceptual method from which numerous specific algorithms have evolved. A vast body of literature is testimony to this. For an overview see e.g. [12]. The success of SQP methods depends on the existence of rapid and accurate algorithms for solving quadratic programs. We will see in the following short recapitulation of the sequential quadratic programming approach that here preconditioning comes in.

The so called full sequential quadratic programming method iterates on the full set of variables, i.e., state, control, and Lagrange multipliers. The Lagrangian function corresponding to problem (1.1.9) is, with a multiplier $\lambda \in I\!R^m$, given by

$$L(\phi,u,\lambda) = f(\phi,u) + \lambda^T c(\phi,u).$$

It is often required in the derivation of sequential quadratic programming methods that the second–order sufficient conditions for a minimum of problem (1.1.9) are satisfied,

see e.g. [12], [36, Ch. 3]. Our decision variable is naturally partitioned into the state and control variables, ϕ and u. We refer to the pair (ϕ, u) also as z. Because z is naturally partitioned we slightly strengthen the requirement that the gradient of the constraints with respect to z have linearly independent columns.

Assumption 1.1.1 *Assume that $f, c \in C^2(\mathbb{R}^{m+k})$. Let for a minimizer $z_* =: (\phi_*, u_*)$ of problem (1.1.9) the following hold.*

1. *The matrix $\nabla_\phi c(z_*)$ is invertible.*
2. *The Hessian of the Lagrangian function with respect to the variables $z = (\phi, u)$ is at the optimizer symmetric positive definite on the null space of the linearized constraints $\nabla_z c(z_*)^T$, i.e.,*

$$x^T \nabla^2_{zz} L(z_*, \lambda_*)\, x > 0 \quad \forall x \neq 0 \;\; \text{s.t.} \;\; \nabla_z c(z_*)^T x = 0.$$

Assumption 1.1.1 guarantees that z_* is an isolated local minimum of (1.1.9) and that a unique multiplier vector λ_* exists with

$$\nabla_z L(z_*, \lambda_*) = \nabla_z f(z_*) + \lambda_*^T \nabla_z c(z_*) = 0. \tag{1.1.10}$$

In the algorithm, such a minimizer z_* and the corresponding multiplier λ_* are usually approximated via the sequential solution of so called KKT systems. This can be derived with the help of Newton's method in the following way.

The system composed of the first order necessary optimality conditions for (1.1.9), i.e., (1.1.10), and of the constraints is given by the equations

$$L_\phi(\phi, u, \lambda) = \quad f_\phi(\phi, u) + c_\phi^T(\phi, u)\, \lambda \;\; = 0, \tag{1.1.11}$$

$$L_u(\phi, u, \lambda) = \quad f_u(\phi, u) + c_u^T(\phi, u)\, \lambda \;\; = 0, \tag{1.1.12}$$

$$L_\lambda(\phi, u, \lambda) = \qquad c(\phi, u) \qquad\quad = 0. \tag{1.1.13}$$

Newton's method

$$\nabla^2_{(\phi,u,\lambda)} L(\phi, u, \lambda) \begin{pmatrix} s_\phi \\ s_u \\ s_\lambda \end{pmatrix} = -\nabla_{(\phi,u,\lambda)} L(\phi, u, \lambda)$$

can be applied in order to solve the system of optimality conditions (1.1.11) – (1.1.13). In that instance, the system

$$\begin{pmatrix} L_{\phi\phi} & L_{\phi u} & c_\phi^T \\ L_{u\phi} & L_{uu} & c_u^T \\ c_\phi & c_u & 0 \end{pmatrix} \begin{pmatrix} s_\phi \\ s_u \\ s_\lambda \end{pmatrix} = - \begin{pmatrix} f_\phi + c_\phi^T \lambda \\ f_u + c_u^T \lambda \\ c \end{pmatrix}, \tag{1.1.14}$$

with all quantities evaluated at the current point (ϕ_c, u_c, λ_c), is then solved in consecutive steps. An obvious update strategy, compare [12], is with some step length α to set $(\phi_+, u_+, \lambda_+) = (\phi_c, u_c, \lambda_c) + \alpha\,(s_\phi, s_u, s_\lambda)$.

We note that (1.1.14) are the necessary and sufficient optimality conditions for the quadratic minimization problem

$$\min_{(s_\phi, s_u)} \tfrac{1}{2} \begin{pmatrix} s_\phi \\ s_u \end{pmatrix}^T \begin{pmatrix} L_{\phi\phi} & L_{\phi u} \\ L_{u\phi} & L_{uu} \end{pmatrix} \begin{pmatrix} s_\phi \\ s_u \end{pmatrix} + \begin{pmatrix} f_\phi \\ f_u \end{pmatrix}^T \begin{pmatrix} s_\phi \\ s_u \end{pmatrix}$$

subject to

$$\begin{pmatrix} c_\phi \\ c_u \end{pmatrix}^T \begin{pmatrix} s_\phi \\ s_u \end{pmatrix} = -c$$

provided that the second partial derivative L_{zz} of the Lagrangian function with respect to state and control (ϕ, u),

$$L_{zz} = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} \\ L_{u\phi} & L_{uu} \end{pmatrix}, \tag{1.1.15}$$

is positive definite on the null space of the linearization $(\, c_\phi \,|\, c_u \,)$ of the constraints.

In this work, we denote the linearly equality constrained quadratic programming problem in standard form, corresponding to a subproblem within a general sequential quadratic programming method, as

$$\min_{(\phi, u)} \tfrac{1}{2} \begin{pmatrix} \phi \\ u \end{pmatrix}^T \begin{pmatrix} L_{\phi\phi} & L_{\phi u} \\ L_{u\phi} & L_{uu} \end{pmatrix} \begin{pmatrix} \phi \\ u \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix}^T \begin{pmatrix} \phi \\ u \end{pmatrix} \tag{1.1.16}$$

subject to

$$A\phi + Bu = b. \tag{1.1.17}$$

We usually consider A to represent a discretized partial differential equation, corresponding to what was referred to as equation solve above. Thus, a solve with A is typically expensive and can be assumed to dominate overall computational cost. Also, A can be taken to be nonsingular. The matrix B represents the influence of the control u. The dimensions of the quantities are

$$L_{\phi\phi} \in I\!R^{m\times m},\ L_{\phi u}, L_{u\phi}^T \in I\!R^{m\times k},\ L_{uu} \in I\!R^{k\times k},\ A \in I\!R^{m\times m},\ B \in I\!R^{m\times k},$$
$$\phi \in I\!R^m,\ u \in I\!R^k,\ c \in I\!R^m,\ d \in I\!R^k,\ b \in I\!R^m.$$

In our context, m is the number of state variables. It is typically considerably larger than the number k of control variables. This is especially true since we consider boundary control problems. We also use $C = (A \mid B)$ to refer to the constraints (1.1.17). With this notation, the null space of the constraints is handily written with the help of

$$W = \begin{pmatrix} -A^{-1}B \\ I_k \end{pmatrix}. \tag{1.1.18}$$

This null space representation is canonical for problems of optimal control. The reduced Hessian H of (1.1.16) s.t. (1.1.17) is given as the projection of (1.1.15) onto the null space of the constraints,

$$H = W^T L_{zz} W = B^T A^{-T} L_{\phi\phi} A^{-1} B - L_{u\phi} A^{-1} B - B^T A^{-T} L_{\phi u} + L_{uu} \,. \quad (1.1.19)$$

The existence of solutions of the quadratic programming problem (1.1.16) s.t. (1.1.17) is guaranteed if the matrix in (1.1.15) is positive semidefinite and if the objective function is bounded from below on the set of feasible points. In this case, the necessary and sufficient optimality conditions for problem (1.1.16) s.t. (1.1.17) are given by the so called KKT conditions, see e.g. [36, Ch. 3],

$$\begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix} \begin{pmatrix} \phi \\ u \\ \lambda \end{pmatrix} = \begin{pmatrix} -c \\ -d \\ b \end{pmatrix}. \quad (1.1.20)$$

Consistently throughout this work, we refer to the system matrix of (1.1.20) as K,

$$K = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix}. \quad (1.1.21)$$

The dimension of K is $2m + k$ for which we will consistently use N. We set the following three requirements on K and the submatrices of the KKT system. These are not in full correspondence to Assumption 1.1.1, but present a relaxation.

Assumption 1.1.2 *Let for the system K in (1.1.21) and its submatrices hold:*

1. *K is nonsingular.*
2. *K is symmetric.*
3. *A is nonsingular.*

The second item of Assumption 1.1.2 is an immediate consequence of the requirement that f and c be twice continuously differentiable. Note that we do not require symmetric positive definiteness of $L_{\phi\phi}$, L_{uu}, or L_{zz}. The third item states solvability of the state equation and corresponds to the first item of Assumption 1.1.1.

Note that we do not set the requirement corresponding to the second item of Assumption 1.1.1 throughout our treatment. The vast majority of our results need not require that L_{zz} is symmetric positive definite on the null space of the constraints C, i.e., that the reduced Hessian is symmetric positive definite. Neither do we require that L_{zz} be invertible and that consequently the Schur complement of K exist. We will see in Chapter 6 that these assumptions are commonly made in the literature. Properties of the KKT system that can be derived from Assumption 1.1.2 and that are relevant for our treatment are assembled in Section 5.2.1.

Preconditioning From the foregoing short recapitulation it follows that in each step of sequential quadratic programming methods, a linear system

$$K\,x = r \tag{1.1.22}$$

with $K \in I\!R^{N\times N}$ ($N = 2m + k$) symmetric indefinite has to be solved. It is crucial for overall performance to do that efficiently. Whenever discretizations of partial differential equations are involved, the KKT systems generally tend to large dimensions and exhibit a specific sparse structure. The use of iterative solvers is usually advocated under these circumstances. Iterative methods are advantageous in that they do not require to assemble and store the entire system K, but, instead, only a matrix–vector multiplication $K\,v$ per iteration. Common choices are so called Krylov subspace methods. Two especially well–known Krylov subspace methods are the conjugate gradient routine, see e.g. [39], and the generalized minimum residual method, GMRES [70]. Typically, if no restart occurs, Krylov subspace methods generate iterates

$$x_j \in x_0 + \mathcal{K}_j(K, r_0) \equiv span\left\{K^l r_0\right\}_{l=0}^{j-1},$$

where $\mathcal{K}_j(K, r_0)$ denotes the Krylov subspace of order j generated by K and the initial residual r_0. Initial residual means that $r_0 = K\,x_0 - r$ for a starting guess x_0. A number of Krylov subspace methods, most prominently the conjugate gradient, the minimum residual, and the generalized minimum residual algorithms, are N–step procedures in exact arithmetic in that they stop with the exact solution of (1.1.22) after at most N iterations. This is in general judged a too high computational expense. It is customary to regard Krylov subspace methods as genuinely iterative procedures with termination based upon an iteration maximum and the residual norm $||r_j||$. Convergence analysis is the basis for the construction of preconditioners. Preconditioners are often that convergence–enhancing tool that renders iterative methods efficient. The general issue in preconditioning is to construct a system that is equivalent to the original system and that is easier to solve. The ideal preconditioner can be thought of to approximate the inverse of the original system matrix. But the use of the exact inverse is in general prohibitively expensive. Clearly, cost and effectiveness of a preconditioner have to be weighted against each other.

Our interest in this work is the construction of block preconditioners. Block preconditioners are e.g. considered in [5], [7], [34], [49], [54], [69], [73], [74]. This previous related work is reviewed in Chapter 6. Block preconditioners are here to be understood such that the preconditioners are composed of blocks and that they maintain the block structure of the original system. This allows to exploit that structure to computational advantage and to specifically address the problematic parts of the original system. In the ideal case, the blocks in the preconditioner are the inverses

of the submatrices of the original system (or of products of the submatrices). But it can usually not be assumed that the exact solution is affordable. In the general case, the preconditioner for the full system is composed of preconditioners for the submatrices. Such block preconditioners, composed of preconditioners for submatrices, can generally be adapted to specific applications by incorporating known effective preconditioners for subsystems of the system matrix.

The preconditioner proposed in this work is for suitable P_A and P_H given by

$$\tilde{K} = \begin{pmatrix} 0 & 0 & P_A^T \\ 0 & P_H & B^T \\ P_A & B & 0 \end{pmatrix}.$$

Obviously, the structure of K is well captured in $\tilde{K}$. The preconditioner is indefinite like the original K in (1.1.21). Thus, all eigenvalues of the preconditioned system $\tilde{K}^{-1}K$ lie on one side of the origin. They are also well separated from the origin. The preconditioned system is not normal, i.e., not unitarily diagonalizable. Although convergence behavior of iterative methods is in the nonnormal case not necessarily well described by the eigenvalues, eigenvalue information does prove useful for the system $\tilde{K}^{-1}K$. An upper limit to the number of GMRES steps is the degree of the minimum polynomial of the system.

A clear–cut analysis of the minimum polynomial is performed for the case where the exact constraints C are maintained. This corresponds to the choice $P_A = A$. Realistically, the admission of approximate constraints $P_A \approx A$ in the preconditioner is highly desirable. The relaxation to admit P_A is a generalization of previous work and represents an important step towards computational efficiency. The eigenvalue analysis is disturbed in this case, but can be qualitatively maintained.

The starting point for analyzing the eigenvalue distribution is to consider the case where the exact constraints are maintained, $P_A = A$. In this situation, the eigenvalues of the preconditioned system are the desired eigenvalue 1 and at most k eigenvalues that are distinct from 1. The key observation is that the request eigenvalue 1 has high algebraic multiplicity $2m$ and low geometric multiplicity 2. This delays convergence of the generalized minimum residual method by only one step. The remaining k eigenvalues are those of $P_H^{-1}H$. The matrix H, see (1.1.19), is the reduced Hessian of the underlying quadratic programming problem (1.1.16) s.t. (1.1.17). Choosing $P_H = H$ lets GMRES terminate with the exact solution (in exact arithmetic) after 3 steps. The cheaper and computationally sensible choice $P_H = I$ can be shown to permit capturing of an invariant subspace of the system after at most $k + 2$ steps. In comparison to the almost N steps that are needed by Krylov subspace methods on the ill–conditioned original system, $k + 2$ steps, with $k \ll m$, represent major improvement. The pre-

conditioner is especially successful if the number k of controls is small. For instance, expectations can be set high with boundary control problems.

1.2 Overview

The present work is structured as follows.

In Chapter 2 the basis is laid for the first application considered in this work. The relevant equations are derived. The partial differential equations model flow and heat transport in a porous medium. Commonly used simplifications are explained that make the equations practical to handle. The application is formulated as a problem of optimal control that is governed by partial differential equations. Also, the choice of formulation is justified. A review, by no means complete, of previous approaches to the combination of aquifer simulations with optimization is appended.

Anticipatory to the numerical results for the application that are presented in the subsequent chapter, we state in Chapter 3 that a conventional smooth optimizer is easily trapped in this application's landscape. We turn our attention to two specialized optimization routines, Implicit Filtering and the Nelder–Mead algorithm. These we employ for the considered application. The routines are presented along with theoretical background concerning inaccurate function evaluations.

In Chapter 4 our first practical application is described in–depth with its hydrogeologic features. Subsequently, the numerical setting of the discrete optimization problem is explored. This is tied to software issues. To the codes that are used to solve the partial differential equations underlying the optimization we refer as the simulation codes. The simulation codes are briefly reviewed to explain their main features. Finally, the optimization results for the case study are described and analyzed.

We address our second application in the consecutive chapters. In our numerical treatment of this application, all ingredients for the full–system approach are provided. This enables us to solve the linear quadratic control problem via the solution of one linear system, the KKT system. For the linear solve, we use iterative methods — but only in conjunction with an effective preconditioner. Chapter 5 is devoted to the analysis of the preconditioner that we propose upon the solution of large sparse KKT–like systems with features similar to that of our application. The preconditioner's cost and effects are discussed, this including implementational issues and considerations for the choice of subpreconditioners. As prerequisites, preconditioning and a number of Krylov subspace methods are reviewed.

A review of several preconditioners for symmetric indefinite systems is added in the following chapter, Chapter 6. An abundance of literature can be found, both concerning all–purpose preconditioners and application–specific preconditioners. We do

not pretend to attempt a complete survey. Nevertheless, besides pointing to all–purpose preconditioners that are suitable for indefinite systems, we refer to a large group of block preconditioners that are designed in a similar way as the preconditioner of Chapter 5. A number of the assembled preconditioners are consequently examined in detail.

The derivation of the application, a ground water hydraulic problem, is done in Chapter 7. Its discretization is a quadratic function governed by a linear partial differential equation. The KKT system is then set up. The solution to the optimization problem is discussed. The numerical performance of the preconditioner analyzed in Chapter 5 is compared to a number of preconditioners reviewed in the preceding chapter. Also treated at large is the preconditioner's performance on this example problem in several configurations of subpreconditioners.

In addition to the summarizing comments in Sections 4.3 and 7.3, conclusions are drawn in Chapter 8. Possible extensions to this work are sketched, both in terms of the applications and of the specific optimization approaches.

Finally, the Appendix is designed to provide information about aquifers and ground water flow and to relate some of the basic definitions and assumptions used in the derivation of the common model.

Chapter 2

An Optimal Control Problem

In this chapter we derive and state an optimal control problem governed by partial differential equations. The motivation behind this problem is the application that we address in Chapter 4 and only briefly delineate here: Our focus is on the drinking water supply of a region. The considered zone is the recharge zone for a cluster of drinking water wells. Within this zone lies an industrial zone with several additional wells, mostly also extracting water. Besides, reinjection of heated water into the ground occurs, this causing a temperature anomaly in the ground water. The goal is to hold low increases in the ground water temperature at the water supply wells. Control will be exerted via selected wells.

In this chapter we derive how the problem described above in very short terms can be modeled mathematically. The model involves two partial differential equations. One partial differential equation models the ground water motion. The second partial differential equation models heat transport in water and soil. In the general formulation, ground water flow and heat transport in the ground are interdependent. The problem thus requires the solution of a coupled system of two partial differential equations. In order to derive a practical formulation, several simplifications are introduced which we develop throughout the chapter.

2.1 Ground Water Flow

This section deals with the basic law governing the motion of ground water in aquifers. Only saturated flow is considered, and all variables and parameters have an average meaning in a porous medium regarded as a continuum. The flow equation is based on Darcy's law and a continuity equation. The combination of these two describes flow in an aquifer. The mathematics of this section can be understood without a hydrologic derivation of the equations. Nevertheless, the most important notions, e.g. the continuum approach and the hydraulic approach as well as Darcy's law, can be looked up in the Appendix. Also addressed there is the range of validity of Darcy's law.

The Flow Equation

The basic differential equation for ground water flow in aquifers is given by

$$S_0 \frac{\partial}{\partial t}\phi(x,y,z,t) = \nabla \cdot (K(x,y,z)\,\nabla\phi(x,y,z,t)) + \hat{q}^{\phi}(x,y,z,t), \tag{2.1.1}$$

where $(x,y,z) \in \hat{\Omega} \subset \mathbb{R}^3$ are the spatial variables and $t \in [0,t_1]$ denotes time. The solution to Equation (2.1.1) is the piezometric head ϕ, $\phi = \phi(x,y,z,t)$. The coefficient K which depends on the spatial variables (x,y,z) is a coefficient of proportionality and is called hydraulic conductivity. The hydraulic conductivity expresses the ease with which a fluid is transported through a porous medium. Depending on both medium and fluid properties, the hydraulic conductivity K can be a scalar or a (3×3)–tensor with values that are constant or that vary in space. Compare the Appendix. The specific storativity S_0 of the porous medium of an aquifer is the volume of water released from storage or added to it in a unit volume of aquifer per unit change in piezometric head. The term $\hat{q}^{\phi}(x,y,z,t)$ accounts for sources and sinks of water in the considered domain.

For a homogeneous isotropic medium, the hydraulic conductivity $K(x,y,z)$ is a scalar value depending on the spatial variables, $k_f(x,y,z) \in \mathbb{R}$. In case of constant hydraulic conductivity K or k_f, Equation (2.1.1) reduces to

$$S_0 \frac{\partial}{\partial t}\phi(x,y,z,t) = k_f\,\Delta\phi(x,y,z,t) + \hat{q}^{\phi}(x,y,z,t) \tag{2.1.2}$$

with the Laplace operator $\Delta\phi = \nabla \cdot (\nabla\phi)$.

When both fluid and solid are assumed to be incompressible, the storage term S_0 in Equations (2.1.1) and (2.1.2) vanishes. This also happens when the flow is steady. In that instance, the time–dependent equation (2.1.2) reduces to the Laplace equation,

$$k_f\,\Delta\phi(x,y,z) = -\hat{q}^{\phi}(x,y,z). \tag{2.1.3}$$

Solution of any of the Equations (2.1.1) to (2.1.3) allows to compute so called Darcy velocity v_D,

$$v_D(x,y,z,t) = -K(x,y,z)\,\nabla\phi(x,y,z,t).$$

Since flow takes place only through part of the porous medium, the remaining part being occupied by the solid (compare the Appendix), the effective porosity n is needed to determine the field velocity v,

$$v(x,y,z,t) = \frac{1}{n}\,v_D(x,y,z,t) = -\frac{1}{n}\,K(x,y,z)\,\nabla\phi(x,y,z,t).$$

In this work, we do not distinguish between effective and total, or average, porosity, but, for simplification, always refer to the effective porosity n.

Problem–Oriented Simplifications

In this part those problem–oriented simplifications are summarized which allow to develop a practical ground water flow model. See also the Appendix.

- ◇ The continuum approach is employed, i.e., all variables and parameters have an average meaning in a porous medium that is regarded as a continuum.
- ◇ Incompressibility of fluid and soil as well as constant density and viscosity of the fluid are assumed.

These two important assumptions are already incorporated into the preceding equations. Another important simplification allows to reduce the three–dimensional partial differential equation to an equation in two dimensions.

- ◇ The Dupuit assumption for an unconfined aquifer is also known as the hydraulic approach. It is equivalent to assuming that equipotential surfaces are strictly vertical. This is based on the fact that flow in aquifers is essentially horizontal. For a justification see the Appendix. We write

$$h(x,y,t) = \phi(x,y,.,t) \quad (x,y) \in \Omega, t \in [0,t_1]\,.$$

Thus, Equation (2.1.1) can be reduced to

$$S\,\frac{\partial}{\partial t}h(x,y,t) = \nabla \cdot (K(x,y)\,\nabla h(x,y,t)) + q^h(x,y,t) \tag{2.1.4}$$

with spatial variables $(x,y) \in \Omega \subset I\!R^2$ and time $t \in [0,t_1]$. The solution to Equation (2.1.4) is the piezometric head h, $h = h(x,y,t)$. As before, compare also the Appendix, $q^h(x,y,t)$ accounts for sources and sinks, S is a storage coefficient, and $K(x,y)$ is the hydraulic conductivity. Analogous changes apply to Equations (2.1.2) and (2.1.3). For a homogeneous isotropic medium, write

$$S\,\frac{\partial}{\partial t}h(x,y,t) = \nabla \cdot (k_f(x,y)\,\nabla h(x,y,t)) + q^h(x,y,t)\,. \tag{2.1.5}$$

If flow is steady, time–dependency vanishes so that Equation (2.1.5) reduces to

$$\nabla \cdot (k_f(x,y)\,\nabla h(x,y,t)) = -q^h(x,y,t)\,. \tag{2.1.6}$$

The distribution of $h = h(x,y,t)$ in an aquifer is obtained by solving Equation (2.1.4) subject to appropriate boundary and initial conditions. Knowledge of h then allows to compute the field velocity v, with porosity n as before,

$$v(x,y,t) = -\frac{1}{n}\,K(x,y)\,\nabla h(x,y,t). \tag{2.1.7}$$

Further Conditions

Each of the Equations (2.1.1) to (2.1.3) and (2.1.4) to (2.1.6) presented in this section is a partial differential equation which describes a class of phenomena. To obtain a particular solution corresponding to a specific problem, it is necessary to provide supplementary information. These must include:

- ◇ The geometry of the domain in which the considered flow takes place.
- ◇ Values of all relevant physical coefficients.
- ◇ Specifications of the initial conditions to describe the initial state of the fluid in the considered flow domain.
- ◇ Specifications of the boundary conditions to describe how the fluid in the considered domain interacts with its surroundings.

Initial conditions are the specification of h at all points within the domain Ω at some initial time t, usually taken as $t = 0$. This can be written, with h_0 a known function, as $h(x, y, 0) = h_0(x, y)$ in Ω.

Boundary conditions describe the flow on the boundary of the considered domain. Three main types of boundary conditions are generally distinguished, the Dirichlet boundary conditions, Neumann boundary conditions, and the mixed or Cauchy boundary conditions. See the Appendix also. We consider boundary parts Γ_D, Γ_N, and Γ_M, with $\Gamma_D \cap \Gamma_N \cap \Gamma_M = \emptyset$, $\Gamma_D \cup \Gamma_N \cup \Gamma_M = \partial\Omega$. Let $h_D(x, y, t)$, $h_N(x, y, t)$, and $h_M(x, y, t)$ be real–valued functions defined on $\Gamma_D \times [0, t_1]$, $\Gamma_N \times [0, t_1]$, and $\Gamma_M \times [0, t_1]$, respectively. Then Equation (2.1.4) is, complete with boundary and initial conditions, given in the following form.

$$
\begin{array}{rcll}
S\,\frac{\partial}{\partial t}h(x,y,t) & = & \nabla\cdot(K(x,y)\,\nabla h(x,y,t)) & \\
 & & +q^h(x,y,t) & (x,y,t)\in\Omega\times[0,t_1] \\
h(x,y,t) & = & h_D(x,y,t) & (x,y,t)\in\Gamma_D\times[0,t_1] \\
\frac{\partial}{\partial n}h(x,y,t) & = & h_N(x,y,t) & (x,y,t)\in\Gamma_N\times[0,t_1] \\
h(x,y,t) & & & \\
+\frac{\partial}{\partial n}h(x,y,t) & = & h_M(x,y,t) & (x,y,t)\in\Gamma_M\times[0,t_1] \\
h(x,y,0) & = & h_0(x,y) & (x,y)\in\Omega
\end{array}
\tag{2.1.8}
$$

2.2 Heat Transport

We now address the partial differential equation that describes heat transport in an aquifer. The main mechanisms affecting this transport are heat conduction, convection, and hydrodynamic dispersion. Convective heat transport is described on the basis of the ground water flow introduced in Section 2.1. This requires knowledge of the flow regime prior to solving the transport equation. In the general case, temperature changes can influence fluid and soil properties. Variations in the temperature field

cause changes in the liquid's density and viscosity. These, in turn, affect the flow regime that depends on these properties. Thus we are faced with a coupled system of two partial differential equations. Its numerical solution is in most cases impractical. We introduce and justify those simplifications that entail the sequential solve of the equations. Subsequently, a comparison is drawn between the equations that model heat and solute transport in an aquifer. This comparison leads to a discussion of the equation parameters that are relevant for the application in Chapter 4. The parameters determine the character of the partial differential equations under consideration, this being the last point discussed in this section.

The Transport Equation

The main mechanisms affecting heat transport in a porous medium are heat conduction, convection, and hydrodynamic dispersion. Heat conduction is the thermal exchange between water and solid in the aquifer. The model that is developed here is based on the commonly made assumption that, through conduction, an initial difference in temperature between fluid and solid disappears instantaneously. Convection describes heat transport with the flow of water. The term advection is used synonymously to convection in this work. This phenomenon can be described on the basis of the average flow of ground water. Also, conduction can be described on the macroscopic level. On the microscopic level, because of the shape of the interconnected pore space, velocity variations in magnitude and direction occur that cause a spreading of particles. This is called mechanical dispersion or convective diffusion. The term hydrodynamic dispersion is used to denote on the macroscopic level the spreading that results from both mechanical dispersion and molecular diffusion. However, heat transport through molecular diffusion can for a porous medium flow in general be neglected. Its contribution is of interest only for flow velocities well below those that are usually exhibited, compare e.g. [71, p. 22]. The presentation in this section is based on [9, Ch. VII], [25, Ch. 10.3], [28], [57, Ch. 4.2], [76], [77].

The equation modeling heat transport in an aquifer is given by

$$\begin{aligned} \tfrac{\partial}{\partial t}T(x,y,z,t) = &-\nabla\cdot(\kappa(T)\,T(x,y,z,t)\,v(x,y,z,t)) \\ &+\nabla\cdot(D(v)\,\nabla T(x,y,z,t)) + \hat{q}^T(x,y,z,t) \end{aligned} \tag{2.2.1}$$

with spatial variables $(x,y,z) \in \hat{\Omega} \subset I\!R^3$ and time $t \in [0,t_1]$. The solution to (2.2.1) is the temperature T, $T = T(x,y,z,t)$. Sources and sinks of heat are modeled by the term $\hat{q}^T = \hat{q}^T(x,y,z,t)$. The first–order term in (2.2.1) describes the convective transport, and the second–order term models the dispersive phenomena. The ground water flow enters the equation through the velocity v, $v = v(x,y,z,t)$, directly in the convective term. The flow regime also influences the dispersive term via the effective heat dispersion coefficient of Equation (2.2.1), compare [77],

$$D(v) = D_e(v) + \frac{\lambda_e}{\rho_a c_a} I\,. \tag{2.2.2}$$

In (2.2.2), λ_e is the effective thermal conductivity of the aquifer with units $[J/smK]$, $\rho_a c_a$ is the heat capacity of the aquifer with units $[J/m^3K]$, and $D_e = D_e(v)$ is the tensor of hydrodynamic dispersion with units $[m^2/s]$. The coefficient $\kappa(T)$ in Equation (2.2.1) is given by

$$\kappa(T) = \frac{n\,\rho_w c_w(T)}{\rho_a c_a}\,. \tag{2.2.3}$$

Here, $\rho_w c_w$ is the heat capacity of water with units $[J/m^3K]$. The heat capacity of the aquifer $\rho_a c_a$ is the average of the heat capacities of water $\rho_w c_w$ and solid $\rho_s c_s$, weighted with the porosity n, see e.g. [25, p. 278],

$$\rho_a c_a = n\,\rho_w c_w + (1-n)\,\rho_s c_s\,.$$

Since the density ρ_w of water depends on the temperature T, we write $\rho_w = \rho_w(T)$ and denote the dependency by $\kappa(T)$. Not only the density ρ_w of water, but also its kinematic viscosity ν_w depends on the temperature T, so that $\nu_w = \nu_w(T)$ also. We recall the definition of hydraulic conductivity K, compare the Appendix, in the homogeneous, isotropic case as the scalar

$$k_f = \frac{\rho_w\, g\, k_0}{\nu_w}\,.$$

Thus, we denote K which was known to depend on the spatial variables, i.e., $K = K(x,y,z)$, now as $K(x,y,z,T) = K(.,T)$. We see with this symbolic notation that (2.1.1) and (2.2.1) are interdependent and that their combination

$$\begin{aligned} S_0\,\tfrac{\partial}{\partial t}\phi(.) &= \nabla\cdot(K(.,T)\,\nabla\phi(.)) + \hat{q}^{\phi}(.) \\ v(.) &= -\tfrac{1}{n}\,K(.,T)\,\nabla\phi(.) \\ \tfrac{\partial}{\partial t}T(.) &= -\nabla\cdot(\kappa(T)\,T(.)\,v(.)) + \nabla\cdot(D(v)\,\nabla T(.)) + \hat{q}^{T}(.) \end{aligned} \tag{2.2.4}$$

constitutes a coupled system of two partial differential equations with nonconstant coefficients. The numerical solution of such a system is currently in most cases impractical. Compare e.g. [48], [67].

In this short description of the coupled system we have so far neglected initial conditions and boundary conditions. Of course, initial and boundary conditions are necessary to solve the partial differential equations in (2.2.4). This leads to an additional point requiring a simplification of the considered equations. In practice, many parameter values and boundary conditions are unknown. For instance, the natural temperature distribution of ground water cannot be determined exactly. Instead, in view of the accessible data and of numerical practicality, a simplified model is considered. This is now addressed.

Problem–Oriented Simplifications

In this part we describe those problem–oriented simplifications which allow to develop a practical heat transport model for the determination of anthropogenic temperature

anomalies in ground water. Facts assembled in [25], [71], [76], [77] justify the simplifications for the application that we have in mind. The most important simplifications focus on the coupling between flow and temperature.

- ◇ Density effects due to temperature changes are not considered. That means that the density ρ_w is constant.
- ◇ Changes in viscosity due to temperature changes are negligibly small. Thus, the viscosity ν_w is constant.

This allows the decoupling of the two partial differential equations in (2.2.4). We consider the hydraulic conductivity as dependent on the spatial variables alone, i.e., $K = K(x, y, z)$, and we consider the coefficient κ defined in (2.2.3) to be constant.

In addition, we make several assumptions that have implications on the modeling and on the numerical treatment of the problem. The specific characteristics of heat transport in the subsurface, including lack of data, usually require problem–oriented simplifications in both model concepts and model dimensionality.

- ◇ Heat storage and conductivity, i.e., interchanges with the atmosphere and the aquifer basis, can be represented by a term for sources and sinks. Also, heat exchange processes in the cover layer, e.g. due to anthropogenic influences, can be represented by a term for sources and sinks of heat.
- ◇ Due to the exponential decay of temperature changes with increasing depth below ground and to the long–term nature of thermal processes in ground water it is in general admissible to equate the natural ground water temperature with the annually averaged air temperature.
- ◇ Due to the exponential decay of temperature changes with increasing depth below ground and to the long–term nature of thermal processes in ground water it is in general admissible to neglect the vertical dimension and treat the heat transport process in two spatial variables in the horizontal plane.
- ◇ Temperature anomalies in ground water require several years to decades to develop. This justifies to rely on successive flow fields that are averaged over time.

It is for instance in [27, Ch. 9] established that the effective perturbation depth of temperature fluctuations at the Earth's surface is on the order of $10m$. No appreciable temperature changes in response to seasonal variations are exhibited in our application.

The application of these simplifications to (2.2.1) leads to

$$\begin{aligned} \frac{\partial}{\partial t}T(x,y,t) &= -\nabla \cdot (\kappa T(x,y,t)\, v(x,y,t)) \\ &\quad +\nabla \cdot (D(v)\, \nabla T(x,y,t)) + q^T(x,y,t) \end{aligned} \tag{2.2.5}$$

with spatial variables $(x, y) \in \Omega \subset \mathbb{R}^2$ and time $t \in [0, t_1]$. The coefficients are

$$\kappa = \frac{n\,\rho_w c_w}{\rho_a c_a}, \quad D(v) = D_e(v) + \frac{\lambda_e}{\rho_a c_a} I \tag{2.2.6}$$

with $D_e(v)$ defined by

$$D_e(v) = \frac{1}{\|v\|} \begin{pmatrix} \alpha_L\, v_x^2 + \alpha_T\, v_y^2 & (\alpha_L - \alpha_T) v_x v_y \\ (\alpha_L - \alpha_T) v_x v_y & \alpha_T\, v_x^2 + \alpha_L\, v_y^2 \end{pmatrix}. \tag{2.2.7}$$

In (2.2.5), α_T stands for transversal dispersion length and α_L for longitudinal dispersion length. The quantities v_x and v_y are the entries of the velocity vector v. Compare e.g. [9, Ch. 7], [25, Ch. 10.3], [57, p.65ff.], [71, p. 19ff.].

Ground water flow enters the equation through the velocity v as defined in (2.1.7). Sources and sinks of heat are modeled by the term $q^T(.)$. The solution h to (2.1.5) is, under the assumptions stated above, independent of the temperature field T. This allows the sequential treatment of the two equations.

In order to define a solvable problem, Equation (2.2.5) must be complemented with initial and boundary conditions. Three main types of boundary conditions are distinguished, Dirichlet, Neumann, and mixed boundary conditions. Consider boundary parts Γ_D, Γ_N, and Γ_M, with $\Gamma_D \cap \Gamma_N \cap \Gamma_M = \emptyset$, $\Gamma_D \cup \Gamma_N \cup \Gamma_M = \partial\Omega$. Consider also real–valued functions $T_D(x,y,t)$, $T_N(x,y,t)$, and $T_M(x,y,t)$ defined on $\Gamma_D \times [0,t_1]$, $\Gamma_N \times [0,t_1]$, and $\Gamma_M \times [0,t_1]$, respectively. Then Equation (2.2.5) takes the following form.

$$\begin{aligned}
\tfrac{\partial}{\partial t}T(x,y,t) &= -\nabla\cdot(\kappa\, T(x,y,t)\, v(x,y,t)) \\
&\quad +\nabla\cdot(D(v)\,\nabla T(x,y,t)) \\
&\quad +q^T(x,y,t) && (x,y,t)\in\Omega\times[0,t_1] \\
T(x,y,t) &= T_D(x,y,t) && (x,y,t)\in\Gamma_D\times[0,t_1] \\
\tfrac{\partial}{\partial n}T(x,y,t) &= T_N(x,y,t) && (x,y,t)\in\Gamma_N\times[0,t_1] \\
T(x,y,t) & \\
+\tfrac{\partial}{\partial n}T(x,y,t) &= T_M(x,y,t) && (x,y,t)\in\Gamma_M\times[0,t_1] \\
T(x,y,0) &= T_0(x,y) && (x,y)\in\Omega
\end{aligned} \tag{2.2.8}$$

The combination of Equations (2.2.8) and (2.1.8) that allows the sequential solve can now be cast in the following form. First, compute the piezometric head h from the flow equation.

$$\begin{aligned}
S\,\tfrac{\partial}{\partial t}h(x,y,t) &= \nabla\cdot(K(x,y)\,\nabla h(x,y,t)) \\
&\quad +q^h(x,y,t) && (x,y,t)\in\Omega\times[0,t_1] \\
h(x,y,t) &= h_D(x,y,t) && (x,y,t)\in\Gamma_D\times[0,t_1] \\
\tfrac{\partial}{\partial n}h(x,y,t) &= h_N(x,y,t) && (x,y,t)\in\Gamma_N\times[0,t_1] \\
h(x,y,t) & \\
+\tfrac{\partial}{\partial n}h(x,y,t) &= h_M(x,y,t) && (x,y,t)\in\Gamma_M\times[0,t_1] \\
h(x,y,0) &= h_0(x,y) && (x,y)\in\Omega
\end{aligned}$$

Second, determine the field velocity v.

$$v(x,y,t) = -\tfrac{K(x,y)}{n}\,\nabla h(x,y,t) \qquad (x,y,t)\in\Omega\times[0,t_1]$$

Third, get the desired temperature distribution as the solution of the transport equation.

$$\begin{aligned}
\tfrac{\partial}{\partial t}T(x,y,t) &= -\nabla\cdot(\kappa T(x,y,t)\, v(x,y,t)) \\
&\quad +\nabla\cdot(D(v)\,\nabla T(x,y,t)) \\
&\quad +\hat{q}^T(x,y,t) && (x,y,t)\in\Omega\times[0,t_1] \\
T(x,y,t) &= T_D(x,y,t) && (x,y,t)\in\Gamma_D\times[0,t_1] \\
\tfrac{\partial}{\partial n}T(x,y,t) &= T_N(x,y,t) && (x,y,t)\in\Gamma_N\times[0,t_1] \\
T(x,y,t) & \\
+\tfrac{\partial}{\partial n}T(x,y,t) &= T_M(x,y,t) && (x,y,t)\in\Gamma_M\times[0,t_1] \\
T(x,y,0) &= T_0(x,y) && (x,y)\in\Omega
\end{aligned}$$

We have now established the equation modeling heat transport in that form that we use in our numerical treatment. The numerical treatment is described in Chapter 4. This includes a description of the software code that we use to solve the transport equation. The code has been designed for solute transport. The justification for its use follows.

Heat versus Solute Transport

This part addresses the partial differential equations that model heat and solute transport in an aquifer, respectively. We will see that the two equations are similar and that a numerical model that solves one of the equations can be readily adapted to solve the other equation as well. Compare [8, Ch. 10.7], [25, Ch. 10.3], [71, p. 19f.].

Heat transport was already investigated above. Recall that the main mechanisms affecting heat transport in a porous medium are heat conduction, convection, and hydrodynamic dispersion. The equation that models heat transport in an aquifer is, neglecting initial and boundary conditions, given by

$$\tfrac{\partial}{\partial t}T(.) = -\nabla\cdot(\kappa T(.)\, v(.)) + \nabla\cdot(D(v)\,\nabla T(.)) + \hat{q}^T(.)\,. \tag{2.2.9}$$

The solution to (2.2.9) is the temperature $T(.)$. The ground water flow enters the equation through the velocity $v(.)$ directly in the convective part and also via the tensor $D_e(v)$ of hydrodynamic dispersion defined in Equation (2.2.7). Notation is used in this section as before. The coefficients κ and $D(v)$ are defined in Equation (2.2.6). We recall that n denotes porosity, that λ_e is the effective thermal conductivity of the aquifer with units $[J/smK]$, and that $\rho_w c_w$ and $\rho_a c_a$ are the heat capacity of water and of the aquifer, respectively, with units $[J/m^3K]$.

The movement of a solute in an aquifer is determined by three main mechanisms, by diffusion, convection, and dispersion. Again, we use the term advection synonymously to convection. In addition to these main physical phenomena there will in general be other physical, chemical, and biological reactions. These are usually taken into account with a term $\hat{q}^s$ for sources and sinks of mass. A tracer is a water–soluble

substance that does not affect the physical and chemical properties of water. The transport of a tracer is described by

$$\frac{\partial}{\partial t}C(.) = -\nabla \cdot (C(.)\, v(.)) + \nabla \cdot ((d\, I + D_e(v))\, \nabla C(.)) + \hat{q}^s(.)\,. \qquad (2.2.10)$$

Here, C stands for volumetric solute concentration in mobile phase with units $[kg/m^3]$, while d is the coefficient of molecular diffusion with units $[m^2/s]$.

The preceding Equation (2.2.10) is the basis for the numerical model underlying the transport model `MT3D`. This is the transport model that we used in our numerical experiments. It is described in detail in Section 4.2.

Transport of heat energy in ground water is thus described in (2.2.9) by convective transport and a phenomenon similar to dispersion of a solute. Compare Equations (2.2.9) and (2.2.10). The dispersive part comprises the thermal conductivity of both phases, fluid and solid, corresponding to molecular diffusion in the solute transport case, and a "fictive" thermal conductivity that is caused by velocity variations of flow in the pore space, leading to a mixing due to aquifer inhomogeneities. The dispersive transport rates of heat and solute transport are comparable, see [71, p. 20]. This allows to adjust input to the solute transport model through replacing d by $\lambda_e/\rho_a c_a$.

In a domain where convective transport is dominant, the simultaneous transport of solute and heat shows a retardation of the heat front versus the front of a conservative tracer. This is due to the heat capacity of the aquifer. The relationship between the velocities of the heat and tracer fronts can be estimated with the help of the porosity n of the aquifer and of the heat capacities $\rho_w c_w$ and $\rho_a c_a$ of water and the aquifer, respectively. Compare [25, Ch. 10.3], [71, p. 21]. A numerical solute transport model can be adapted to heat transport by taking into account a retardation factor. The factor that translates solute transportation time into temperature transportation time is

$$R_t = \frac{\rho_a c_a}{n\, \rho_w c_w}$$

which approximately takes the value 2 in our computations.

We have now already begun to discuss the parameters that we used in our numerical experiments. The application and the results of the numerical experiments are described in detail in Chapter 4. Since all parameters have now been introduced we will at this point discuss the parameter choice. This determines the character of the equations of interest as the subsequent discussion points out. We list here the soil parameters and hydrothermal parameters used in the example problem. The problem of data procurement was already addressed. In general, data is expensive and difficult to acquire, for instance through field experiments and measurements. The considered regions are usually quite large, say, tens or hundreds of square kilometers, so that one will often have to work with average values. Several average values for different types of soil, e.g. for longitudinal and transversal dispersion lengths, are listed in [57, p. 69f.], [76]. See also [71, p. 22f.]. For the aquifer of our application, formed of gravel, we use the values of Table 2.2.1.

Table 2.2.1: Soil and hydrothermal parameters of the application.

k_0	$\approx$	$10^{-10}m^2$	soil permeability
μ	$\approx$	$1.3 \cdot 10^{-3} \frac{kg}{m \cdot s}$	dynamic viscosity of water (at $T = 10°C$)
n	$\approx$	30%	porosity of aquifer
ρ_w	$\approx$	$1000 \frac{kg}{m^3}$	density of water (at $T = 10°C$)
$\rho_w c_w$	$\approx$	$4.185 \frac{MJ}{m^3 \cdot °C}$	heat capacity of water
ρ_s	$\approx$	$2650 \frac{kg}{m^3}$	density of solid (quartz) (at $T = 10°C$)
c_s	$\approx$	$0.7 \frac{kJ}{kg \cdot °C}$	specific heat of solid (quartz)
$\rho_s c_s$	$\approx$	$1.8 \frac{MJ}{m^3 \cdot °C}$	heat capacity of solid
$\rho_a c_a$	$\approx$	$2.5 \frac{MJ}{m^3 \cdot °C}$	heat capacity of aquifer
λ_e	$\approx$	$2 \frac{J}{s \cdot m \cdot °C}$	effective thermal conductivity of solid
α_L	$\approx$	$10m$	longitudinal dispersion length
α_T	$\approx$	$1m$	transversal dispersion length

Classification of the Equations

The parameter choice allows to classify the partial differential equations that govern the optimization problem stated in Section 2.3. Equations (2.1.1) and (2.2.1) and their two–dimensional versions, Equations (2.1.4) and (2.2.5), respectively, are linear partial differential equations of second order. Second-order equations are usually classified as hyperbolic, parabolic, or elliptic. See e.g. [44, p. 40ff.]. This classification is both of theoretical interest and of practical significance. For instance, it is typically inefficient to use that numerical treatment on a parabolic partial differential equation that is efficient for a hyperbolic equation. Compare Section 4.2.

This classification is now applied to the equations at hand. We denote the spatial variables (x, y) by $X \in \Omega$, and, as before, time by $t \in [0, t_1]$. If the spatial derivative is written with the subscript X and the time derivative with the subscript t, Equation (2.1.8) takes the form

$$S\, h_t(X,t) = \nabla \cdot (K(X)\, \nabla h(X,t)) + q^h(X,t). \tag{2.2.11}$$

This simplifies to

$$\nabla \cdot (K(X) \nabla h(X)) = -q^h(X) \tag{2.2.12}$$

in the stationary case. Should the hydraulic conductivity be independent of the location, Equation (2.2.12) becomes

$$K\, \Delta h(X) = -q^h(X).$$

In that case, the equation, which is essentially the Laplacian, is obviously elliptic. Even if the tensor $K(X)$ depends on the spatial variables, it is in any case symmetric

and thus diagonalizable. It can be transformed to diagonal form, so that with positive hydraulic conductivities the condition for elliptic equations still holds in the considered region.

In the nonstationary case (2.2.11), assuming isotropic conditions, we find that the equation

$$S\,u_t = k_f(X) \cdot u_{XX} + q^h(X)\,,$$

is parabolic. The classification is unchanged under anisotropic conditions.

The equation modeling heat transport in an aquifer, Equation (2.2.1), is with the nomenclature introduced above given as

$$T_t(X,t) = -\nabla \cdot (\kappa T(X,t)\, v(X,t)) + \nabla \cdot (D(v)\, \nabla T(X,t)) + q^T(X,t) \quad (2.2.13)$$

with spatial variables X and time t. Classifying this equation according to what has been delineated for second–order equations one would call (2.2.13) parabolic. (There is no second time–derivative u_{tt} and no mixed derivative u_{Xt}.) Recall that the heat equation is a typical example for a parabolic equation. However, this classification does not adequately mirror the typical properties of (2.2.13). In the considered application, the convective part of the heat transport equation is usually dominant. Compare what is said about heat versus solute transport at the end of the preceding paragraph. The diffusion/dispersion part, which is the second–order term, is typically of minor importance. The coefficient $D(v)$ can under normal circumstances be assumed to be smaller than κ by at least two orders of magnitude. Thus, the linear convective part of Equation (2.2.13) is dominant, giving the transport equation hyperbolic character.

2.3 Statement of the Optimal Control Problem

In this section we state the optimal control problem in the formulation that we are concerned with. The application that we have in mind was delineated at the beginning of this chapter and is described in detail in Chapter 4. A review of other approaches to ground water management problems in the literature is appended to our problem formulation. This discussion of different approaches motivates again our chosen approach. We treat the application of Chapter 4 in a black–box formulation. We have discussed our situation in the Introduction already. In our case, only the equation solve is provided for by the given software, and the matrices of the discretization are not readily available.

We do not at this point intend to specify in detail the control that will be exercised. Recall that any control is tied to wells and their stress rates. The derivation of the basic law governing the motion of ground water in aquifers in Section 2.1 and the problem–oriented simplifications led to Equation (2.1.8). Since this equation is two–dimensional in space, the influence of wells on the flow regime enters (2.1.8) through the term $q^h(x,y,t)$ that accounts for sources and sinks of water in the considered domain. In a fully three–dimensional treatment of Equation (2.1.1), the influence of wells would be modeled through boundary conditions in three–dimensional space.

In the two–dimensional view, however, the wells can only be considered as an "inner boundary condition" and are modeled through $q^h(.)$.

In Section 2.2, considering the basics of transport in a porous medium and problem–oriented simplifications furnished the partial differential equation that models heat transport in an aquifer. Equation (2.2.8) is phrased in two–dimensional space. Sources and sinks of temperature within in the domain are accounted for by $q^T(x, y, t)$.

Thus, the control u that will be exercised enters the differential equations via the terms $q^h(.)$ and $q^T(.)$ which account for sinks and sources of water and temperature, respectively. These we denote by $q^h(x, y, t; u(x, y, t))$ and $q^T(x, y, t; u(x, y, t))$. The control u, which describes well action in our application, is composed of two parts. We write $u(x, y, t) = (u_h(x, y, t), u_T(x, y, t))$, or, in short–hand, $u(.) = (u_h(.), u_T(.))$. The first component which we call the flow component enters the partial differential equation that determines the flow regime. Since the flow must be known in order to solve the heat equation, it implicitly enters the heat equation. Explicitly considered in the transport equation, however, is only the temperature component of the control.

We are interested in controlling the temperature T in the considered region by using the barrier wells for pumping. We do not want to set a particular target temperature or particular pumping rates. Instead, we treat desired pumping rates as targets by absorbing them into an objective function of tracking–type rather than to treat them as constraints and fixed bounds. The control directly enters the objective function via some cost or penalty term. The temperature also enters the objective function. The flow regime is currently not explicitly considered in the objective function,

$$J(u(.), T(.)) = \langle u(.) - u^d(.), u(.) - u^d(.)\rangle + \langle w, T(.)\rangle.$$

Viewing the piezometric head h and the temperature T as independent variables, i.e., independent of u, the optimal control problem can be cast in a form that is usually referred to as all–at–once formulation. In such formulation, one seeks to minimize the objective function with respect to the variables T and u,

$$\min_{(T(.),u(.))} J(u(.), T(.))$$

under constraints. These constraints include for our application the sequential solution of two partial differential equations. Other constrained formulations are encountered in the following review of other efforts to deal with ground water management issues.

Alternative to viewing state and control variables as independent is considering the temperature T as dependent on the control u. If the temperature is considered as dependent on the control, it will enter the objective function as $T(.) = T(u)(.)$ in symbolic notation, or, with all arguments filled in, as $T(x, y, t; u(x, y, t))$. The temperature T is in this version implicitly defined via the control u. This leads to an unconstrained optimization problem. It is usually called black–box formulation. See Figure 2.3.1.

It is appropriate at this point to mention that, although the formulation is very general in this chapter, we are *not* dealing with decision variables u and T that are

truly functions. Instead, we are effectively only dealing with their discrete versions. The control u is the finite–dimensional input to a simulation code, and T is furnished as the solution to such a code.

We use a black–box formulation in the numerical treatment of our example problem. It was derived in the introductory Chapter 1 why this is the way to go in the setting of our first application. In this project, given software codes furnish the equation solve only. Our goal is to provide tools for optimization in this setting. The application is described in Chapter 4, the numerical results specifically in Section 4.3. The combined effects of simplifications in modeling, difficult data procurement, and the sequential use of off–the–shelf software codes lead to inaccuracy in the function evaluation. Also, nonsmoothness is observed. Special methods for optimization are thus required to which we turn in Chapter 3.

Figure 2.3.1: The optimal control problem in black–box formulation.

$$\min_{u(.)} J(u(.), T(.; u(.)))$$

where T(.;u(.)) is defined through

$$\begin{aligned}
S\,\tfrac{\partial}{\partial t}h(x,y,t) &= \nabla\cdot(K(x,y)\,\nabla h(x,y,t)) \\
&\quad +q^h(x,y,t;u_h(x,y,t)) && (x,y,t)\in\Omega\times[0,t_1]\\
h(x,y,t) &= h_D(x,y,t) && (x,y,t)\in\Gamma_D\times[0,t_1]\\
\tfrac{\partial}{\partial n}h(x,y,t) &= h_N(x,y,t) && (x,y,t)\in\Gamma_N\times[0,t_1]\\
h(x,y,t) & \\
+\tfrac{\partial}{\partial n}h(x,y,t) &= h_M(x,y,t) && (x,y,t)\in\Gamma_M\times[0,t_1]\\
h(x,y,0) &= h_0(x,y) && (x,y)\in\Omega\\
\\
v(x,y,t) &= -\tfrac{K(x,y)}{n}\,\nabla h(x,y,t) && (x,y,t)\in\Omega\times[0,t_1]\\
\\
\tfrac{\partial}{\partial t}T(x,y,t) &= -\nabla\cdot(\kappa T(x,y,t)\,v(x,y,t)) \\
&\quad +\nabla\cdot(D(v)\,\nabla T(x,y,t)) \\
&\quad +q^T(x,y,t;u_T(x,y,t)) && (x,y,t)\in\Omega\times[0,t_1]\\
T(x,y,t) &= T_D(x,y,t) && (x,y,t)\in\Gamma_D\times[0,t_1]\\
\tfrac{\partial}{\partial n}T(x,y,t) &= T_N(x,y,t) && (x,y,t)\in\Gamma_N\times[0,t_1]\\
T(x,y,t) & \\
+\tfrac{\partial}{\partial n}T(x,y,t) &= T_M(x,y,t) && (x,y,t)\in\Gamma_M\times[0,t_1]\\
T(x,y,0) &= T_0(x,y) && (x,y)\in\Omega
\end{aligned}$$

Before shifting our focus to specialized optimization routines, we review optimization approaches to ground water management that can be found in the literature. In recent years, as Wagner [81] states in a 1995 review article, "the aquifer simulation model has been combined with techniques of optimization to address important ground water management problems." The combined simulation and optimization model aims at accounting for the complex behavior of the ground water system and identifying the best management strategy under consideration of the management objectives and constraints. Simulation–optimization ground water management models have been developed for a variety of applications. The applications include the restoration of contaminated ground water and the control of aquifer hydraulics. Ground water management models are commonly divided into ground water hydraulic management and in ground water quality management. Ground water hydraulic management treats quantitative aspects and is concerned only with the underlying flow. Ground water quality management generally encompasses flow and transport modeling. The article [81] points to studies dealing with stochastic and combinatorial approaches to ground water management. Our review focuses on methods for continuous variables. Issues specific to the simulation are absent in our review.

In several studies of ground water hydraulic and of ground water quality management models dating back to the eighties, linear optimization methods are used. We briefly review the papers [16], [40], [68]. Pelka and Bogacki set up a linear model to solve ground water quantity management problems in [68]. Their two–dimensional ground water flow model is realized numerically with finite elements. The finite element mass matrix arising from the flow discretization is assembled explicitly as a full matrix. It is, together with the matrices accounting for box constraints on the piezometric head h, box constraints on the well stress rates u, and linear constraints on u, then put together in one large system. In case of a linear objective function the solution is handled in a straightforward way with the well–known simplex algorithm for linear programming. See e.g. [22]. Nonlinear objective functions are dealt with by repeatedly linearizing the function and repeatedly performing the linear optimization. Computations are presented for the stationary case. As the authors remark, this approach can be extended to the transient case, but this seems impractical. Matrices arising from the discretization of differential equations tend to large dimensions as the grid is refined, and the sheer size of the systems makes them difficult to handle.

Similarly, a variant of the simplex algorithm solves the linear programming problem of the ground water hydraulic management model set up by Chau in [16]. Also using finite elements in the flow simulation, a linear program is developed with equality constraints arising through the discretized flow equation and with inequality constraints on drawdown, water demand, and hydraulic gradient.

A different linear model, also solved with a variant of the simplex algorithm, is developed by Gorelick in [40]. The simulator is in this case two–dimensional as well. It is a combination of a finite–difference flow code and a method–of–characteristics solute transport model. The simulator is used to set up the so called "concentration response matrix" which is then the constraint matrix of a linear programming problem.

This matrix is obtained as the simulator output to a series of so called unit source injections. This is linearly independent and normalized decision variable input. It describes the influence of pollutant sources to contaminant concentrations at certain observation wells over time. The optimization objective is to maximize solute disposal rates while maintaining a given ground water quality at the observation wells. Numerical problems are reported with the method of characteristics of the solute transport model. The solutions to the solute transport model had to be smoothed out in order to be usable in the optimization problem. Multiple period source management is considered in [40], but computational problems are encountered due to the size of the considered systems.

According to Bear and Sun [10], a new direction for the planning in ground water remediation is introduced with [41] in 1984. Gorelick, Voss, Gill, Murray, Saunders, and Wright combine nonlinear optimization software with a numerical simulator. The simulator is SUTRA, a very general two–dimensional finite–element simulation model for ground water flow and transport which is available from the United States Geological Service. It is used in [41] for the simulation of ground water flow with nonreactive single species transport for a fluid of constant density. Like in our work, the flow and transport equations are thus discoupled and solved sequentially. Moreover, like in our work, the simulator is used in a black–box manner. It is called repeatedly in the course of optimization to furnish the function evaluations.

The nonlinear optimization software used in [41] is called a "development version" of MINOS by the authors. An overview of the methods employed by MINOS can be found in the book of Gill et al. [36]. The current features of the software package are highlighted at http://www.sbsi-sol-optimize.com. MINOS is a software package for solving large–scale optimization problems where objective and constraint functions may be linear or nonlinear. The nonlinear functions should be smooth but need not be convex. For nonlinear objective functions combined with linear constraints, MINOS uses a reduced–gradient method with quasi–Newton approximations to the reduced Hessian. For problems with nonlinear constraints, MINOS uses a projected Lagrangian method. It solves a sequence of subproblems in which the constraints are linearized and the objective is an augmented Lagrangian which involves all nonlinear functions. MINOS makes use of nonlinear function and gradient values. If some of the gradients are unknown, they are estimated by finite differences.

As mentioned above, the simulator is treated as a subroutine that is called by the optimization procedure for function and Jacobian evaluations. The authors stress three noteworthy features of this approach. First, it is quite general in that it can be easily extended to more complex systems (involving e.g. complicated chemistry and heat transport as well as viscosity and density changes which require to solve a coupled system of equations) as soon as a simulation model for such systems is accessible. This is of course true in our work also. We are not limited to a special simulator, and nonlinearity constitutes no impediment to the employed optimization routines. Second, the simulation model is a separately testable module. Numerical difficulties in the simulation can be worked out prior to usage within an optimization framework. While this is true in principle, numerical problems do still occur with heavily used simulation

software, and are likely for self–coded simulation codes. Also, the sequential use of different simulation codes can entail uncontrollable truncation errors. The third feature is that, because the simulation model is a separate module, a numerical approximation of the Jacobian can be easily obtained. While it is true that finite–differencing can be effectively used to obtain an approximation to first–order information, the validity of this information must be carefully investigated. We treat this aspect in the following Chapter 3. We do not want to assume that derivative information acquired through finite–differencing is fully accurate. This, however, is implicit in the investigations of Gorelick et al. in [41].

Considered in [41] are hypothetical systems with different objective functions, linear and quadratic. One example is to determine the minimum total pumping rate needed to reduce contaminant concentration below a given quality standard at specified locations in the domain. The objectives are considered in conjunction with nonnegativity constraints on the control u and inequality constraints on the simulation outcome. The simulation constitutes a nonlinear transformation of pumping rates u into concentrations at specified locations. No nonsmoothness is reported in the examples. A typical iteration history is not given. The number of admitted function evaluations has a high average of about 300.

A comparable approach is pursued by Xiang, Sykes, and Thomson in 1996. They perform a case study in [83] concerning optimal remedial pumping for a contaminated aquifer. The objective is to minimize total pumping while removing two ground water contaminant species to a sufficient extent. The problem is formulated with a simple linear objective function, the sum of the pumping rates, under box constraints on the pumping rates and with upper bounds on the contaminant concentration. Contaminant concentration is computed with a finite–difference model. The software package `NPSOL` developed by Gill, Murray, Saunders, and Wright is employed to solve the optimization problem. It is a software package for solving constrained, possibly nonlinear, optimization problems. Like with `MINOS`, the current features of the software package are highlighted at the site `http://www.sbsi-sol-optimize.com`, and an overview of the methods employed by `MINOS` can be found in the book of Gill et al. [36]. The objective and constraint functions should be smooth. The code uses an SQP algorithm for an augmented Lagrangian merit function. The user must provide subroutines that define the objective and nonlinear constraint functions. Their gradients, if not provided, are approximated by finite differences. Gradients are not provided in this case study.

Xiang et al. note that various forms of numerical problems can occur in the simulation. These are primarily attributed to the presence of advective terms in the transport equation. These numerical inaccuracies may have significant impact on the validity of a simulation and thus on the validity of optimization. The authors note that those numerical problems can significantly affect the smoothness properties required by the optimization tools that they employ, and that this may hinder the proper performance of the optimization. Moreover, convexity and smoothness of the quantities are generally difficult to determine. In this study nonconvexity and nonsmoothness

is observed in a number of instances. Nevertheless, Xiang et al. can report on fast improvement in the first steps of the employed optimization routine. This study also observes significant slowdown in iterative improvement within the optimization process as the concentrations are lowered. This is attributed to numerical inaccuracies due to smaller fractional concentration values. The authors note that further investigations concerning the convex and smooth properties and the effects of numerical problems in ground water simulation on optimization are worthwhile, as distinct point in favor of our pursued approach.

Bear and Sun stress in [10] that in most previous studies on ground water remediation, well stress rates and well locations are considered time–independent decision variables. Ground water flow is often approximated as steady, but solute transport processes in aquifers are not. Thus, Bear and Sun remark that constant pumping and injection strategies may fail to yield optimal remediation results in a transient ground water regime. In order to deal with changes in pumping rates over time, Culver and Shoemaker introduce a time–varying control law for water resource management problems in [24]. In this approach, a set of derivatives is computed to which we have no access. Also, because we only have very limited information about changes in boundary conditions over time, we restrict our analysis to steady–state pumping rates. In [24] like in many other studies, all wells are assumed to be installed at the beginning of clean–up time. Embarking from the observation that the process of determining well locations is ignored, Bear and Sun [10] formulate a multi–stage optimization problem. This is appropriate for their application which also happens to have a still considerably larger time horizon than is present in our application. Well locations, in any case, are not subject to optimization in our application. It is not in many other applications also. In many cases, the selection of pumping sites is not based primarily on the hydrogeology of the aquifer system. Practical considerations such as a desire to use existing wells (which is true for our application, see Chapter 4), land ownership, site access, legal issues, or preferred local hydrogeologic conditions restrict those areas where wells may be situated, compare e.g. [16], [41]. Moreover, the choice of pumping locations used in a specific analysis is only done within the limits of the spatial discretization dictated by the mesh and leads to a problem with discrete variables. We thus do not pursue this issue.

Chapter 3

Applicable Optimization Routines

A general algorithm for nonlinear optimization will generally be inefficient if applied to problems with special features. Certain problem characteristics have a great impact on ease of optimization, e.g. the form of constraints, if there are any, and problem size. For some problems exploiting the sparsity in the problem data leads to significant improvement in efficiency. This is important e.g. in our second application, a ground water hydraulic management problem that we treat in the later chapters, Chapters 5, 6, and 7. One of the fundamental properties of problem functions with respect to ease of optimization is differentiability. As a general rule, algorithms tend to become more successful and robust as more information is provided.

Most optimization software is designed to solve smooth problems. This is the case that we call typical and address in Section 3.2. Methods intended for smooth problems cannot be the method of choice when derivative information is unreliable. The latter is a fact for the practical application that we deal with here. This application is described in detail in Chapter 4. It is already referred to in Section 3.2 to establish the grounds for those methods that are the topic of this chapter, methods for noisy functions.

We perceive the noisy optimization problem as

$$\min_{x\in I\!R^n} \hat{f} : I\!R^n \to I\!R\,, \tag{3.0.1}$$

where the objective, $\hat{f}$, is composed of a smooth part, f, with a simple form, e.g. a convex quadratic, and a perturbation, φ,

$$\hat{f} = f + \varphi\,, \quad f \in C^1(I\!R^n)\,,\, \varphi \in L^\infty(I\!R^n)\,. \tag{3.0.2}$$

It is assumed that the noise φ is "much smaller" than the smooth part,

$$\max_{x\in I\!R^n} |\varphi(x)| \ll \max_{x\in I\!R^n} |f(x)|\,.$$

The study of optimization methods that do not require gradients is an active research area, e.g. [52]. The Implicit Filtering Algorithm described in Section 3.4 is less than ten years old. Some of the methods are classic, however, like for example the

Nelder–Mead algorithm. See Section 3.3. Many derivative–free methods examine a simplex of points in $\mathbb{R}^n$ at each iteration and then change the simplex in response. The fundamental idea is that through an organized sampling enough information is collected to approximate the gradient by differences, and that the accuracy in the difference approximation can be used to analyze the convergence. Care must be taken for problems of the form (3.0.1) not to make the difference increments so small as to attempt to differentiate the noise. The goal is to extract as much information as possible from the smooth part f without wasting effort in a futile attempt to minimize the noise φ. A general method for nonsmooth functions will in general be inefficient in terms of speed and reliability when the objective does have additional smoothness properties and should therefore be used only when there is no suitable alternative available. A substantial disadvantage of direct search methods is that few, if any, guarantees can be made concerning convergence. Often, these methods were developed to solve specialized problems and may be of very limited general usefulness. Their heuristic nature is reflected by a large number of parameters to be selected, and their performance often crucially depends on the choice of these parameters.

The optimization routines that we employ for the application introduced in Chapter 4 are described in Sections 3.2, 3.3, and 3.4, respectively. Theoretical and numerical properties of the different algorithms are considered. Before turning to these methods, we address auxiliary issues.

3.1 Line Search and Gradient Approximation

Line search procedures and gradient approximations are common tools in the optimization of smooth functions. We introduce the basic concepts in this section. Also covered are extensions of these concepts to the nonsmooth case.

3.1.1 Line Search Procedures

There are few optimization algorithms that do not utilize a line search procedure. Provided that the objective function is sufficiently smooth, one will usually for the solution of the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \qquad (f:\mathbb{R}^n\to\mathbb{R}) \tag{3.1.1}$$

by an iterative process in each step compute a descent direction p satisfying

$$\nabla f(x_c)^T p < 0. \tag{3.1.2}$$

Note that in case x_c is not an optimizer or a saddle point, the negative gradient, $-\nabla f(x_c)$, constitutes a descent direction. The condition (3.1.2) guarantees for a sufficiently small positive scalar δ descent in the objective function, i.e., there exists a $\delta > 0$ such that

$$f(x_c+\delta p) < f(x_c). \tag{3.1.3}$$

There are several ways to use the information contained in (3.1.2). One way is the method of line searches. The question answered by a line search procedure is: Given a descent direction p, how to choose λ to get an "acceptable" new iterate

$$x_+ = x_c + \lambda p\,?$$

For a discussion of what is "acceptable" in this context, a classic reference is e.g. [26, p. 116ff.]. There are two main points to be considered in order to set up an efficient line search algorithm. One is to prevent very small decreases in the function value relative to the step lengths. Thus it is required that the average rate of decrease from x_c to x_+ be at least some prescribed fraction of the initial rate of decrease in that direction, i.e., pick $\alpha \in (0,1)$ and choose λ_c from among those scalars satisfying $\lambda > 0$ and

$$f(x_c + \lambda p) \leq f(x_c) + \alpha \cdot \lambda \nabla f(x_c)^T p\,. \tag{3.1.4}$$

Condition (3.1.4) is referred to as "sufficient decrease condition". The second point is to prevent very small steps relative to the initial rate of decrease in the function value. For this it is required that the rate of decrease of the function value in direction p at x_+ be at least some prescribed fraction of the rate of decrease in p at x_c, i.e.,

$$\nabla f(x_+)^T p \geq \beta \cdot \nabla f(x_c)^T p \tag{3.1.5}$$

for some fixed constant $\beta \in (\alpha, 1)$.

Conditions (3.1.4) and (3.1.5) are based on work of Armijo, compare [26, p. 116ff.]. They lead to globally convergent algorithms. Computational experience has shown that it is not advisable to accurately solve the one–dimensional optimization problem

$$\min_{\delta} f(x_c + \delta p)$$

that can be derived from (3.1.3). The exact minimization is called Cauchy rule. Instead, it is common practice now to follow a backtracking strategy, e.g. the Armijo rule. See e.g. [26, p. 126ff.], [52, p. 40ff.]. If the starting parameter $\lambda = 1$ fails to satisfy the criterion in use, one repeatedly reduces the step size and tests for sufficient decrease. This is detailed in Figure 3.1.1.

The considerations related so far apply to the smooth case, in particular the sufficient decrease condition (3.1.4) for steepest descent and related algorithms. With this background, one can analogously define a sufficient decrease condition for direct search algorithms. Compare [52, p. 138]. Direct search algorithms are intended to solve the minimization problem (3.1.1) for nondifferentiable functions f by only comparing function values, i.e., without computing or explicitly approximating the derivative. In Section 3.1.3 we introduce the simplex gradient, a concept that generalizes finite–difference approximations. The simplex gradient $D(f : S)$ of f evaluated on a simplex S is a tool for the analysis of direct search methods. By lack of a search direction p in such methods, the analogue of the smooth sufficient decrease condition (3.1.4) reads

$$f(x_+) \leq f(x_c) - \alpha \left|\left| D(f : S) \right|\right|^2 . \tag{3.1.6}$$

Figure 3.1.1: Armijo line search strategy

Armijo line search strategy

Preprocessing Step:
Initialize starting point x, descent direction p.
Find function value f(x).
Evaluate ∇f(x)^{T}p.
Set $\alpha \in (0, 1/2)$.
Set $\rho \in (0, 1)$.

Algorithm:
Set $\sigma_0 = 1$.
Evaluate f(x + σ_0 p).
WHILE f(x + σ_0 p) - f(x) $\leq \alpha\sigma_0 \nabla$f(x)^{T}p DO
$\sigma_0 \leftarrow \sigma_0 + 1$
Evaluate f(x + σ_0 p).
END WHILE

Set $\lambda = \sigma_0 \cdot \rho$.
Evaluate f(x + λ p).
WHILE f(x + λ p) - f(x) $> \alpha\lambda \nabla$f(x)^{T}p DO
$\lambda \leftarrow \lambda \cdot \rho$
Evaluate f(x + λ p).
END WHILE

3.1.2 The Finite–Difference Gradient

We now turn to the finite–difference gradient and recall some of its properties as an approximation to the analytic gradient. We will see that the quality of the approximation deteriorates when dealing with noisy functions. This is considered here only for the one–dimensional case.

Recall that the first derivative $f'(x)$ of a smooth function $f : \mathbb{R} \to \mathbb{R}$ can be replaced by three different stencils, the forward or right difference $D_h^+ f(x)$, the backward or left difference $D_h^- f(x)$, and the central or symmetric difference $D_h^c f(x)$. These are defined as, see e.g. [45, p. 43],

$$\begin{aligned} D_h^+ f(x) &= \tfrac{1}{h}\,(f(x+h) - f(x)), \\ D_h^- f(x) &= \tfrac{1}{h}\,(f(x) - f(x-h)), \\ D_h^c f(x) &= \tfrac{1}{2h}\,(f(x+h) - f(x-h)). \end{aligned}$$

The quality of the difference schemes is not identical. Consider the function f on a closed set $Q \subset \mathbb{R}$.

Lemma 3.1.1 *Let $[x-h, x+h] \subset Q$. For the forward, backward, and central differences the following estimates hold,*

$$D_h^{\pm} f(x) = f'(x) + h\,R\,, \quad R \leq \tfrac{1}{2}\,\|f\|_{C^2(Q)}\,, \quad f \in C^2(Q)\ ,$$
$$D_h^{c} f(x) = f'(x) + h^2\,R\,, \quad R \leq \tfrac{1}{6}\,\|f\|_{C^3(Q)}\,, \quad f \in C^3(Q)\ .$$

This result asserts that, for suitable f, the forward and backward differences are first–order accurate, and the central differences are second–order accurate, i.e.,

$$D_h^{\pm} f(x) - f'(x) = \mathcal{O}(h)\,, \quad f \in C^2(Q)\,,$$
$$D_h^{c} f(x) - f'(x) = \mathcal{O}(h^2)\,, \quad f \in C^3(Q)\,.$$

In the presence of errors in the function evaluations, finite–difference approximations must be handled with special care. Compare e.g. [50, p. 75ff.], [52, p. 17f.]. Suppose that only functions can be evaluated and that gradients must be computed with differences. Suppose in addition that the functions are inaccurate, i.e., suppose that not the "true" function value $f(x)$ is furnished, but only $f(x)$ in conjunction with some error $\varphi(x)$, compare Equation (3.0.2),

$$\hat{f}(x) = f(x) + \varphi(x)\,. \tag{3.1.7}$$

Then the (forward) difference approximation to f' at x with increment h is given by the sum of the forward difference of f and of the forward difference of φ,

$$\begin{aligned} D_h^{+}\hat{f}(x) &= \tfrac{1}{h}\,(\hat{f}(x+h) - \hat{f}(x)) \\ &= \tfrac{1}{h}\,(f(x+h) + \varphi(x+h) - f(x) - \varphi(x)) \\ &= D_h^{+} f(x) + \tfrac{1}{h}\,(\varphi(x+h) - \varphi(x)) \\ &= D_h^{+} f(x) + D_h^{+}\varphi(x)\,. \end{aligned}$$

Assuming that the noise φ is essentially bounded on the considered interval by some constant ε, i.e., $\varphi \in L^\infty(Q)$, $\varphi(x) \leq \|\varphi\|_{L^\infty(Q)} \leq \varepsilon$, the quality of the approximation is

$$f'(x) - D_h^{+}\hat{f}(x) = \mathcal{O}(h + \frac{\varepsilon}{h})\,.$$

The quantity $h + \varepsilon/h$ is minimized when $h = \sqrt{\varepsilon}$. This means that very large *and* very small difference steps h can lead to strongly unreliable results. For instance, if $\varphi(x)$ is a result of floating–point round–off in full machine precision, i.e., $\varepsilon \approx 10^{-15}$, the analysis indicates that $h \approx 10^{-7}$ is a reasonable choice. One would not want to use larger steps in order to achieve maximum accuracy, and likewise taking a (too small) step like $h \approx 10^{-15}$ is not advisable.

For the application that we introduce in Chapter 4, $\varphi(x)$ is not floating–point round–off in the order of machine precision. It is considerably larger than 10^{-15}. Assuming an error margin of 1% for the output of the computational model, an upper range ε is estimated to be of size 10^{-3}. This is discussed in detail in Section 4.1.2.

The above discussion implicitly assumed $||x|| \approx 1$. If this is not the case, h should be scaled to reflect that. Thus,

$$h = ||x|| \sqrt{\varepsilon}$$

is appropriate. This, with typical values $||x|| \approx 10^{-2}$ in the application, leads to step sizes h not smaller than 10^{-4}. Smaller values will furnish unreliable results.

3.1.3 The Simplex Gradient

Several direct search methods, e.g. the Nelder–Mead algorithm, compare Section 3.3, examine a simplex of points at each iteration. A helpful tool for the analysis of such methods is the simplex gradient. It can also be used for the interpretation of Implicit Filtering, compare Section 3.4. The following material is taken from [13] and [52].

Definition 3.1.2 *A* simplex $S \in \mathbb{R}^n$ *is the convex hull of* $n+1$ *points* $x_j \in \mathbb{R}^n$ *(*$j = 1, \ldots, n+1$*). The point* x_j *is the* j*–th* vertex *of* S*. The simplex can be written as a* $n \times (n+1)$ *– matrix, where the* $(n+1)$ *column vectors are the vertices,*

$$S = (x_1, x_2, \ldots x_{n+1}) \; .$$

The $n \times n$*–matrix of* simplex directions *is*

$$V = V(S) = (x_2 - x_1, x_3 - x_1, \ldots, x_{n+1} - x_1) = (v_1, \ldots, v_n) \, .$$

The simplex S *is said to be* nonsingular *when the corresponding matrix of simplex directions* $V(S)$ *is nonsingular. The* simplex diameter *d(S) is*

$$d(S) = \max_{1 \le i,j \le n+1} ||x_i - x_j||_{\ell^2} \, .$$

Two oriented lengths *are given by*

$$\sigma_+(S) = \max_{2 \le j \le n+1} ||x_1 - x_j||_{\ell^2} \, , \quad \sigma_-(S) = \min_{2 \le j \le n+1} ||x_1 - x_j||_{\ell^2} \, .$$

We here employ the ℓ^2*–vector norm*

$$||v||_{\ell^2} = \frac{1}{n} \sum_{i=1}^{n} v_i^2 \, , \qquad v = (v_1, \ldots, v_n)^T \in \mathbb{R}^n \, ,$$

and use the induced matrix norm. The ℓ^2*–condition number*

$$\kappa(V) = ||V||_{\ell^2} \cdot ||V^{-1}||_{\ell^2}$$

is referred to as the simplex condition number. *The* vector of objective function differences $\delta(f : S)$ *is given by*

$$\delta(f : S) = (f(x_2) - f(x_1), f(x_3) - f(x_1), \ldots, f(x_{n+1}) - f(x_1))^T \, .$$

Note that the matrix of simplex directions and the vector of objective function differences depend on which of the vertices is labeled x_1. We assume throughout this section that the vertices are ordered according to their objective function values in increasing order, i.e.,

$$f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1}) . \tag{3.1.8}$$

Definition 3.1.3 *Let S be a nonsingular simplex with vertices x_j ($j = 1, \ldots, n$). The* simplex gradient $D(f : S)$ *is*

$$D(f : S) = V^{-T} \delta(f : S) .$$

This definition of the simplex gradient is motivated by the following first–order estimate, see [51], [52, Lemma 6.2.1].

Theorem 3.1.4 *Let S be a simplex. Let ∇f be Lipschitz continuous in a neighborhood of S with Lipschitz constant $2L$. Then there is a positive constant K which depends only on L s.t.*

$$\|\nabla f(x_1) - D(f : S)\|_{\ell^2} \leq K \, \kappa(V) \, \sigma_+(S) .$$

The analogue of Theorem 3.1.4 for objective functions of type (3.0.2) is given in the following theorem, see [51], [52, Lemma 6.2.2]. Obviously, depending on the size of the noise on the considered simplex, the quality of the approximation can be worse by orders of magnitude than in the case without noise. The influence of the noise is reduced when σ_+ is for the current simplex larger than the noise. However, this also necessarily adversely affects the approximation estimate. Thus, in approaching an optimizer it is necessary that the noise diminish faster than simplex size in order to get a good approximation.

Theorem 3.1.5 *Let S be a nonsingular simplex. Let $\hat{f}$ satisfy* (3.0.2)*. Let ∇f be Lipschitz continuous in a neighborhood of S with Lipschitz constant $2L$. Then there is a positive constant K which depends only on L s.t.*

$$\|\nabla f(x_1) - D(\hat{f} : S)\|_{\ell^2} \leq K \, \kappa(V) \left(\sigma_+(S) + \frac{\|\varphi\|_{L^\infty(S)}}{\sigma_+(S)} \right) .$$

Typically, in unconstrained optimization, the size of the gradient can be used to monitor progress of the iteration. One naturally asks what the consequences of a small simplex gradient are. The following theorem shows that a small simplex gradient is necessary for a good approximation, but by far not sufficient in the noisy setting.

Theorem 3.1.6 *Let $\hat{f}$ satisfy* (3.0.2)*. Let ∇f be continuously differentiable in a compact set $Q \subset \mathbb{R}^n$. Assume that the smooth part f has a unique critical point x^* in Q. Then there is a positive constant K_Q s.t. for any simplex $S \subset Q$ with vertices x_j ($j = 1, \ldots, n$) the following estimate holds,*

$$\|x_1 - x^*\| \leq K_Q \left(\|D(\hat{f} : S)\| + \kappa(V) \left(\sigma_+(S) + \frac{\|\varphi\|_{L^\infty(S)}}{\sigma_+(S)} \right) \right) .$$

All results related so far deal with the forward difference simplex gradient. We now define the central difference simplex gradient. The quality of the information obtained from the central simplex gradient is higher than that of the forward. Applying central differences, the numerical scheme is less sensitive to noise than in the one–sided derivative case. Analogous results are well known for finite–difference approximations to the gradient, see Section 3.1.2.

Definition 3.1.7 *Let S be a nonsingular simplex in $\mathbb{R}^n$ with vertices x_i ($i = 1, \ldots, n+1$) and simplex directions $v_j = x_{j+1} - x_1$ ($j = 1, \ldots, n$). The* reflected simplex *$R = R(S)$ is the simplex with vertices x_1 and $r_j = x_1 - v_j$ ($j = 1, \ldots, n$). The* central simplex gradient *$D_C(f : S)$ is defined as*

$$D_C(f : S) = \frac{1}{2} \left(D(f : S) + D(f : R) \right) .$$

Note that in the case $n = 1$ and $x_2 = x_1 + h$ we have $r_2 = x_1 - h$. Hence the forward difference simplex gradient for S and R reads

$$D(f : S) = \frac{1}{h}(f(x_1 + h) - f(x_1)) , \quad D(f : R) = \frac{1}{-h}(f(x_1 - h) - f(x_1)) ,$$

and

$$D_C(f : S) = D_C(f : R) = \frac{1}{2h}(f(x_1 + h) - f(x_1 - h))$$

is the usual central difference.

For the central difference simplex gradient, second–order analogues to the preceding Theorems 3.1.4, 3.1.5, and 3.1.6, can be proven. However, the requirements to reach these results are stronger. Not only the first, but also the second derivative of the smooth part of the objective is for these results required to be Lipschitz continuous. Compare Lemma 3.1.1. We state these results without further comment.

Theorem 3.1.8 *Let S be a nonsingular simplex. Let $\nabla^2 f$ be Lipschitz continuous in a neighborhood of $S \cup R(S)$ with Lipschitz constant $3L$. Then there is a positive constant K which depends only on L s.t.*

$$\|\nabla f(x_1) - D_C(f : S)\|_{\ell^2} \leq K \, \kappa(V) \, \sigma_+^2(S) .$$

Theorem 3.1.9 *Let S be a nonsingular simplex. Let $\hat{f}$ satisfy* (3.0.2)*. Let $\nabla^2 f$ be Lipschitz continuous in a neighborhood of $S \cup R(S)$ with Lipschitz constant $3L$. Then there is a positive constant K which depends only on L s.t.*

$$\|\nabla f(x_1) - D_C(\hat{f} : S)\|_{\ell^2} \leq K \, \kappa(V) \left(\sigma_+^2(S) + \frac{\|\varphi\|_{L^\infty(S)}}{\sigma_+(S)} \right) .$$

Theorem 3.1.10 *Let $\hat{f}$ satisfy* (3.0.2)*. Let $\nabla^2 f$ be continuously differentiable in a compact set $Q \subset \mathbb{R}^n$. Assume that the smooth part f has a unique critical point x^* in Q. Then there is a positive constant K_Q s.t. if a simplex S and its reflection R are both subsets of Q, then the following estimate holds,*

$$||x_1 - x^*||_{\ell^2} \leq K_Q \left(||D_C(\hat{f} : S)||_{\ell^2} + \kappa(V) \left(\sigma_+^2(S) + \frac{||\varphi||_{L^\infty(S)}}{\sigma_+(S)} \right) \right) .$$

The assumptions of Theorem 3.1.10, complemented with a sequence of uniformly well–conditioned simplices and decaying noise and oriented length,

$$\kappa(V(S^k)) \leq M \ \forall k \,, \ \lim_{k\to\infty} \sigma_+(S^k) = 0 \,, \ \lim_{k\to\infty} \frac{||\varphi||_{S^k}}{\sigma_+(S^k)} = 0$$

allow to conclude convergence to x^* from a sequence of small central simplex gradients. See [52, Th.6.2.8].

Usage of central differences is advantageous in another respect. One has information on the values of $\hat{f}$ at enough points to make an important qualitative judgement. In order to evaluate a central simplex gradient, $\hat{f}$ must be sampled at x_1 and at the points $x_1 \pm v_j$ ($j = 1, \ldots, n$). If

$$\hat{f}(x_1) \leq \hat{f}(x_1 \pm v_j) \quad \forall j = 1, \ldots, n \tag{3.1.9}$$

then the validity of using the simplex gradient as a descent direction or as a measure of stationarity is questioned. Condition (3.1.9) is called stencil failure. Stencil failure is used as a termination criterion in the implementation of the Implicit Filtering algorithm. This is based on a result from [13], see also [52, Th. 6.2.9]. It shows that in the case of stencil failure the gradient of the smooth part f of the objective is already small.

3.2 A Typical Optimization Code

The unconstrained minimization problem is to minimize a real–valued function f of n variables. This generally means the search for a local minimizer, i.e., for a point x^* s.t.

$$f(x^*) \leq f(x) \quad \forall x \in \{x : ||x - x^*|| \leq \epsilon\}$$

for an appropriate norm $||.||$ and an appropriate scalar $\epsilon > 0$. It is standard to express this problem as

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.2.1}$$

and to refer to f, $f : \mathbb{R}^n \to \mathbb{R}$, as the objective function.

The classical way to solve such problems (3.2.1) is to use methods like steepest descent and Newton's method or variations thereof in connection with a line search

strategy. This approach requires that f be sufficiently smooth and is based on the optimality conditions, see e.g. [36], [52].

The approach can fail if the objective function has discontinuities or irregularities. Such nonsmooth effects can be caused, for example, by truncation error or noise in the internal calculations for f. We lay down in Chapter 4 that noise is present in our application, its cause a combination of modeling, data, and implementational issues. This is a distinct hint that the smooth approach may not work for this problem. Moreover, only function values are at our disposal in the application, and no analytic first– or second–order information. Derivative information is customarily approximated via finite differences with small increments h, compare e.g. [52, pp. 17ff., 112ff.]. This is a feasible approach under the assumption that the function evaluation is accurate to machine precision. In the case where only function evaluations are available and where these must be assumed to be inaccurate, finite–difference approximations must be handled with special care. See Section 3.1.2. This is the reason why the smooth approach does not furnish usable results for our application.

We are in this section concerned with a subroutine that is designed to work the classical way, the subroutine `UNCMND`, part of the software package `NMS`. Although module `UNCMND` itself uses multiple subroutines that are quite general, few choices are given to the user of module `UNCMND`. This version of the code is designed to be easy to use. We first describe the subroutine "as is" and then point out where we went deeper into the code than intended for the user. Modifications to the code were made hoping to put it to use for the practical application that we consider.

Module `UNCMND` is designed to minimize a smooth nonlinear function f of n variables. The routine uses a quasi–Newton algorithm with line search to minimize the function. For an introduction to quasi–Newton methods see e.g. [26, Chs. 6, 8]. Line searches are treated in Section 3.1.1. The specifics of the methods are not of importance in this context.

Only function values have to be supplied for use in the routine. A subroutine to furnish the function value at any point must be given by the user. An analytic derivative is not required. It is assumed that the function values can be obtained accurately, i.e., to a precision determined by computer arithmetic. First–order derivative values are obtained by finite–differencing. The algorithm computes an approximation to the second derivative matrix of the nonlinear function using quasi–Newton techniques. A quadratic model of the nonlinear objective function is minimized to obtain a search direction, and an approximate minimum of the nonlinear function along the search direction is found using a line search.

When applied to the application described in Chapter 4, shortcomings of the routine are obvious. One minor problem is that there is no provision to restrict the number of function evaluations – although the user is asked to indicate whether the objective function is expensive to evaluate in which case it is customary to measure computational cost in terms of function evaluations. As usual in smooth optimization, a maximum number of iterations is set. This value is even preset. One iteration of the optimization method usually requires ten function evaluations. An additional, more

important problem with the code is that the user has no control over the increment h that is used to compute the finite–difference approximations. UNCMND uses an internally computed step size for the finite–differencing. This step size is small, usually of size 10^{-10}. Input changes that small are not read by the simulation code, thus leading to a finite–difference gradient identical to zero. No progress can be made at all.

It is not an option to change either the scaling of the input to the simulation code as to make changes of size 10^{-10} or smaller relevant or to just change the current input reading routine such that changes of size 10^{-10} and smaller are read. As is explained in Section 4.1.2, an error margin of about 0.1% in the input values must be assumed for our application. This error margin is relative to the input size and not an absolute value. Considering the usual size of the input data, input changes of size 10^{-10} are not relevant. As is laid down in Section 4.1.2, only changes of size 10^{-4} or larger are relevant. Thus, simply scaling the data, routine UNCMND would work with too small difference increments and be trapped by variations in the noise φ.

Due to this situation we leave this level of the code intended for the user. The lower–level subroutines, as mentioned before, are more general than the upper–level interface. One addition made by us was a function evaluation count in order to keep track of the expense of the computations. The most important change that we introduced into this setting was to leave the "fully accurate function" setting that is assumed in the condensed code. The function was newly declared to carry a relative noise of size 10^{-5}. This change leads to increments $h \in [10^{-3}, 10^{-4}]$ for the finite–difference computations. Thus, the entries of the finite–difference gradient stay quite large as must be expected. A "typical" convergence criterion, e.g. the requirement that the finite–difference gradient be of size of numerical roundoff, can hardly be satisfied. Convergence criteria for gradient–based iterative methods are usually tied to the size of the gradient. Terminating the iteration when the norm of the gradient at the current iterate is sufficiently small relative to the gradient at the starting point is reasonable under standard assumptions, compare [52, p. 21]. But for our application, such a criterion cannot be satisfied. In our numerical experiments, the relaxed criterion 10^{-3} for the size of the finite–difference gradient was never satisfied, even less a "typical" criterion of the order of machine epsilon. Relaxing this criterion even more to 10^{-1}, for instance, allows to stop due to this criterion, but no considerable progress of the algorithm is possible under these circumstances. Another stopping criterion is the step size near a solution. Although a small difference between consecutive iterates is in general not sufficient for convergence, compare [26, Ch. 6.3], this criterion is used in the code as well to indicate possible stagnation of the algorithm. Typically, the algorithm stopped when the relaxed criterion 10^{-3} for the difference between consecutive steps was met, or else when a prescribed number of iterations was exhausted. Progress was made thanks to fine–tuning below user–level. But the overall performance of the algorithm was clearly inferior to that of the routines described in the sections below, the Implicit Filtering and the Nelder–Mead algorithms.

Extensive testing was done with this routine in the early stages of optimization for the considered application. In subsequent work, we were able to plot optimization

landscapes for the application. To obtain such a landscape, all but two of the optimization variables are fixed. The function value is then plotted against the two varying values. See Figure 4.3.16 in Section 4.3. The landscape with its nondifferentiable rim shows that the routine presented in this section is very likely to fail on the application of interest. Also, the landscape shows distinct local minima with rather large basins of attraction. One problem, very distinctly observable for this routine and also present with the Nelder–Mead algorithm, is sensitivity to the starting point. The routine that is employed here, not designed as a global optimization routine, is likely to be trapped in local minima.

In summary, a code like `UNCMND`, designed to solve the unconstrained minimization problem (3.2.1) with a smooth objective f and not designed to deal with noisy functions, is despite the fine–tuning not suitable for our application. This motivates us to pay attention to methods that can deal with noisy functions described by Equation (3.0.2). The Nelder–Mead and Implicit Filtering algorithms are the topic of the following sections, Sections 3.3 and 3.4, respectively.

3.3 The Nelder–Mead Algorithm

The Nelder–Mead simplex algorithm, first published in 1965 [64], is a popular direct search method for multidimensional unconstrained optimization. The Nelder–Mead algorithm attempts to minimize a scalar–valued nonlinear function f of n real variables using only function values. This means that derivatives of the objective function are neither computed nor explicitly approximated. The algorithm thus falls in the general class of *direct search methods*. For a discussion of these methods and references see e.g. [52].

Upon start on a nondegenerate simplex, the Nelder–Mead algorithm maintains at each step a nondegenerate simplex, a fact which makes the method well defined. See [55]. In each iteration, one or more test points are computed along with the associated function values, and the iteration terminates with a new simplex such that the function values at its vertices satisfy some form of descent direction compared to the previous simplex. Typically, the Nelder–Mead algorithm requires only one or two function evaluations per iterations, compare [55], and is considered "cheap" in terms of function evaluations. This is a very attractive feature when function evaluations are expensive or time–consuming as is the case in many industrial applications. In addition, compare [55], the Nelder–Mead algorithm typically produces significant improvement in the first few iterations. Under performance considerations, this constitutes another important feature. In many applications, as is the case in our example problem, the search for an optimum is spurious. Due to time and computing restrictions, a percentual improvement of some performance measure is all one can reasonably expect in practice. Also contributing to the widespread usage of the Nelder–Mead algorithm is its simplicity.

We now informally describe the steps that are done in an iteration of the Nelder–Mead algorithm. A compressed version of the algorithm can be found in Figure 3.3.6.

The idea is to change the simplex form according to information gathered by comparing function values in an organized way.

Let a nondegenerate simplex S be given and its $n+1$ vertices $x_1, \ldots, x_{n+1} \in \mathbb{R}^n$. Compare Definition 3.1.2. The objective function f is evaluated at the vertices, and the vertices sorted according to their objective function values in increasing order, $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$. Compare (3.1.8). We call x_1 the best vertex and x_{n+1} the worst. The specifics of the sort and tie–breaking rules are purported not to have large impact. See [13], [55]. The algorithm attempts to replace the worst vertex x_{n+1} with a new point of the form

$$x(\delta) = (1+\delta)\,\bar{x} - \delta\, x_{n+1}\,, \tag{3.3.1}$$

where $\bar{x}$ is the centroid of the convex hull of the vertices $x_1, x_2, \ldots, x_n$, i.e.,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,. \tag{3.3.2}$$

The value of δ is selected from a sequence of parameters

$$-1 < \delta_{ic} < 0 < \delta_{oc} < \delta_r < \delta_e$$

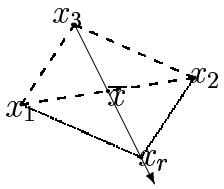
corresponding to rules that we describe now.

Reflection The first step is to reflect the worst vertex x_{n+1} at the centroid $\bar{x}$ given in Equation (3.3.2). See Figure 3.3.1 for the reflection. The reflected point x_r is computed as

$$x_r = \bar{x} + \varrho\,(\bar{x} - x_{n+1}) = (1+\varrho)\,\bar{x} - \varrho\, x_{n+1}\,,$$

where ϱ is called the *reflection coefficient*. We have $\delta_r = \varrho$. The objective f is evaluated at x_r and the result compared to the other function values.

Figure 3.3.1: Reflection of simplex with $\varrho = 1$.



In case

$$f(x_1) \leq f(x_r) \leq f(x_n)$$

the reflected point is accepted as the new vertex and the step is complete. Otherwise proceed with the next trial point by either performing an expansion or a contraction.

Expansion In case

$$f(x_r) < f(x_1)\,,$$

i.e., when the function value in the reflected point is smaller than in the best vertex, even more progress might be made by expanding the simplex along the search direction. See Figure 3.3.2. With χ the *expansion coefficient*, $\delta_e = \varrho\,\chi$. The *expanded vertex* is given as

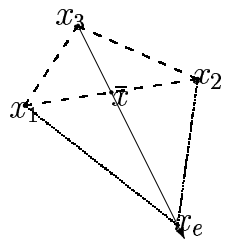
$$\begin{aligned} x_e &= \bar{x} + \chi\,(x_r - \bar{x}) \\ &= \bar{x} + \varrho\,\chi\,(\bar{x} - x_{n+1}) \\ &= (1 + \varrho\,\chi)\,\bar{x} - \varrho\,\chi x_{n+1}\,. \end{aligned}$$

If there is further improvement in the objective function value,

$$f(x_e) < f(x_r)\,,$$

x_e is accepted as the new vertex. Otherwise, x_{n+1} is replaced by x_r.

Figure 3.3.2: Expansion of simplex with $\chi = 2$.



Contraction In case

$$f(x_r) > f(x_n)\,,$$

a contraction is performed. This can be done as an outside or inside contraction.

Outside Contraction If

$$f(x_n) < f(x_r) < f(x_{n+1})\,,$$

an outside contraction is performed. See Figure 3.3.3. The *outside contraction point* x_{oc} is given by

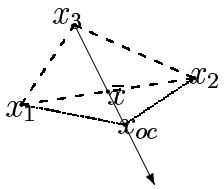
$$\begin{aligned} x_{oc} &= \bar{x} + \gamma\,(x_r - \bar{x}) \\ &= \bar{x} + \gamma\,\varrho\,(\bar{x} - x_{n+1}) \\ &= (1 + \varrho\,\gamma)\,\bar{x} - \varrho\,\gamma\,x_{n+1}\,, \end{aligned}$$

with γ the *contraction coefficient* and $\delta_{oc} = \varrho\,\gamma$. In case

$$f(x_{oc}) \leq f(x_r)\,,$$

the outside contraction is accepted and determines the new vertex. Otherwise, a shrink step is done. See below.

Figure 3.3.3: Contraction of simplex with $\gamma = 0.5$.



Inside Contraction In case the function value of the reflected point is larger than that of the worst vertex,

$$f(x_r) \geq f(x_{n+1})\,,$$

it is possible that the current simplex already contains the minimum. Thus, an inside contraction is performed. We use the contraction coefficient γ with negative sign, see Figure 3.3.4. The *inside contraction point*, x_{ic}, corresponding to $\delta_{ic} = -\gamma$, is given by

$$x_{ic} = \bar{x} - \gamma\,(\bar{x} - x_{n+1}) = (1-\gamma)\,\bar{x} + \gamma x_{n+1}\,.$$

If the inside contraction furnishes a point that is better than the worst vertex,

$$f(x_{ic}) < f(x_{n+1})\,,$$

then x_{n+1} is replaced by x_{ic}. Otherwise, a shrink step is done. See below.

According to these rules, in each iteration of the Nelder–Mead algorithm a new point is determined which obeys Equation (3.3.1) and which forms a new simplex together with the n better vertices. Problems can occur, however, if a contraction is done and neither outer nor inner contraction improve the function value. A shrink step is then performed.

Shrink In case

$$f(x_{ic}) > f(x_{n+1})\,, \qquad f(x_{oc}) > f(x_{n+1})\,,$$

a shrink step is done. Only the best vertex x_1 is kept and all adjoining edges are cut by a ratio determined through a parameter σ. See Figure 3.3.5.

Changes in the simplex are determined by the parameter values ϱ, χ, γ, and σ. According to Nelder and Mead [64] these must satisfy the conditions

$$\varrho > 0\,, \quad \chi > 1\,, \quad \chi > \varrho\,, \quad 0 < \gamma < 1\,, \quad 0 < \sigma < 1\,.$$

Figure 3.3.4: Contraction of simplex with $\gamma = -0.5$.

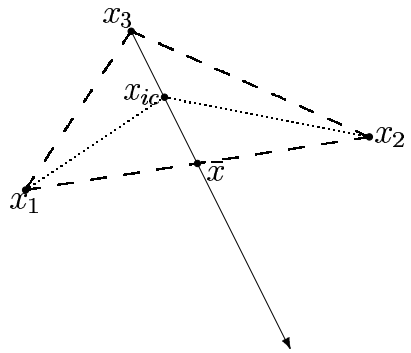
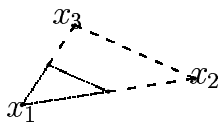


Figure 3.3.5: Shrinking of simplex.



In practice, the following combination is common choice,

$$\varrho = 1\,, \quad \chi = 2\,, \quad \gamma = \frac{1}{2}\,, \quad \sigma = \frac{1}{2}\,.$$

We now address the issue of convergence of the Nelder–Mead algorithm. Despite its widespread use, few theoretical results are known for the original Nelder–Mead algorithm as is stated in [55]. The original Nelder–Mead algorithm uses the difference

$$f(x_{n+1}) - f(x_1) \tag{3.3.3}$$

in function values as termination criterion. Near to the minimum, the best and the worst function values will be close to each other, this motivating to monitor (3.3.3). The algorithm stops when (3.3.3) becomes smaller than a predetermined tolerance $\epsilon > 0$.

For three decades since the publication of the original paper [64], the few known facts consisted mainly of negative results. In [60], McKinnon constructs a family of functions containing strictly convex functions with up to three continuous derivatives causing the Nelder–Mead algorithm to converge to a nonstationary point. In [55], Lagarias et al. present convergence results for the Nelder–Mead algorithm when applied

to strictly convex functions in one or two variables. In higher dimensions, the Nelder–Mead algorithm cannot be guaranteed to converge to a minimizer. Despite this lack of satisfying convergence results, the performance of the Nelder–Mead algorithm in practice is generally good, as confirm several authors, e.g. [52], [55].

Motivated by the counterexample by McKinnon [60], Kelley [51] proposes a test for sufficient decrease which, if passed for the entire iteration, will guarantee convergence of the Nelder–Mead algorithm to a stationary point for smooth objective functions. Failure of this condition indicates potential stagnation. It is this sufficient decrease condition and the notion of the simplex gradient that the following results are based on. We cast the main properties of the iteration in the form of an assumption.

Assumption 3.3.1 *Let for all steps $k = 1, 2, \ldots$ $(k \in I\!N)$ the following hold.*

- ◇ *The simplex S^k is nonsingular.*
- ◇ *The vertices are in each step ordered according to the objective function values in increasing order. Compare Equation* (3.1.8)*.*
- ◇ *The average function value $\underline{f}^k$ on the vertices of S^k is reduced in each iteration.*

Assumption 3.3.1 is satisfied by the Nelder–Mead iterates if the initial simplex directions are linearly independent and if no shrink steps are taken. When starting with a nonsingular simplex S^0, the nonsingularity of each of the simplices S^k, ($k = 1, 2, \ldots$, $k \in I\!N$) is a result from [55]. The reduction of the average function value is provided in the Nelder–Mead algorithm as long as no shrink step is done. By reflection, expansion, and by contraction, the worst vertex is replaced by a new vertex with a better function value. Thus, the average function value

$$\underline{f} = \frac{1}{n+1} \sum_{i=1}^{n+1} f(x_i) \tag{3.3.4}$$

is reduced. This is considered as progress in the iteration. Note that a Nelder–Mead iteration not necessarily results in a reduction in the *best* function value.

Based on the average function value $\underline{f}$ and the idea of line search procedures, compare specifically Equation (3.1.6) in Section 3.1.1, a sufficient decrease condition can be formulated for the Nelder–Mead algorithm,

$$\underline{f}^{k+1} - \underline{f}^k < -\alpha \left\| D(f : S^k) \right\|^2 . \tag{3.3.5}$$

Here, $\alpha > 0$ is a small parameter, typically $\alpha = 10^{-4}$. Compare [51], [52]. Condition (3.3.5) uses the simplex gradient $D(f : S)$, this being justified by the approximation of the gradient ∇f by the simplex gradient, see Theorem 3.1.4.

Under the assumptions of Theorem 3.1.4 for the sequence of Nelder–Mead simplices it can be shown for sufficiently smooth objective functions f that accumulation points of the Nelder–Mead iterates are critical points for f.

Theorem 3.3.2 *Let a sequence $\{S^k\}_{k=0}^{\infty}$ of simplices satisfy Assumption* 3.3.1. *Let for this sequence the gradient ∇f be Lipschitz continuous in a neighborhood of each of the S^k with Lipschitz constants $2L^k$. Let the sequence of Lipschitz constants $\{L^k\}_{k=0}^{\infty}$ be uniformly bounded. Assume that the sequence $\{\underline{f}^k\}_{k=0}^{\infty}$ of average function values on S^k is bounded below. Then if the sufficient decrease condition* (3.3.5) *holds for all but finitely many k and if*

$$\lim_{k\to\infty} \kappa(V(S^k))\,\sigma_+(S^k) = 0\,, \tag{3.3.6}$$

then any accumulation point of the simplices is a critical point of f.

An analogous result can be deduced for perturbed functions. Compare Theorem 3.1.5. The result for noisy functions $\hat{f}$ that satisfy (3.0.2) with a smooth part f reflects that the resolution is limited by the size of the noise φ. As soon as $\sigma_+(S^k)$ is smaller than $\|\varphi\|_{L^\infty(S^k)}$, no more information on the smooth part f can be obtained by evaluating $\hat{f}$ at the vertices of S^k.

Theorem 3.3.3 *Let a sequence of simplices satisfy Assumption* 3.3.1. *Let ∇f be Lipschitz continuous in a neighborhood of S^k with Lipschitz constants $2L^k$. Let the Lipschitz constants $\{L^k\}_{k=0}^{\infty}$ be uniformly bounded. Assume that the sequence $\{\underline{f}^k\}_{k=0}^{\infty}$ of average function values on S^k is bounded below. Then if the sufficient decrease condition* (3.3.5) *holds for all but finitely many k and if*

$$\lim_{k\to\infty} \kappa(V(S^k)) \left(\sigma_+(S^k) + \frac{\|\varphi\|_{L^\infty(S^k)}}{\sigma_+(S^k)}\right) = 0\,, \tag{3.3.7}$$

then any accumulation point of the simplices is a critical point of f.

Both Theorems 3.3.2 and 3.3.3 impose, in addition to the conditions discussed above, a condition on the simplex condition number. Conditions (3.3.6) and (3.3.7) prevent that the simplex condition can grow unbounded. This is motivated by Theorem 3.1.6 and by the counterexample of McKinnon [60]. However, a large simplex condition number is not *per se* an occurrence which has to be prevented. The ability of the Nelder–Mead simplices to vary in shape presents a substantial advantage over other direct search methods, compare [52]. In the noisy case, compare Theorem 3.3.3, the situation is even more complex. Not only the simplex condition number, but also the size of the noise φ plays a relevant role. As one expects, a large error makes fulfillment of condition (3.3.7) difficult.

Figure 3.3.6: The Nelder–Mead algorithm

The Nelder–Mead algorithm

Preprocessing Step:
Set number `maxeval` of admissible function evaluations.
Set termination tolerance `tol` for convergence criterion $\mathtt{f}(\mathtt{x}_{n+1}) - \mathtt{f}(\mathtt{x}_1) < \mathtt{tol}$.

Algorithm:
Initialize starting simplex S and its vertices $\mathtt{x}_1, \ldots, \mathtt{x}_{n+1}$.
Find corresponding function values $\mathtt{f}(\mathtt{x}_i)$.
Sort vertices according to objective function values in increasing order.
WHILE $\mathtt{f}(\mathtt{x}_{n+1}) - \mathtt{f}(\mathtt{x}_1) > \mathtt{tol}$ DO
Compute $\bar{\mathtt{x}}$ and $\mathtt{x}_r$.
`reflect`: If `maxeval` is exceeded, exit.
Evaluate $\mathtt{f}(\mathtt{x}_r)$.
If $\mathtt{f}(\mathtt{x}_1) \le \mathtt{f}(\mathtt{x}_r) < \mathtt{f}(\mathtt{x}_n)$, replace $\mathtt{x}_{n+1}$ by $\mathtt{x}_r$. Goto `sort`.
`expand`: If `maxeval` is exceeded, exit.
If $\mathtt{f}(\mathtt{x}_r) < \mathtt{f}(\mathtt{x}_1)$, compute $\mathtt{x}_e$ and evaluate $\mathtt{f}(\mathtt{x}_e)$.
If $\mathtt{f}(\mathtt{x}_e) < \mathtt{f}(\mathtt{x}_r)$, replace $\mathtt{x}_{n+1}$ with $\mathtt{x}_e$. Goto `sort`.
If $\mathtt{f}(\mathtt{x}_e) \ge \mathtt{f}(\mathtt{x}_r)$, replace $\mathtt{x}_{n+1}$ with $\mathtt{x}_r$. Goto `sort`.
`contract outside`: If `maxeval` is exceeded, exit.
If $\mathtt{f}(\mathtt{x}_n) \le \mathtt{f}(\mathtt{x}_r) < \mathtt{f}(\mathtt{x}_{n+1})$, compute $\mathtt{x}_{oc}$ and evaluate $\mathtt{f}(\mathtt{x}_{oc})$.
If $\mathtt{f}(\mathtt{x}_{oc}) \le \mathtt{f}(\mathtt{x}_r)$, replace $\mathtt{x}_{n+1}$ with $\mathtt{x}_{oc}$. Goto `sort`.
If $\mathtt{f}(\mathtt{x}_c) > \mathtt{f}(\mathtt{x}_r)$, goto `shrink`.
`contract inside`: If `maxeval` is exceeded, exit.
If $\mathtt{f}(\mathtt{x}_r) \ge \mathtt{f}(\mathtt{x}_{n+1})$, compute $\mathtt{x}_{ic}$ and evaluate $\mathtt{f}(\mathtt{x}_{ic})$.
If $\mathtt{f}(\mathtt{x}_{ic}) < \mathtt{f}(\mathtt{x}_{n+1})$, replace $\mathtt{x}_{n+1}$ with $\mathtt{x}_{ic}$. Goto `sort`.
If $\mathtt{f}(\mathtt{x}_{ic}) \ge \mathtt{f}(\mathtt{x}_{n+1})$, goto `shrink`.
`shrink`: If `maxeval` is exceeded, exit.
For $2 \le i \le n+1$: Set $\mathtt{x}_i = \mathtt{x}_1 - (\mathtt{x}_i - \mathtt{x}_1)/2$. Compute $\mathtt{f}(\mathtt{x}_i)$.
`sort`: Sort vertices according to obj. function values in increasing order.
END WHILE

3.4 Implicit Filtering

We now examine an algorithm for the optimization of noisy functions that is considerably younger than the Nelder–Mead algorithm. The Implicit Filtering algorithm, originally formulated in 1991, is described in [38]. Implicit Filtering in its simplest

unconstrained form is the steepest descent algorithm with difference gradients. The difference increment is reduced in size as the iteration progresses so that in fact the algorithm is a repeated call of steepest descent. The algorithm can be viewed as an extension of the projected gradient algorithm, see e.g. [52, p. 91f.], for it can easily be adapted to incorporate bound constraints. For problems that are sufficiently smooth near a minimizer, it is also possible to obtain faster convergence in the terminal phase of the iteration by incorporating quasi–Newton methods, see [19].

Implicit Filtering has been developed for a particular class of optimization problems. The objective functions under focus are composed of a smooth part and a perturbation which we refer to as noise. Compare Equation (3.0.2). This is appropriate in the setting of our example problem, where noise is introduced into the model function through two circumstances. First, any data for the considered problem is relevant only within an error margin of about 0.1%, causing noise in the output that may be as large as 1%. See Section 4.1.2. Second, the cost function evaluation involves the sequential black–box type use of two software codes, `MODFLOW` and `MT3D`, for the solution of the discrete flow and transport equations. This introduces severe round–off error that makes acquiring derivative information a delicate task.

Our first issue in this section is the description of the basic algorithm and its analysis with the tool developed in Section 3.1.3. We subsequently address the constrained version of the algorithm and discuss implementational issues.

In its simplest unconstrained form, the implicit filtering algorithm is the steepest descent algorithm with difference gradients. The difference gradient being only an approximation, the computed steepest descent direction may fail to be a descent direction, and the line search may fail. The remedy which coins this algorithm is to reduce in size the difference increment as the iteration progresses. So the basic algorithm is to perform steepest descent repeatedly at different scales h. This is done using a finite–difference gradient D_h in combination with an Armijo line search. Compare Section 3.1.1 for the line search. The procedure requires a decreasing and finite sequence of difference steps $\{h_i\}_{i=1}^m$ that are called *scales*. On each scale, the iteration terminates after stencil failure, compare Section 3.1.3, after line search failure due to exceeding a maximum number of backtracking steps, compare Section 3.1.1, after exceeding a maximum number of steps, or upon satisfying a predetermined termination tolerance

$$||D_h f(x)|| \leq \tau\, h\,.$$

We now turn back to the simplex gradient for the analysis of the algorithm. Let $x \in I\!R^n$ be given and a difference increment $h > 0$. Let $S(x,h)$ be the right simplex from x with edges having length h. This means that $S(x,h)$ has the vertices x and $x + h\, e_i$ $(i = 1, \ldots, n)$, where $e_i \in I\!R^n$ denotes the i–th unit vector. So $V = I_{n\times n}$ and $\kappa(V) = 1$. In this case, the central difference gradient $D_h^c \hat{f}$, compare Definition 3.1.7, and the central simplex gradient $D_c(\hat{f} : S(x,h))$ coincide,

$$D_h^c \hat{f}(x) = D_c(\hat{f} : S(x,h))\,.$$

In this section we denote by S^k the simplex on the current scale h_k, i.e., $S^k = S(x, h_k)$.

Since $h_k = \sigma_+(S^k)$ and $\kappa(V^k) = 1$, Theorem 3.1.5 implies not only the convergence result for the Nelder–Mead algorithm in Theorem 3.3.3, but also the convergence result in Theorem 3.4.1. Among the assumptions for this result is, like in the previous section, the requirement that the noise decay fast near the solution.

Theorem 3.4.1 *Let $h_k \to 0$ as $k \to \infty$. Let x_k be the iterates of the Implicit Filtering algorithm. Assume that there is no line search failure for all but finitely many k. If the requirement*

$$\lim_{k\to\infty}(h_k^2 + \frac{\|\varphi\|_{L^\infty(S^k)}}{h_k}) = 0 \tag{3.4.1}$$

holds, then any limit point of the sequence of iterates $\{x_k\}_{k=1}^\infty$ is a critical point of the smooth part f.

Proof: If either the sufficient decrease condition

$$\hat{f}(x_c + \lambda\, D_h\hat{f}(x_c)) \le \hat{f}(x_c) + \alpha\cdot\lambda\,\|D_h\hat{f}(x_c)\|^2$$

holds, compare Equation (3.1.4) in Section 3.1.1, or if stencil failure occurs,

$$\hat{f}(x_1) < \min_{j=1,\ldots,n}\hat{f}(x_1 \pm h\, e_j)\,,$$

compare Equation (3.1.9) in Section 3.1.3, for all but finitely many k, then

$$D_{h_k}\hat{f}(x_k) = D_C(f : S^k) \to 0\,.$$

Hence, using the condition (3.4.1) above and, under appropriate differentiability assumptions, Theorem 3.1.9,

$$Df(x_k) \to 0\,.$$

This shows that limit points of the sequence of iterates $\{x_k\}_{k=1}^\infty$ are critical for the smooth part f. □

Apart from the algorithm's conception as optimizer for noisy functions it has an additional attractive feature in that it can be easily adapted to accommodate simple box constraints. The consideration of such simple constraints is often particularly appropriate when dealing with industrial problems where the optimum may not be known but where often a good knowledge exists about "reasonable" size of input or where certain restrictions apply to admissible input. In our example problem, this is not necessary. Although a "reasonable" input range should in principle be obeyed, this is not violated in the computations anyway, compare Section 4.3. Nevertheless, we use the algorithm in its constrained implementation. We abbreviate *Implicit Filtering for Constrained Optimization* as IFFCO. IFFCO is designed to find the optimizer of functions of the special form (3.0.2) subject to simple box constraints. Recall the form

of the objective that has been introduced at the beginning of this chapter, (3.0.2). This we now consider on a simple set $Q \subset \mathbb{R}^n$,

$$\hat{f} = f + \varphi, \quad f \in C^1(Q)\,,\, \varphi \in L^\infty(Q)\,.$$

The box constraints are determined by the hyperbox Q,

$$Q = \{x \in \mathbb{R}^n : l^i \leq x^i \leq u^i \quad \forall i = 1, \ldots, n\}\,, \tag{3.4.2}$$

which is defined via the upper and lower bounds l^i and u^i on the i-th variable. We will throughout this section assume that Q is bounded and denote its diameter by d_Q, $d_Q = \max\{u^i - l^i\}$. The problem description is thus with the objective $\hat{f}$, $\hat{f} : \mathbb{R}^n \to \mathbb{R}$,

$$\min_{x \in Q} \hat{f}\,. \tag{3.4.3}$$

It is still assumed that the noise be small in magnitude relative to the smooth part,

$$\max_{x \in Q} |\varphi(x)| \ll \max_{x \in Q} |f(x)|\,.$$

IFFCO in its basic form is the call of the projected–gradient algorithm, repeated at different scales h. The complete algorithm is given in Figure 3.4.1. Additional steps are necessary if a quasi–Newton routine is integrated. The so called scales are a decreasing and finite sequence of difference steps $\{h_i\}_{i=1}^m$. The call of the projected–gradient routine is iterated with h_i for $i = 1, \ldots, m$. On each scale, the projected–gradient algorithm is terminated when

$$\|x - \mathcal{P}(x - D_{h_i} f(x))\| \leq \tau h_i\,.$$

The parameter τ controls the termination tolerance. In this termination criterion, $\mathcal{P}$ denotes the projection onto the feasible set Q,

$$\mathcal{P}(x)^i = \begin{cases} u^i & x^i > u^i, \\ x^i & l^i \leq x^i \leq u^i, \\ l^i & l^i < x^i. \end{cases} \tag{3.4.4}$$

The projection can lead to degeneracy of the current simplex. In that instance, the iterates of IFFCO cannot be shown to obey the results of Theorem 3.4.1.

The noise φ in (3.0.2) can cause the observable sum $\hat{f}$ to have local minima. This traps conventional (local) optimization algorithms. One way to avoid such local minima is to apply a filter to $\hat{f}$, compare [38]. By refining the filter as an iteration progresses, one can hope to find a minimizer of f up to the accuracy allowed by the noise in the observation. This is being mimicked by Implicit Filtering. The finite–difference gradient based method is applied for a scale h, the scale is decreased after convergence, and the gradient based method is applied again. The differencing is used to step over the noise φ at varying levels of resolution. Obviously, a local minimum of

the unperturbed function can at best be identified up to the accuracy permitted by the perturbation. The algorithm cannot be guaranteed to find even a local minimum.

The question of a proper termination criterion for IFFCO has not yet been satisfyingly answered. A bypass is to allow for restarting the algorithm with the previously computed solution. This leads to the concept of a *minimum at all scales*. The minimum at all scales is a point which is left unchanged by IFFCO, i.e., a point which cannot be improved upon by the algorithm at all considered scales $h_1, \ldots, h_m$.

Definition 3.4.2 *The point x is called a* minimum at all scales *for $\hat{f}$ if the projected–gradient routine leaves x invariant for all $h = h_1, \ldots, h_m$.*

Such a minimum at all scales is computed by applying IFFCO and restarting, if necessary, until each call to the projected–gradient routine leaves the current point unchanged. Definition 3.4.2 ignores how the result of the projected–gradient algorithm is tied to the current termination criterion which mainly depends on τ. Heuristically, an initial τ that is small could lead to entrapment in a local minimum. However, if τ is too large, no progress can be made. The projected–gradient algorithm terminates on entry if the convergence criterion is too weak.

The goal to find a minimum at all scales rather than a global minimum or a local minimum reflects those problems that have motivated the development of Implicit Filtering. These are problems where the function evaluations must be assumed to be inaccurate. Spikes and deep valleys in the function surface are likely to be due to noise in the function, and it is not desirable to stop at one of these locations. A relation between a minimum at all scales and the global minimum of the smooth part f can be established. Under rather strong assumptions on the decay of the error near the global minimum, it is shown in [38] that a minimum at all scales is a first–order estimate of the global minimum.

In the last part of this section we address implementational issues. The algorithm has been implemented as a `FORTRAN` code under the name `IFFCO` by a work group at the Center for Research in Scientific Computation (CRSC) at North Carolina State University. The current code allows to use the finite–difference based gradient projection method on its own or in conjunction with two quasi-Newton methods to have a descent direction computed. The updates SR1 and BFGS are provided for. We used SR1 in our computations. An Armijo line search method is integrated in `IFFCO`. The code, documented in [17] and [37], is available at the web address `http://www4.ncsu.edu/eos/users/c/ctkelley/www/tim.html`. An additional feature of the Implicit Filtering algorithm is its potential for parallelization. This can be done by simply performing those function evaluations in parallel that are needed for the finite–difference gradient.

The algorithm has several iterative parameters and requires algorithmic decisions in its implementation. One issue is the termination tolerance τ, denoted as `tol` in the compressed version of the algorithm in Figure 3.4.1. The performance of Implicit Filtering can be very sensitive to this termination tolerance. Small values of `tol` will

lead to stagnation of the algorithm, and large values of `tol` will entail premature termination. The choice `tol`$= 1$ served best in our application. Another very important choice is the range for the scales. The sequence of scales is at best a guess at the level of noise in the problem. This is inherent to the considered problem class, because the true error margin cannot be known. If several of the scales are smaller than the level of noise, the line search will fail. Work at these scales is wasted. For our application, a finite–difference increment smaller than 10^{-4} cannot be expected to furnish usable results. Since the starting scale 0.5 is the default, the range for the scales is $[0.5, 10^{-4}]$ in our computations.

Figure 3.4.1: `IFFCO`

`IFFCO` : Version without restarts

Preprocessing Step:
Set range [`minh`,`maxh`] for finite difference `h`.
Set number `maxit` of admissible iterations.
Set number `maxeval` of admissible function evaluations.
Set number `maxcut` of admissible backtracking steps in line search.
Set termination tolerance `tol`
for convergence criterion $\|\mathtt{x} - \mathcal{P}(\mathtt{x} - \mathtt{D_h f(x)})\| \leq \mathtt{tol} \cdot \mathtt{h}$.

Algorithm:
Initialize starting point `x`.
Find corresponding function value `f(x)`.
Initialize `h` to `maxh`.
WHILE `h` $\geq$ `minh` DO
Calculate finite difference gradient $\mathtt{D_h f(x)}$.
WHILE
- no stencil failure occurs
- no line search failure occurs
- `maxit` is not exceeded
- `maxeval` is not exceeded
- `maxcut` is not exceeded
- conv. criterion is not satisfied DO

Perform line search.
Update `x`.
Evaluate conv. criterion.

END WHILE
Reduce `h`.
END WHILE

Chapter 4

A Practical Application

We have developed an optimal control problem governed by partial differential equations in Chapter 2. See Section 2.3 specifically. The underlying partial differential equations which model flow and transport in porous media are introduced in Sections 2.1 and 2.2, respectively. The optimal control problem has been developed with the practical application in mind to which we turn now.

In Section 4.1 we describe the problem at hand with its hydrogeologic features. Subsequently, the numerical setting of the discrete optimization problem is explored. Each function evaluation in the course of optimization constitutes a solve of the underlying partial differential equations. We turn to the simulation codes in Section 4.2. Their black–box type use determines the problem formulation and accounts for numerical problems. Optimization results which have been obtained with the Implicit Filtering and Nelder–Mead algorithms are given in Section 4.3.

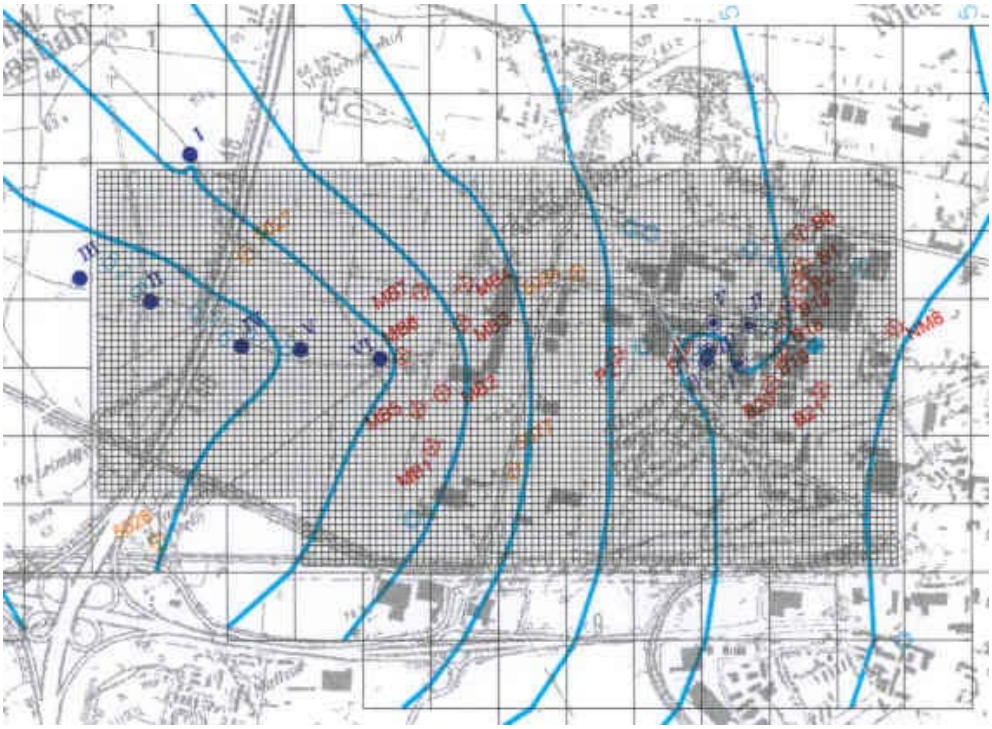
4.1 The Setting

This section is devoted to the first application to which our industrial partner turned our attention. An abstract formulation of the application is set up in Chapter 2, and the basis for its numerical handling within an optimization framework is laid in Chapter 3. Here, we depict the specific application for the first time. Its hydrogeologic features are supplemented by their respective translation within a numerical scheme. The grid is defined, and results of the simulation are presented. A simulation constitutes one function evaluation within an optimization framework. The optimization goal is defined in Section 4.1.2. Along with this goes the description of the numerical setting.

4.1.1 The Hydrogeologic Setting

Our overall concern is the drinking water supply of the region shown in Figure 4.1.1. We now describe in short terms the main hydrogeologic features of this region. These have been provided by Technologieberatung Grundwasser und Umwelt GmbH, a company of consulting engineers for ground water and water resources.

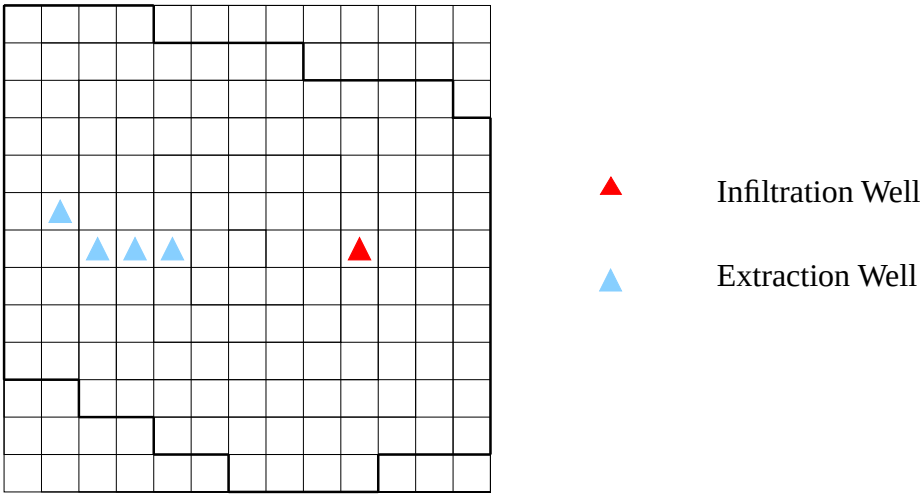
Figure 4.1.1: The considered region.



The water bearing zone of the considered region is dominated by a stream in its northern part. The considered region is the recharge zone for a cluster of drinking water wells which are located in its western part. The drinking water wells are extracting about 5.4 million m^3 per year. Within the considered region also lies an industrial zone with several additional wells, mostly extracting water for industrial purposes. One company, however, does not only extract water for use within their cooling system, but also reinjects the same amount of heated water into the ground. This is a considerable 2.4 million m^3 per year. The temperature of the reinjected water is approximately $5°C$ higher than the temperature of ground water in its vicinity. Reinjection of the heated water leads to a temperature anomaly in the ground water. An overall temperature increase of about $1°C$ occurs at the drinking water well nearest to the infiltration well. This increase is attributed to the effects of the infiltration. Concerns are raised due to the judicial requirement that anthropogenic changes of ground water properties be minimized. The location of the drinking water wells and the infiltration well within the grid is shown in Figure 4.1.2. That figure depicts a coarse grid set up to reflect the shape of the considered region.

The aquifer of principal interest is formed of gravel. The water level is under normal hydrological conditions approximately $10m$ below ground surface. The aquifer has an average thickness of $10 - 12m$. Its base is a relatively impermeable clay layer. The gravel forms an aquifer of good to very good hydraulic conductivity. The hy-

Figure 4.1.2: Well location within the grid.



draulic conductivity is of size $5 \cdot 10^{-3}\, m/s$. Local variations in conductivity occur. But it it not possible to perform accurate measurements of these variations. Similar statements hold for other soil parameters. For instance, in laboratory tests, only a range was determined for porosity, 25% to 35%. We use the values of Table 4.1.1 in our computations. Data procurement presents a serious problem. In general, reliable and accurate data is expensive and difficult to get, for instance through field or laboratory experiments and measurements. The regions that are investigated for hydrological purposes are usually quite large, often encompassing tens or hundreds of square kilometers, so that one often has to work with average values. We have to do so as well. For different types of soil, several average parameter values, e.g. for longitudinal and transversal dispersion lengths, are listed in [57, p. 69f.], [76]. See also [71, p. 22f.]. In Table 4.1.1, we list the most important soil and hydrothermal parameters that are used in our application. For the significance of the different parameters compare Sections 2.1, 2.2, and 4.2. In Section 2.2, several parameters were discussed already. See specifically Table 2.2.1.

We now turn to additional features of the considered region. These relate to the boundary conditions of the discrete flow and transport models. A uniform grid represents the considered region in Figure 4.1.6. Different symbols denote the three kinds of boundary conditions that occur in the flow model. Boundary conditions for the transport model are shown in Figure 4.1.7.

For low to medium water height in the stream, the aquifer is unconfined. Confined conditions will occur only in the vicinity of the stream when the stream has high water. The aquifer is hydraulically connected to the stream via the riverbed which exhibits high permeability. Thus, the stream has dominant influence on the water supply of the region. This influence is determined by hydraulic properties of the riverbed and the difference between piezometric head in the aquifer and water height in the river. The hydraulic properties of the riverbed are reflected in the discrete model by

Table 4.1.1: Main soil and hydrothermal parameters of the application.

k_0	$\approx$	$10^{-10}\,m^2$	soil permeability
n	$\approx$	30%	porosity of aquifer
$\rho_w c_w$	$\approx$	$4.185\frac{MJ}{m^3 \cdot °C}$	heat capacity of water
$\rho_a c_a$	$\approx$	$2.5\frac{MJ}{m^3 \cdot °C}$	heat capacity of aquifer
α_L	$\approx$	$10m$	longitudinal dispersion length
α_T	$\approx$	$1m$	transversal dispersion length

the leakage factor which is set to $5 \cdot 10^{-6}\,1/s$ for all river nodes of the grid. In mathematical terms, the river imposes mixed boundary conditions. Compare the Appendix. A ground water flux into the considered region occurs from the south. The flux is assumed to be a constant $5\,l/s$ for every km of the boundary. This corresponds to Neumann boundary conditions for this part of the boundary. For the remaining part of the boundary, fixed potentials are assumed. Prescribed potentials lead to Dirichlet boundary conditions. In addition, one has to deal with recharge rates, e.g. due to precipitation. While in a three–dimensional setting this constitutes a boundary condition, in the two–dimensional model it is, like the influence of wells, considered in a term for sources and sinks of water. Recharge rates are estimated to be a uniform $6\,l/(s \cdot km^2)$ for the entire region.

The flow simulation is done by `MODFLOW`. The code is described in Section 4.2. In Figure 4.1.3, equipotential lines are plotted. The curvature of the lines indicates that the extraction wells, compare Figure 4.1.2, dominate the flow regime in that part of the region. This dominant influence is also obvious in Figure 4.1.4 which shows the flow regime in the center of the considered region. The main flow direction is west. However, the influence of the river in the north is considerable, and local variations in the main flow direction occur due to the wells. The infiltration well in the eastern part of the center part as well as the extraction wells are clearly distinguishable. For the location of the wells compare also Figure 4.1.2.

For the transport simulation, `MT3D` is used. See Section 4.2 for a description of the code. The infiltration well is given an initial temperature $1°C$ in excess over the temperature of the remaining domain. We view temperature changes as percentual. Boundary conditions differ from those of the flow model in that there are no Dirichlet boundary conditions. The river and the fixed flow potentials impose mixed boundary conditions for the transport model, and the inflow from the southern boundary gives rise to Neumann boundary conditions. The boundary conditions for the computational model are shown schematically in Figure 4.1.7.

Figure 4.1.3: Equipotential lines of the flow model.

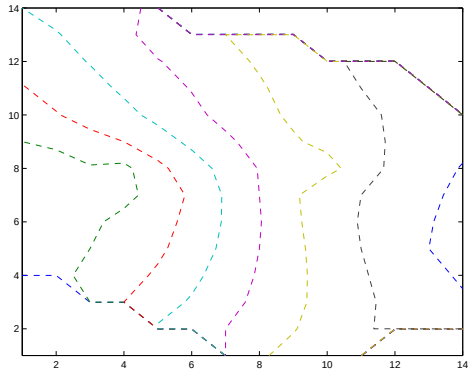


Figure 4.1.4: Flow regime in center part of the considered region.

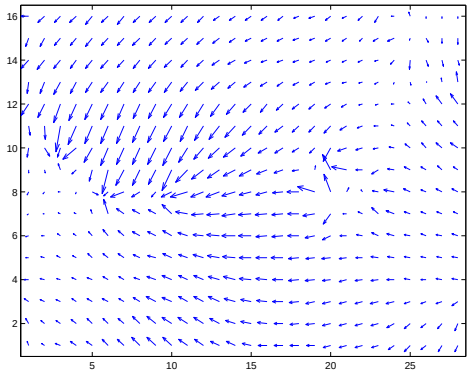
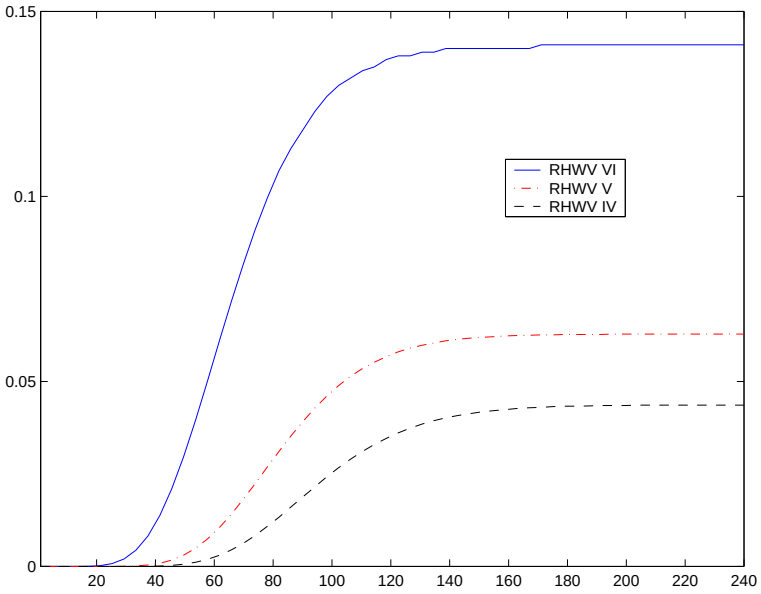


Figure 4.1.5: Development of temperature at drinking water wells. Percentual temperature increase in $°C$ versus time in months.



We now describe the simulation outcome in the original setting. In Figure 4.1.5, temperature increase at the drinking water wells is given. Well VI is nearest to the infiltration well. The wells V and IV are aligned horizontally downstream. Obviously, the further away a given observation point is from the infiltration well, the longer it takes for the plume to reach this point, and the smaller the temperature increase is in general. The temperature development at the drinking water wells shown in Figure 4.1.5 is of primary concern for the optimization. Compare Section 4.1.2. The time frame of the figure corresponds to twenty years. Tracer traveling time from the infiltration well to the first extraction well IV lies between 28 and 34 months. This sig-

nifies that after this time span 50% of final concentration at this well can be measured. Incorporating the retardation factor of 2, this translates to a temperature increase by 50% of final temperature increase at this well about 60 months after $t = 0$. This is confirmed in Figure 4.1.5. Temperature increase is delayed and smaller in value at wells V and IV. We see an increase of 0.15 at the drinking water well that is nearest to the infiltration well. Considering the temperature increase at the infiltration well of about $5°C$, the model accounts for about 75% of the increase exhibited at the drinking water well. This is within sensible margins because seasonal fluctuations in recharge and river data are not taken into account. Note that at all locations a steady state is reached after about twelve years. This justifies to use steady–state well pumping rates for the optimization.

Figure 4.1.6: Boundary conditions for the flow model.

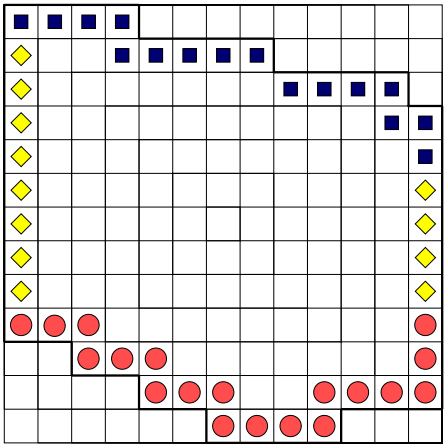
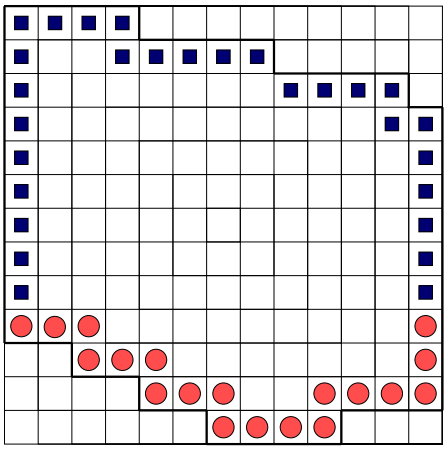


Figure 4.1.7: Boundary conditions for the transport model.



Neumann Mixed Dirichlet

Apart from the main influence of the infiltration and extraction wells, development of the plume within the region also depends on the influence of inflow from the river in the north and on the recharge occurring on the southern boundary. Also, additional wells influence the flow regime. The development of the plume is shown in Figures 4.1.8 and 4.1.9 for a time horizon of five and twenty years. The inflow from the river in the north is visible in the slight bend of the plume. The plume has not yet taken on its final shape after five years. There are no visible changes to its extension any more after twelve years. This is what one expects after seeing the temperature evolution at the drinking water wells in Figure 4.1.5. Note that these computations are done for the original, uncontrolled, case. A detailed analysis and comparison with the case where a control is exercised is done in Section 4.3.

Figure 4.1.8: Temperature plume after 5 years.

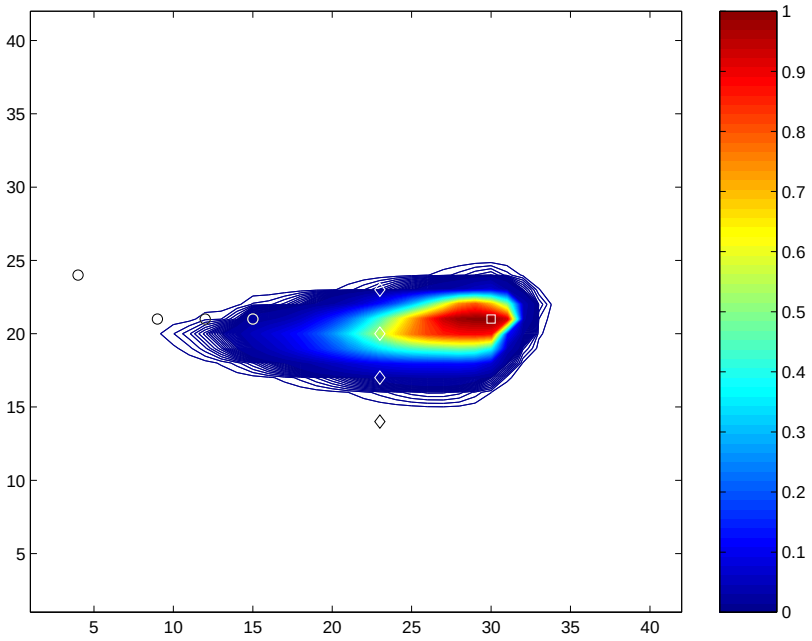
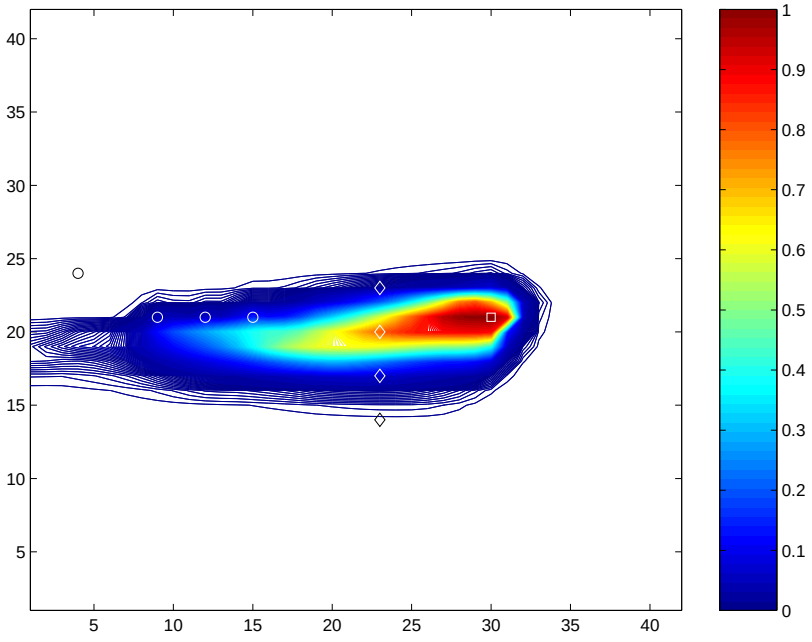


Figure 4.1.9: Temperature plume after 20 years.



4.1.2 The Optimization Goal and Numerical Setting

The overall goal for the optimization is to minimize temperature increase in the vicinity of the drinking water wells at low control cost. The drinking water wells are the wells in the western part of the considered region. Control can in our model be exercised via four wells placed between the infiltration well and the drinking water wells. Compare Figure 4.3.1. The control is exercised by varying the stress rates of these barrier wells.

The objective function for the application is

$$\min_{u \in \mathbb{R}^4} \quad J(u) = \langle u, u \rangle + \langle w, T \rangle \,, \tag{4.1.1}$$

where $u \in \mathbb{R}^4$ is the control variable and where T, dependent on the control, is the temperature distribution in the domain at final simulation time t_1. This is understood as a discrete problem, i.e., $T = T(x, y, t_1) \in \mathbb{R}^m$ is the solution to a discretized version of the state equation described in Section 2.3. Compare Figure 2.3.1. Here, $m \in \mathbb{N}$ is the number of grid points. The nonzero entries of the weight vector $w(x, y)$ are the stress rates of the drinking water wells. The controls u are the steady–state pumping rates of the barrier wells.

This objective function models two different aspects. One goal is to prevent high temperature rises in the drinking water wells. This aspect is taken care of with the part $\langle w, T \rangle$ in Equation (4.1.1). This term allows a weighting of the temperature at these points. Another important point is the cost of any action taken. This is modeled with the product $\langle u, u \rangle$ which takes into account the cost of running the barrier wells.

The objective function (4.1.1) for the application depends on the control u both directly and implicitly via the state T, $T = T(u)$. The decision variables u are the well stress rates which influence the flow regime in the considered region. On the basis of the flow regime, the transport is determined, its output being the state T. The corresponding computations are done by `MODFLOW` and `MT3D`.

This setting comprises numerical difficulties. One problem is the implicit definition of the state T and its computation via software codes. Thus, no analytic gradient is readily available for use within the optimization. Likewise, no analytic second–order information is available. This failure can often be helped – within certain margins that relate to machine precision – via finite–difference approximations to the gradient and possibly even the Hessian. The problematic point is in our case that the function evaluations cannot be considered accurate. The effects on finite–differencing are described in detail in Section 3.1.2 above.

The claimed inaccuracy is due partly to the use of software, partly to modeling issues and data procurement. We use two external software codes for the equation solve. The simulation codes use numerical schemes that are overall first–order accurate. The flow simulation is second–order accurate in space and first–order accurate in time. The transport code is first–order accurate only both in space and time. In addition, the sequential use of the two codes entails rounding errors that cannot be controlled. In the ground water modeling setting, severe inaccuracies are also introduced through modeling and data issues. Two partial differential equations describe the phenomena

of ground water flow and of heat transport with the ground water. The partial differential equations are simplifications of a complex reality. Often, boundary and initial conditions are difficult to handle. Numerous influences overlap, and only the most important can be considered in the numerical model without losing practicability. In addition, data procurement is costly. In practice, soil parameters and boundary and initial conditions may have to be estimated. Even the relevant data cannot in general be measured to an accuracy that might be desirable from a computational standpoint. Piezometric heads, for example, are in general not measured more accurately than in the cm–range. From a practical standpoint, this might often be fully satisfying, however. In this setting, we assume an error margin of about 1% for the output of the computational model. Input to the optimization problem in its black–box formulation are only the control variables u. These stand for well stress rates for which we assume an error margin of about 0.1%.

In the remainder of this section it is explored what the assumed error margins account for in terms of the objective function. The consequences for numerical optimization routines are discussed in Chapter 3. As described there, the application we deal with is perceived as

$$\min_{x \in I\!R^n} \hat{f} : I\!R^n \to I\!R \,.$$

The objective $\hat{f}$ is assumed to be the sum

$$\hat{f} = f + \varphi$$

of a smooth part f with a simple form, e.g. a convex quadratic, and a perturbation φ which we refer to as "noise". We take $f \in C^1(I\!R^n)$ and $\varphi \in L^\infty(I\!R^n)$. It is assumed that the noise φ is "much smaller" than the smooth part, i.e.,

$$\max_{x \in I\!R^n} |\varphi(x)| \ll \max_{x \in I\!R^n} |f(x)| \,.$$

An estimate of the typical size ε of the norm of the error, $||\varphi||$, helps in dealing with the inaccuracy. Compare Section 3.1.2. The input values u are typically of size 10^{-2}. Assuming an error in these values of about 0.1%, a $\mathcal{O}$–calculation leads to an error of size 10^{-10} for the control part $u^T u$ of the objective function (4.1.1). Here, $\mathcal{O}$ stands for the Landau symbol, see e.g. [45]. The weighting vector w also typically is of size 10^{-2}. Since the weighting vector contains the stress rates of the drinking water wells, the error associated with it is comparable to the error in the control variables, i.e., 10^{-5}. The range for the temperature part in the objective function is considerably larger than for the stress rates that have just been discussed. In the uncontrolled case, temperature at the drinking water wells lies between the values 10^{-1} and 10^{-4}, in the controlled case between 10^{-2} and 10^{-5}. Considering an average value of 10^{-3} as relevant and an error of about 1% induced through the combined effects of modeling, data procurement, and the numerical treatment, a $\mathcal{O}$–calculation for the second part $w^T \phi$ leads to an error of size 10^{-9}. In combination with a multiplicative factor $c = 10^6$ which is applied to both parts of the objective function, an absolute upper bound $\varepsilon \approx 10^{-3}$ can be set

on the norm $\|\varphi\|$. This is, relative to typical function values with two to four digits, a noise of 0.001% to 0.00001% in the objective function. Because $\|u\| \approx 10^{-2}$, the results of Section 3.1.2 advise us to use difference increments $h \in [10^{-3}, 10^{-4}]$ in finite–differencing.

In summary, we are faced with an optimization problem with errors in the function evaluation. First– and second–order information, if appropriate, must be approximated via finite differences. This approximation must be performed within ranges determined by the error margin of the model and the data. The situation is rendered even more difficult by the fact that function evaluations are costly. One function evaluation corresponds to the simulation of the flow and the transport processes, i.e., to the sequential solve of two partial differential equations.

This setting poses structural problems to most optimization algorithms and codes. We have thus concerned ourselves with several possibilities to address the problem in question. One choice was an optimization code that can be considered typical for smooth optimization, a quasi–Newton algorithm with line search. It obtains first–order information, if not provided, by finite–differencing under the assumption of accuracy in the order of machine precision. Its deficiencies in the situation of our application is described in Section 3.2. An alternative are methods for nonsmooth functions and methods that are specifically designed for noisy problems. In our numerical experiments, the Nelder–Mead algorithm, see Section 3.3, and Implicit Filtering, see Section 3.4, are used. Not until the numerical treatment of the underlying partial differential equations in the software codes is described we turn to the presentation of the numerical results.

4.2 The Simulation Codes

Throughout this chapter we consider an optimal control problem that is governed by partial differential equations. The optimization problem is given in Figure 2.3.1 in the unconstrained formulation that we use. The partial differential equations model ground water flow and heat transport in an aquifer, respectively. It can be justified to solve the two equations sequentially. Despite the possibility of sequentially solving the equations, the computational cost of the solve is the determining factor in our application and in similar applications. A solve of the governing partial differential equations is what we call a "simulation". In each step of an iterative method that is intended to find a solution to the optimal control problem, a simulation has to be performed.

To perform a simulation, two software codes are used. The first is a flow model, `MODFLOW`, and the second is a transport model, `MT3D`. `MODFLOW` is as a model especially well known by many people working on practical applications in ground water modeling. In 1997, a college textbook [27, Ch. 7.4] calls the code which was released in 1984 "the de facto standard code for aquifer simulation." `MT3D` is in numerous respects written such as to conform with `MODFLOW`. The two codes use an identical spatial discretization, for instance, and data transfer between the codes is provided for.

The company with which we consult on this project and from which this applied problem originates has acquired experience with both codes. The results are trusted within a rather large error margin. It seems preferable to the engineers of this company to use relatively simple, but well–known, implementations rather than to work with sophisticated, state of the mathematical art, simulation codes. Our goal in this part of the project is not to choose the latest and most reliable codes but to provide tools for optimization built on the given simulation setting.

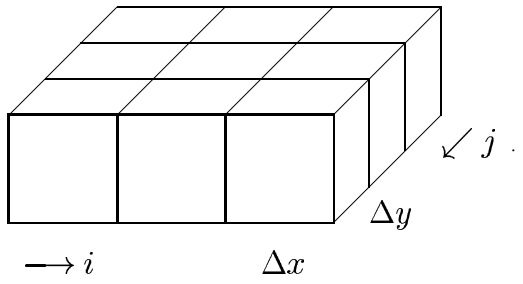
In this section we describe the two codes that our numerical experiments are based on. For both codes the discretization method is described along with their overall capacities. Both are finite–difference `FORTRAN` codes. Common to all finite–difference methods is the idea to replace the differential quotients in a differential equation by difference quotients and to subsequently solve the discretized equations. For a treatment of accuracy and stability for finite–difference–schemes see e.g. [80].

4.2.1 The Flow Code

The flow code is called `MODFLOW` which stands for *modular flow model*. `MODFLOW` is a three–dimensional finite–difference ground water flow model. It has a modular structure which allows it to be easily modified to adapt the code for a particular application. The software is distributed by the United States Geological Survey (USGS). The original code was developed with an extensive User's Guide and released in 1984. This version was superseded in 1988 [59]. It is currently the most used numerical model in the U.S. Geological Survey for ground water flow problems. The code has been supported strongly by the U.S. Geological Survey and is readily available. `MODFLOW` is written in `FORTRAN77`. A detailed documentation for `MODFLOW` is contained in [59]. `MODFLOW` is a three–dimensional flow model in that the three–dimensional region of interest is considered as a stack of different layers. The layers should be chosen such that flow conditions are relatively homogeneous within each layer. Then the hydraulic approach, compare Section 2.1 and the Appendix, can be applied for each layer. Exchange processes between the different layers are taken into account. `MODFLOW` simulates steady and nonsteady flow in an irregularly shaped flow system in which aquifer layers can be confined, unconfined, or a combination of confined and unconfined. Flow from external stresses, such as flow to wells, areal recharge, evapotranspiration, flow to drains, and flow through river beds, can be simulated. Specified head and specified flux boundaries can be simulated as can a head dependent flux across the model's outer boundary. The latter allows water to be supplied to a boundary block in the modeled area at a rate proportional to the current head difference between a "source" of water outside the modeled area and the boundary block.

The aquifer system is discretized into a mesh of blocks, or cells, the locations of which are described in terms of rows, columns, and layers. For a single–layer model like it is appropriate for our application this is visualized in Figure 4.2.1. The rectangular discretization results from a grid of mutually perpendicular lines that may be variably spaced. The formulation is block–centered, i.e., the point where the hydraulic

Figure 4.2.1: Spatial discretization in MODFLOW and MT3D.



head is calculated, the node, is placed at the center of the cell. Compare Figure 4.2.2. The chemical and hydraulic parameters such as hydraulic conductivities are assumed to be uniform over the extent of the cell. It is the responsibility of the user to subdivide the flow region into blocks in which the medium properties can be assumed to be uniform. Simulation time in the flow model is divided into so called stress periods. These are time periods during which all external stress parameters, i.e., sink and source data, are constant. Stress periods are divided into time steps if the simulation is transient.

The ground water flow equation

$$S\,\frac{\partial}{\partial t}h(x,y,t) = \nabla\cdot(K(x,y)\,\nabla h(x,y,t)) + q^h(x,y,t) \tag{4.2.1}$$

is solved using a finite–difference approximation. This is easily traced with a recourse to the derivation of the equation as a combination of Darcy's law and a continuity equation. Through this we go now. For simplicity we assume a subdivision of the domain into squares. We abbreviate the intervals of size Δx in x–direction and of size Δy in y–direction by Δs where appropriate.

The piezometric head is denoted $h_{i,j}$ at node (i,j), $(i=1,\ldots,I; j=1,\ldots,J)$. The discrete hydraulic gradient has the entries

$$\frac{h_{i+1,j}-h_{i,j}}{\Delta x}$$

for the change in x-direction between the neighboring nodes (i,j), $(i+1,j)$ and

$$\frac{h_{i,j+1}-h_{i,j}}{\Delta y}$$

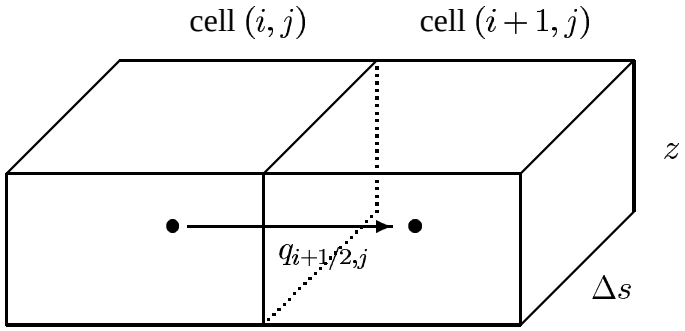
for the change in y–direction between the neighboring nodes $(i,j+1)$, (i,j). The values $K_{i+1/2,j}$ of the hydraulic conductivity between the neighboring nodes (i,j), $(i+1,j)$ are known. Flow between the nodes (i,j), $(i+1,j)$ is given by

$$q_{i+1/2,j} = -K_{i+1/2,j}\cdot\frac{h_{i+1,j}-h_{i,j}}{\Delta x}\,\Delta y\,z, \tag{4.2.2}$$

where z is the vertical extension of the considered cell, so that $\Delta y\, z$ is the area perpendicular to the flow. A visualization is found in Figure 4.2.2. Analogous to the flow in x–direction, flow in y–direction between cells $(i, j+1)$, (i, j) is given by

$$q_{i,j+1/2} = -K_{i,j+1/2} \cdot \frac{h_{i,j+1} - h_{i,j}}{\Delta y} \Delta x\, z.$$

Figure 4.2.2: Flow between adjacent cells in MODFLOW.



The continuity equation states that the excess of inflow over outflow for the cell must equal the mass of water accumulated in the cell over the considered time span. Inflow from external sources to the cell (and outflow to external sinks) must be taken into account, a discretized version of which can be written as

$$q^h_{ij} = q^d_{ij}\, h_{ij} + q^{ind}_{ij}.$$

Inflow from external sources can either be dependent on the head in the cell or be independent of it, therefore the split into the terms q^d, q^{ind}. The spatially discretized flow equation (4.2.1) can be written as

$$S_{ij} \frac{\Delta h_{ij}}{\Delta t} \Delta x\, \Delta y\, z = -(q_{i,j+1/2} + q_{i,j-1/2} + q_{i+1/2,j} + q_{i-1/2,j}) + q^h_{ij}\, \Delta x\, \Delta y\, z.$$

This we transform to

$$\begin{aligned} S_{ij} \tfrac{\Delta h_{ij}}{\Delta t} = \;& K_{i,j+1/2} \cdot \tfrac{h_{i,j+1} - h_{i,j}}{(\Delta s)^2} + K_{i,j-1/2} \cdot \tfrac{h_{i,j-1} - h_{i,j}}{(\Delta s)^2} \\ & + K_{i+1/2,j} \cdot \tfrac{h_{i+1,j} - h_{i,j}}{(\Delta s)^2} + K_{i-1/2,j} \cdot \tfrac{h_{i-1,j} - h_{i,j}}{(\Delta s)^2} \\ & + q^d_{ij}\, h_{ij} + q^i_{ij}. \end{aligned}$$

Here, S_{ij} is the specific storativity of cell (i, j), and $\frac{\Delta h_{ij}}{\Delta t}$ is the finite–difference approximation to the time derivative of the piezometric head to which we turn now.

We define a subdivision of the considered time interval, where for simplicity we again assume equidistant intervals of length Δt. The approximation to the time derivative of the piezometric head at time t_m is given by

$$(\frac{\Delta h_{ij}}{\Delta t})^m = \frac{h_{ij}^m - h_{ij}^{m-1}}{\Delta t},$$

where h_{ij}^m denotes the piezometric head at time step m, $m = 0, \ldots, M$. The implicit Euler scheme which is unconditionally stable leads to the set of equations

$$\begin{aligned} S_{ij}\,\tfrac{h_{ij}^m - h_{ij}^{m-1}}{\Delta t} = \; & K_{i,j+1/2} \cdot \tfrac{h_{i,j+1}^m - h_{i,j}^m}{(\Delta s)^2} + K_{i,j-1/2} \cdot \tfrac{h_{i,j-1}^m - h_{i,j}^m}{(\Delta s)^2} \\ & + K_{i+1/2,j} \cdot \tfrac{h_{i+1,j}^m - h_{i,j}^m}{(\Delta s)^2} + K_{i-1/2,j} \cdot \tfrac{h_{i-1,j}^m - h_{i,j}^m}{(\Delta s)^2} \\ & + q_{ij}^{d,m}\, h_{ij} + q_{ij}^{\mathit{rnd},m}. \end{aligned}$$

This scheme is accurate of order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta^2 s)$ for sufficiently smooth solutions h to (4.2.1). Simple transformations show that a linear system in the unknowns h_{ij} ($i = 1, \ldots, I; j = 1, \ldots, J$) has to be solved for each time step.

4.2.2 The Transport Model

As a transport model, `MT3D` is used. `MT3D` stands for Modular Three–Dimensional Transport Model. The software originates from the Environmental Protection Agency (EPA) of the United States and is distributed by the Center for Subsurface Modeling Support (CSMoS) of the Robert S. Kerr Environmental Research Laboratory. The code, written in `FORTRAN77`, is documented in [67]. `MT3D` can be utilized for a variety of hydrologic settings. The model allows for several different transport boundary conditions. These include e.g. confined, unconfined or variably confined and unconfined aquifer layers, specified concentration or mass flux boundaries, and the solute transport effects of external sources and sinks such as wells, drains, rivers, areal discharge and evapotranspiration. `MT3D` has been designed as a solute transport model, i.e., to simulate advection, dispersion and chemical reactions of dissolved constituents in ground water systems. In our numerical tests, `MT3D` is adapted to simulate heat transport. Since the partial differential equations modeling heat and solute transport are similar, the adaptation is readily done with the correct choice of parameters. This procedure is justified in Section 2.2.

It is assumed by the model that changes in the concentration or temperature field will not significantly affect the flow field. This is equivalent to assuming that the partial differential equations for flow and transport are not coupled. Then, the transport equation can be solved independently of the flow equation. `MT3D` can be used in conjunction with any block–centered finite–difference flow model, in particular in conjunction with `MODFLOW` which is described above. In our numerical experiments, `MODFLOW` is used to generate the velocity field of the ground water system under investigation. Like `MODFLOW`, `MT3D` uses a modular structure and can be considered

a three–dimensional model in that it admits several layers with different soil properties. MODFLOW and MT3D use an identical spatial discretization. The aquifer system is discretized into a mesh of cells described in terms of rows, columns, and layers as we have seen above. Compare Figure 4.2.1. Both use a block–centered formulation. Chemical and hydraulic parameters are assumed to be uniform over the extent of the cell. Temporal discretization is not identical for the two codes, though. Simulation time in the flow model is in the transient case divided into time steps. Due to the implicit time stepping scheme, the size of the time step is not restricted by stability considerations. In the transport model, simulation is based on the flow regime provided by the flow model, and on an explicit transport solution. The length of the time step used for the head solution may be too large for the transport solution. There are stability criteria associated with the latter that we address below. Each time step of the flow solution is therefore further divided into smaller time increments during which the flow is considered to be constant. The small time increments are called transport steps. Since only the transport step is needed in this section, we denote it by Δt like the time step of the flow model.

For the discretization of the transport equation there are two basic choices in the code. One is a "pure" finite–difference scheme, pure in that a finite–difference approximation is used for all parts of the transport equation. The alternative is a method of characteristics. These approaches differ in how they treat the advective part of the transport equation, compare (2.2.5),

$$\frac{\partial}{\partial t}T(x,y,t) = \nabla\cdot(\kappa\, T(x,y,t)\, v(x,y,t)) + \nabla\cdot(D(v)\,\nabla T(x,y,t)) + q^T(x,y,t).$$

The dispersive, second–order, part is discretized with fully explicit central finite differences, and the term accounting for sources and sinks is incorporated in a straightforward way. The advective term is the numerically difficult part,

$$\begin{aligned}&\nabla\cdot(\kappa\, T(x,y,t)\, v(x,y,t))\\ &= \tfrac{\partial}{\partial x}\,\kappa\, T(x,y,t)\, v_x(x,y,t) + \tfrac{\partial}{\partial y}\,\kappa\, T(x,y,t)\, v_y(x,y,t).\end{aligned} \tag{4.2.3}$$

Here, $v_x(x,y,t)$ and $v_y(x,y,t)$ denote the x– and y–directions of the velocity vector v. Compare Equation (2.1.7). The velocity can be computed from the piezometric head h, a discretized version h_{ij} $(i = 1,\ldots,I; j = 1,\ldots,J)$ of which is furnished by the flow code. The x–velocity at the cell interfaces between nodes (i,j), $(i+1,j)$ is given by, compare (4.2.2),

$$v_{x_{i+1/2,j}} = -\frac{1}{n}\,K_{i+1/2,j}\cdot\frac{h_{i+1,j}-h_{i,j}}{\Delta x}.$$

Analogously, we define $v_{x_{i-1/2,j}}$ as the velocity in x–direction between nodes $(i-1,j)$, (i,j) and $v_{y_{i,j-1/2}}$, $v_{y_{i,j+1/2}}$ as the corresponding velocities in y–direction between nodes $(i,j-1)$, (i,j) and (i,j), $(i,j+1)$.

Depending on the flow direction the quantity $T_{x_{i+1/2,j}}$ is defined,

$$T_{x_{i+1/2,j}} = T_{i,j} \quad \text{if } v_{x_{i+1/2,j}} > 0, \quad T_{x_{i+1/2,j}} = T_{i+1,j} \quad \text{if } v_{x_{i+1/2,j}} < 0, \tag{4.2.4}$$

and $T_{x_{i-1/2,j}}$, $T_{y_{i,j-1/2}}$, $T_{y_{i,j+1/2}}$ are defined analogously. A discrete version of (4.2.3), neglecting κ, can be written as

$$\frac{(vT)_{x_{i+1/2,j}} - (vT)_{x_{i-1/2,j}}}{\Delta x} + \frac{(vT)_{y_{i,j+1/2}} - (vT)_{y_{i,j-1/2}}}{\Delta y}. \tag{4.2.5}$$

Here we abbreviate $v_{x_{i+1/2,j}} T_{x_{i+1/2,j}} = (vT)_{x_{i+1/2,j}}$ and in an analogous way the quantities $v_{x_{i-1/2,j}} T_{x_{i-1/2,j}}$, $v_{y_{i,j+1/2}} T_{y_{i,j+1/2}}$, and $v_{y_{i,j-1/2}} T_{y_{i,j-1/2}}$. This upstream weighting scheme is used for the spatial discretization in the "pure" finite–difference method. With an explicit time stepping,

$$(\frac{\Delta T_{ij}}{\Delta t})^{m+1} = \frac{T_{ij}^{m+1} - T_{ij}^{m}}{\Delta t},$$

the advective part of the transport equation becomes

$$\frac{T_{ij}^{m+1} - T_{ij}^{m}}{\Delta t} = \kappa \frac{((vT)^m_{x_{i+1/2,j}} - (vT)^m_{x_{i-1/2,j}} + (vT)^m_{y_{i,j+1/2}} - (vT)^m_{y_{i,j-1/2}})}{\Delta s},$$

or, equivalently,

$$T_{ij}^{m+1} = T_{ij}^{m} + \kappa \frac{\Delta t}{\Delta s} ((vT)^m_{x_{i+1/2,j}} - (vT)^m_{x_{i-1/2,j}} + (vT)^m_{y_{i,j+1/2}} - (vT)^m_{y_{i,j-1/2}}). \tag{4.2.6}$$

The scheme (4.2.6) can only be stable if

$$\kappa \frac{\Delta t}{\Delta s} \max(v_x, v_y) \leq 1, \tag{4.2.7}$$

a condition which we refer to as Courant condition. Compare [80, Ch. 5.7]. Consistency is of order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta s)$.

The second basic choice for the evaluation of the advective term is a method of characteristics. This is based on a major property of hyperbolic equations, observable for instance for the one–way wave equation

$$v_t + a\, v_x = 0, v(x,0) = f(x) \quad (a > 0, x \in I\!R, t > 0),$$

a one–dimensional model problem. The solution $v(x,t) = f(x - a\,t)(x \in I\!R, t > 0)$ is constant along any characteristic curve, $x - a\,t = const.$ The solution is translated to the right with propagation velocity $a > 0$ in spatial domain with time. In the code, an approach referred to as (plain) method of characteristics, a modified method of characteristics, and a hybrid method of characteristics are implemented.

Common to these approaches is the use of a particle–tracking technique to approximate the advective component of the transport process. Since any particle–tracking technique requires the evaluation of velocity at an arbitrary point from hydraulic heads that are calculated at nodal points, it is necessary to use a velocity interpolation scheme.

The one used in this model is piecewise linear. For a point (x_p, y_p) within cell (i, j) one interpolates the x–velocity component by, compare (4.2.2),

$$\begin{aligned} q_x(x_p, y_p) &= (1 - \tfrac{x_p-(x_i-1/2\Delta x)}{\Delta x})\, q_{i-1/2,j} + \tfrac{x_p-(x_i-1/2\Delta x)}{\Delta x}\, q_{i+1/2,j} \\ &= (\tfrac{1}{2} - \tfrac{x_p-x_i}{\Delta x})\, q_{i-1/2,j} + (\tfrac{1}{2} + \tfrac{x_p-x_i}{\Delta x})\, q_{i+1/2,j} \end{aligned} \tag{4.2.8}$$

and the y–velocity component analogously. With the velocity field known, a numerical tracking scheme can be used to move particles from one position to another to approximate the advection of the contaminant or heat. Usually, the first–order Euler algorithm is used for particle tracking,

$$x^{m+1} = x^m + \Delta t \cdot v_x(x^m, y^m), \quad y^{m+1} = y^m + \Delta t \cdot v_y(x^m, y^m).$$

Here, x^{m+1}, y^{m+1} are the particle coordinates at the new time level, x^m, y^m are the particle coordinates at the previous time level, and $v_x(x^m, y^m)$, $v_y(x^m, y^m)$ are the x– and y–components of the Darcy velocity vector evaluated at (x^m, y^m). As before, Δt is the size of the transport step. An upper bound for the size of the transport step, Δt, is determined from the Courant condition (4.2.7),

$$\Delta t \leq \gamma_C\, \Delta s\, \min(\frac{1}{v_x}, \frac{1}{v_y}).$$

The Courant number γ_C represents the number of cells a particle is allowed to move in any direction in one transport step, i.e.,

$$|\frac{x^{m+1} - x_p}{\Delta x}| \leq \gamma_C\,, \quad |\frac{y^{m+1} - y_p}{\Delta y}| \leq \gamma_C.$$

A uniform step size Δt is used for all moving particles during each transport step in the particle–tracking calculations. For particles located in areas of relatively uniform velocity, the first–order Euler algorithm usually has sufficient accuracy. For particles located in areas of strongly converging or diverging flow, near wells, for instance, a first–order algorithm may not be sufficiently accurate unless the transport step Δt is very small. For these circumstances, a higher–order algorithm is provided with the fourth–order Runge–Kutta method.

The practical implementation of the method of characteristics places either a constant number of particles in each cell or determines a number of particles for a cell according to its importance for advective transport. The importance is measured in terms of temperature changes in the vicinity of the cell relative to overall temperature changes. This is more efficient numerically than the constant number of particles. An update of particle placement and temperature distribution is done for the cells after each transport step in which the particles are moved through the domain according to the interpolated velocities. After the evaluation of the advection–driven term of the transport equation, the remaining parts are solved based on the partially computed new temperature profile, and a weighted update is done for the temperature distribution at the current time step.

Alternative to the just presented method of characteristics, the code uses a modified method of characteristics that requires less computational effort. In the modified method of characteristics, rather than moving particles forward with the flow, a particle located at node (i, j) at the end of the current time step is tracked backward. Temperature in the cell (i, j) is then set to the temperature valid at initial particle location. This is calculated via the linear interpolation (4.2.8). This method, despite special provisions for sources and sinks, requires considerably less effort than the method of characteristics. However, it furnishes usable results only either on relatively coarse grids for uniform flow or on relatively fine grids.

Due to these features, a combination of the method of characteristics and the modified version is provided for in the transport code, the so called hybrid method of characteristics. In the description of the numerical results in Section 4.3 we refer to this hybrid method as method of characteristics without further notice. The decision which method should be used (method of characteristics or modified method of characteristics) is, assuming a uniform grid and a relatively uniform dispersion coefficient, based on the socalled mesh Peclet number P. The modified method of characteristics loses accuracy for high flow velocities. This is measured via $\|v\|_1$ in the code. We denote that node by (i_0, j_0) that furnishes maximal flow velocity over the entire (discretized) domain. A coarse spatial discretization Δs makes the computational effort of the method of characteristics acceptable, and a large dispersion coefficient increases overall computational effort. The mesh Peclet number P is defined as

$$P = \frac{\|v_{i_0,j_0}\|_\infty \Delta s}{D(v_{i_0,j_0})},$$

where (i_0, j_0) is the node at which the largest velocity component is computed. The mesh Peclet number suggests to use the method of characteristics if large and the modified method of characteristics if small. If $P > 10$ or if $P < 0.1$, one method is chosen in the code for the entire domain. Otherwise the choice is done cell–wise according to the importance of the considered cell for advective transport. This is measured in terms of temperature changes in the vicinity of the cell relative to overall temperature changes.

4.3 Numerical Results

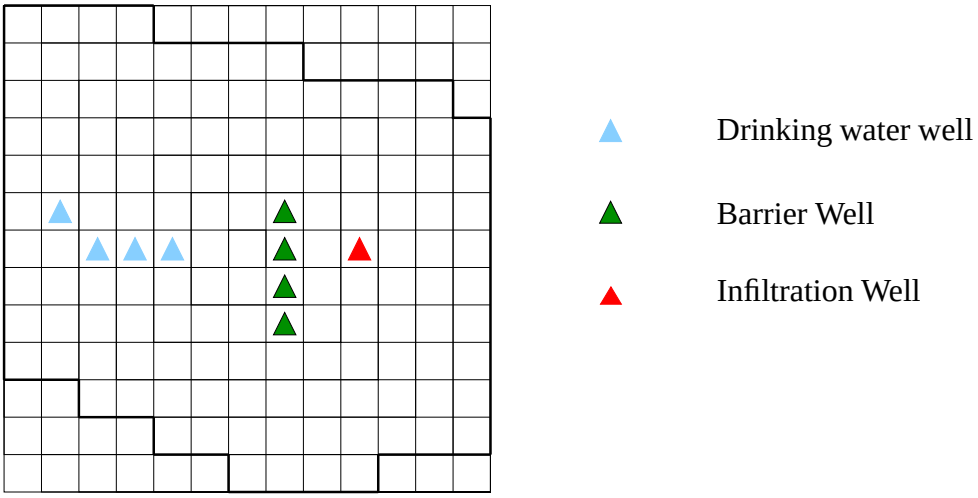
Having fully described the application along with the simulation codes and the chosen optimization routines, we are now in a situation to assemble the numerical results. Recall the objective function for the application,

$$\min_{u \in \mathbb{R}^k} \quad J(u) = \langle u, u \rangle + \langle w, T \rangle \,, \tag{4.3.1}$$

where $u \in \mathbb{R}^4$ is the control variable and $T = T(u, t_1) \in \mathbb{R}^m$ is the temperature distribution in the domain at time t_1. The weight vector w contains as nonzero entries

the stress rates of the drinking water wells. The number of grid points is $m \in I\!N$. The decision variables u are the stress rates of the barrier wells located between the infiltration well and the drinking water wells. See Figure 4.3.1. We assume that the top of the grid lies in northern direction and accordingly describe well location. The drinking water wells are referred to as wells III, IV, V, and VI. Well VI is the one nearest to the infiltration well, well V lies next to well VI in a horizontal line, well IV lies north of well V, and well III is farthest away from the infiltration well. In the objective function, the temperatures at the drinking water wells are weighted with their respective stress rates. These are $0.0325[m^3/s]$ at well VI, $0.0119[m^3/s]$ at well V, $0.0440[m^3/s]$ at well IV, and $0.0461[m^3/s]$ at well III. The remaining entries of the weighting vector w are zero so that the decision is based solely on control cost and temperature distribution at the drinking water wells.

Figure 4.3.1: Well location within the grid.



In the simulation and ensuing optimization, grids of size 14×14, 28×28, 42×42, 56×56, and 70×70 are used. The number of nodes ranges between 196 and $4,900$. See Table 4.3.1. The mesh sizes correspond to distances of $250m$, $125m$, $83.3m$, $62.5m$, and $50m$, respectively, in the application. The computations are instationary. The time horizon for the optimization is twenty years. We equate 360 days to one year. Using a time horizon of twenty years is justified because an equilibrium is reached well within this time period. Compare Figure 4.1.5. After about twelve years of simulated time, a steady state is reached at the drinking water wells. For the flow simulation, 60 time steps of length 120 days are used. For each time step of the flow simulation, 150 time steps are used in the transport simulation. The time step of the transport simulation is called a transport step. Each simulation thus requires 9000 transport steps. The simulation is in the uncontrolled case described already in Section 4.1. The controlled case is discussed below.

We now turn to the optimization results. As mentioned before, the Implicit Filtering and Nelder–Mead algorithms are employed for this task. They are tested for

Table 4.3.1: Number of nodes for different grid sizes.

grid size	14	28	42	56	70
# of nodes	196	784	1,764	3,136	4,900

different grids and starting values. Results are presented and interpreted for various combinations and parameter settings.

The Nelder–Mead algorithm is described in Section 3.3. We make use of a self–coded `FORTRAN` implementation. The modifications advocated by Kelley [51] are incorporated. The Implicit Filtering algorithm is described in Section 3.4. See that section for full references. In our numerical tests, the `FORTRAN` implementation `IFFCO` of Implicit Filtering is put to use. See [17] for handling. This implementation takes box constraints on the decision variables into account, i.e., in our case it allows to set upper and lower bounds on the well rates. This can prove practical in many applications. In our application, for instance, due to capacity restrictions, too large stress rates for the barrier wells cannot be accepted. However, the capacity bounds that are set in our computations are generally not attained. In very few instances only one of the decision variables takes on these extreme values. This rare occurrence is attributable to the bad numerical performance of one discretization scheme that can be selected in the transport code, the method of characteristics. We address the results obtained with this discretization method at the end of this section. First, we describe the results obtained when an upwind scheme is employed in the transport code. With this scheme, none of the decision variables reaches the preset capacity bounds. Thus, even though box constraints are incorporated in `IFFCO`, we consider the treatment of our application as unconstrained optimization.

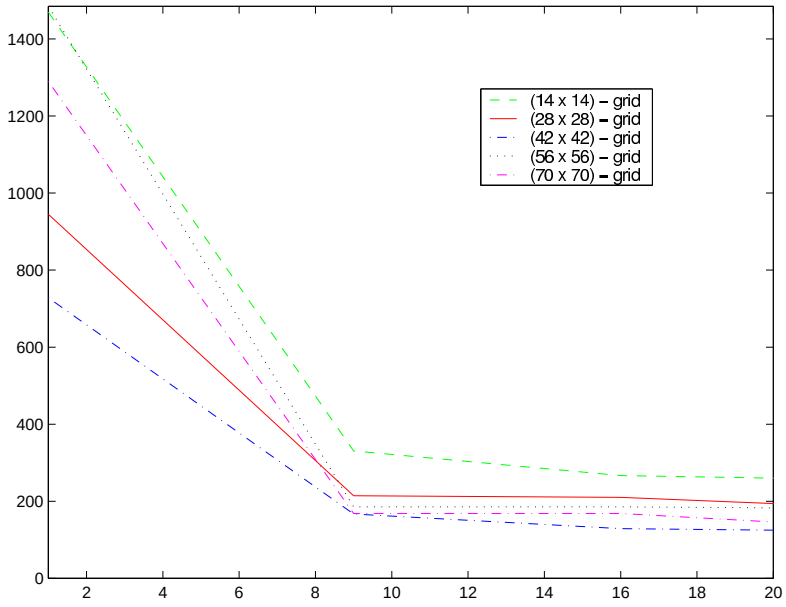
In Figure 4.3.2 we see the development in `IFFCO` iterations over the first 20 function evaluations. Computations are shown for all grids. For the optimization on the different grids identical starting points are chosen. The starting point for the iteration is $u_0 = 0$. A first observation is that the corresponding function values are not identical on the different grids. Also, there is not a consistent development in the function values at the origin with grid refinement. See Table 4.3.2. This point we discuss now, before addressing results of the optimization.

Table 4.3.2: Function values at $u_0 = 0$ for different grid sizes.

grid size	14	28	42	56	70
$J(0)$	1469.5940	945.0380	728.02555	1484.6681	1289.7369

The differences are due to finite–dimensional effects and they are also present in very simple example models. We set up a simplification of the application to detect

Figure 4.3.2: Development in `IFFCO` iterations. Function values versus number of function evaluations for different grids.



the main influence. In this simplified model, the location of the wells is maintained. The shape of the domain and all boundary conditions for the flow and the transport equations are strongly simplified. Parameter values are also strongly simplified. For instance, for the hydraulic conductivity one average value is used over the entire region. Flow in the simplified model takes place from fixed potentials on the eastern boundary of the domain to fixed potentials on the western boundary of the domain. The river which dominates flow in the northern part of the domain for the full application is taken out of the model. Thus, the flow profile is considerably simpler than in the full application described so far, and local influences are by far smaller in size.

The objective function is still given by (4.3.1). At the point $u_0 = 0$, only the temperature part of the objective function is nonzero. In the simplified model like in the full application, at u_0 there are significant changes in the temperatures computed at the extraction wells for the different grids. While the overall reduction in temperature at all considered locations over the first three grids may be attributed to a better approximation of the underlying continuous problem, the increase in function value due to temperature rises on the grid with 56×56 nodes is surprising. This can be explained by scrutinizing the finite–dimensional model. The location of the wells within the region is for each grid recomputed from the exact location. Thus the distances between the grid positions of the infiltration well and the extraction wells vary. These distances along with the average distance and a weighted distance are given in Table 4.3.4. The weighted distance is determined analogously to the weighting via w in the cost function (4.3.1). A comparison between the change in cost given in Table 4.3.3 and the

change in the weighted distance reveals that the unexpected increase in cost on the (56×56)–grid cannot be solely, but certainly to a large part, attributed to well location.

Table 4.3.3: Temperature increase at extraction wells and value of cost function at origin on different grids for simplified application model.

grid size	*VI*	*V*	*IV*	*III*	$J(0)$	change
14	0.575	0.687	0.18100	0.201	4640.010	
28	0.392	0.556	0.03220	0.245	3544.610	-23.6%
42	0.310	0.475	0.01210	0.180	2795.690	-21.1%
56	0.317	0.486	0.01590	0.209	2990.180	$+7.0\%$
70	0.300	0.462	0.00923	0.194	2793.452	-6.6%
84	0.316	0.449	0.00811	0.159	2603.964	-6.8%

Table 4.3.4: Distance between infiltration and extraction wells on different grids. Effects on cost for simplified application model.

grid size	*VI*	*V*	*IV*	*III*	Ø	weighted	change
14	1500.0	1250.0	2015.0	1520.7	1571.4	222.4	
28	1500.0	1250.0	2140.0	1750.0	1660.0	238.5	$+7.2\%$
42	1499.4	1249.5	2180.0	1749.3	1669.6	240.2	$+0.7\%$
56	1501.3	1187.5	2139.4	1688.1	1629.1	234.9	-2.2%
70	1500.0	1200.0	2164.5	1750.5	1653.8	239.0	$+1.7\%$
84	1501.8	1209.3	2141.3	1711.4	1641.0	236.3	$+1.1\%$

Similar effects occur in the full application, amplified through a complex flow profile. This motivates us to consider the full application for two different sets of well locations. First we hold on to the well location considered so far in which the coordinates of the wells within the region are for each grid recomputed from the exact location. Here, the distances between the infiltration well and the extraction wells vary. Second, we also work with "equidistant" wells. This signifies that the distances between the infiltration well and the extraction wells are identical on all grids. The optimization results for that second case are shown only later in this presentation, though. In the ensuing treatment, due to the effects just explained, we show the optimization results for the full application on the different grids on a logarithmic scale. The qualitatively similar behavior on the different grids is still clearly visible. Compare Figures 4.3.3 and 4.3.4.

It was mentioned already that we consider function evaluations as expensive in our application and in similar applications, where each function evaluation involves the solution of partial differential equations. For the CPU time required on a Sun Ultra 60 (1 CPU) for our application see Table 4.3.5. A doubling of the number of subintervals in one spatial direction leads to an increase in the overall number of nodes by a factor of

Figure 4.3.3: Development in IFFCO iterations over 100 function evaluations. Function values versus number of function evaluations for different grids.

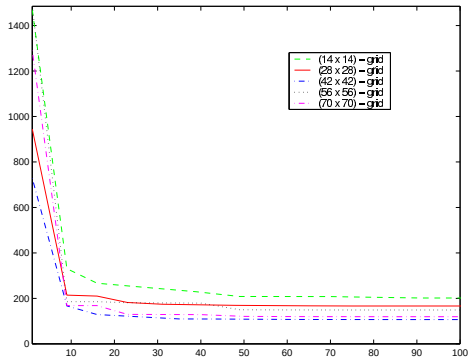
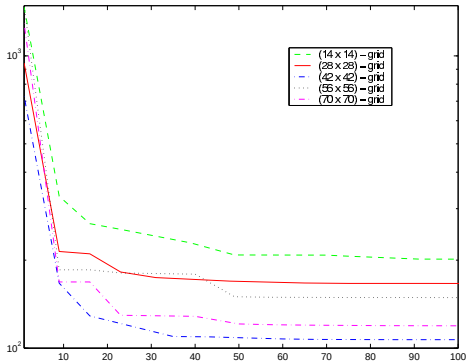


Figure 4.3.4: Development in IFFCO iterations over 100 function evaluations. Function values on logarithmic scale versus number of function evaluations for different grids.



4. This is clearly reflected in the reported computation times that grow proportionally. The simulation times for the considered grid sizes range between less than a minute for the coarsest grid and more than ten minutes for the two largest grids. Simulation times for very difficult applications are typically much higher, but our work here is conceptional. Anticipation of high cost was one reason to choose the Nelder–Mead algorithm which is usually considered as economical in terms of function evaluations, see [55]. Due to cost considerations we typically stop iterating after about 50 function evaluations, sometimes only after 100. An optimization run on the (56×56)–grid, admitting 50 function evaluations, already takes more than 8 hours of computation time. Typically, progress is fast in the first steps of IFFCO. In the cases where more function evaluations were admitted, considerable improvement in the cost function was never achieved. Compare e.g. Figures 4.3.2 and 4.3.3.

Table 4.3.5: CPU time required for simulation of full application on different grids.

grid size	14	28	42	56	70
time in s	42	151	353	614	798

The Nelder–Mead algorithm typically also exhibits fast decrease in the first iterations. For the same starting point $u_0 = 0$ as for the Implicit Filtering iterations shown in Figure 4.3.2, Nelder–Mead iterations are shown in Figure 4.3.5. As observed above for the Implicit Filtering outcome on different grids, the solutions computed by the Nelder–Mead algorithm are not identical on the different grids. Finite–dimensional effects, especially well location, are for a large part responsible for this outcome. Considering also the assumed inaccuracy in the function it would be very surprising to see

high fidelity in the computed coordinates.

But the computed solutions are of practical use. First, they indicate that extraction of heated water at the barrier wells is a sensible thing to do. Temperature increase at the extraction wells is reduced significantly due to action of the barrier wells. This result is visualized in Figures 4.3.10 through 4.3.13. Infiltration of cold water, alternative to the extraction of heated water, has proven less effective. Also, the computed stress rates for the barrier wells are sensible from a practical point of view in that they are in the same range as the rate at the infiltration well. As a third point, the solutions can be used to practical advantage in that they reveal with mesh refinement which barrier well is of most importance. This is the well situated on the same horizontal grid line as the infiltration well. A typical result, this one furnished by the Nelder–Mead algorithm on the (42×42)–grid for a budget of 50 function evaluations and a given infiltration of $0.02735[m^3/s]$ of warm water, is to extract $0.0256[m^3/s]$ from the barrier well located on the same horizontal grid line as the infiltration well, $0.00398[m^3/s]$ from the well located north of this one, $0.00192[m^3/s]$ from the well located immediately south of the first, and to infiltrate about the same amount of cold water, $0.00279[m^3/s]$, via the southernmost barrier well. Similar conclusions as described above for the Nelder–Mead outcome can be drawn from the solutions computed by the Implicit Filtering routine. For the same mesh and an identical budget of function evaluations, `IFFCO` suggests to extract $0.0260[m^3/s]$ from the barrier well located on the same horizontal grid line as the infiltration well, $0.00504[m^3/s]$ from the northern well, $0.00293[m^3/s]$ from the well located immediately south of the first, and to infiltrate $0.00216[m^3/s]$, via the southernmost barrier well. The differences between the stress rates computed by the Implicit Filtering and Nelder–Mead algorithms are often of size 10^{-3}, and of size 10^{-4} for the finer grids for the most used barrier well.

The (42×42)–grid is the most appropriate for our application of all considered grids. The "real–life" distance of about $80m$ that corresponds to this number of grid points already represents a very good resolution of the considered region, but the problem size is still practical to handle in terms of computation time.

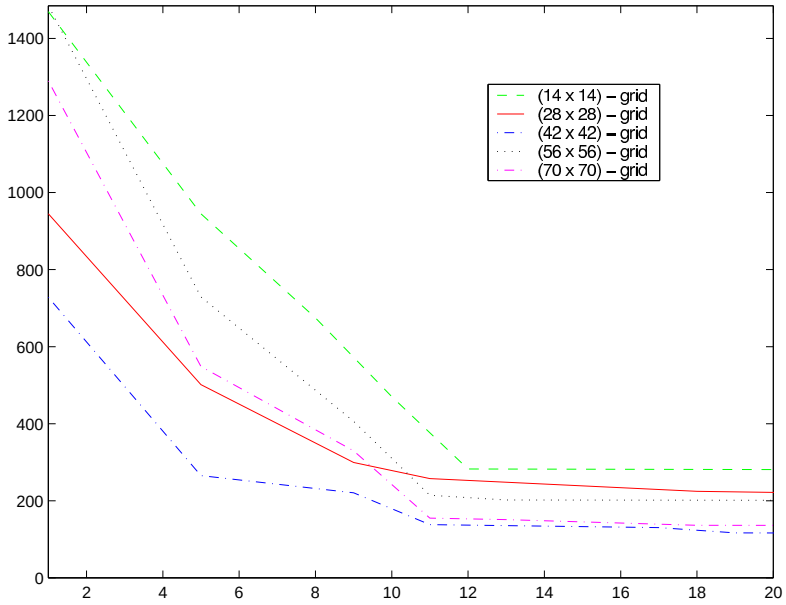
Table 4.3.6: Function values at computed solutions (`IFFCO` and Nelder–Mead with 50 function evaluations) for different grid sizes.

grid size	14	28	42	56	70
$J_{*,IFFCO}$	208.355	167.504	107.111	150.041	121.036
$J_{*,NM}$	232.985	166.569	108.060	162.423	122.645

The computed solutions differ between the two algorithms and, for each algorithm, for different grids. Moreover, when starting from different positions, it must be expected that the Nelder–Mead algorithm detects different local minima. This can be observed in Figure 4.3.7. The routine wanders off into very different directions for different starting values. The Implicit Filtering routine, in contrast, stops at points that are relatively close to each other compared to the Nelder–Mead outcome. See

Figure 4.3.6. This makes a conceptional difference between the Nelder–Mead and Implicit Filtering routines apparent: Implicit Filtering is designed to "filter" noise and, by computing finite differences with large increments, to avoid local minima. The Nelder–Mead may use large increments as well, but is not designed to get a "global" picture of the problem. Compare Sections 3.3 and 3.4.

Figure 4.3.5: Development in Nelder–Mead iterations. Function values versus number of function evaluations for different grids.



In reference to the very different directions that the Nelder–Mead algorithm embarks on with different starting points see also the optimization landscape of our application in Figure 4.3.16. This landscape was obtained with computations on the IBM SP2. Computing activity was partially supported by an allocation from the North Carolina Supercomputing Center. To obtain such a landscape, all but two of the optimization variables are fixed. The function values are then plotted against the two varying values. The landscape clearly shows that distinct local minima with large regions of attraction exist. These can capture the Nelder–Mead iterates. This did not happen with `IFFCO` iterates in our experiments.

In Figure 4.3.8, the flow profile of the application is shown for the uncontrolled case. The figure shows the center part only because it is of main interest. The upper and lower part of the domain are cut away. The infiltration well in the eastern part of the domain is easily detectable from the flow profile, as are the extraction wells in the western part. Note that considerable inflow to the extraction wells occurs from the river in the northern part of the domain, and note also that inflow from the infiltration well takes place both to northern and southern direction. The flow profile is changed when the computed control is exercised. In Figure 4.3.9, the flow profile is given for

Figure 4.3.6: Development in `IFFCO` iterations. Function values versus number of function evaluations for different starting points.

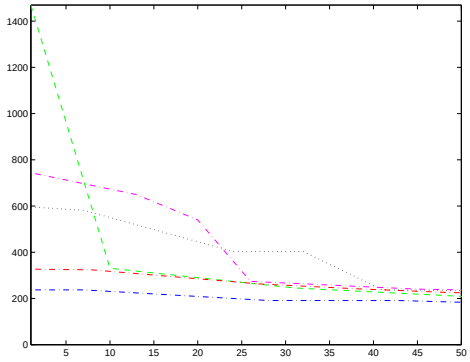
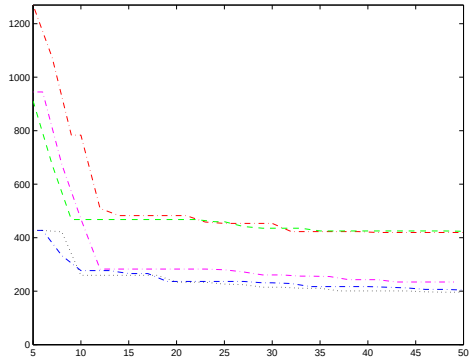


Figure 4.3.7: Development in Nelder–Mead iterations. Function values versus number of function evaluations for different starting points.



the control computed by `IFFCO` on the (28×28)–grid. While the flow profile is practically unchanged in the outermost parts of the domain, the influence of the barrier wells is clearly visible in the central part of the considered domain. Direct flow from the infiltration well to the extraction wells is impeded. The heated water is in large parts attracted by the barrier wells. Here, the three upper wells play the most important role, most notably the well that is located on the same horizontal line as the infiltration well. This is obvious when considering the location of the wells, compare Figure 4.3.1, and the flow profile for the uncontrolled case. This fact is clearly reflected in the solutions that the optimization routines furnish.

Figure 4.3.8: Flow profile in uncontrolled case for (28×28)–grid.

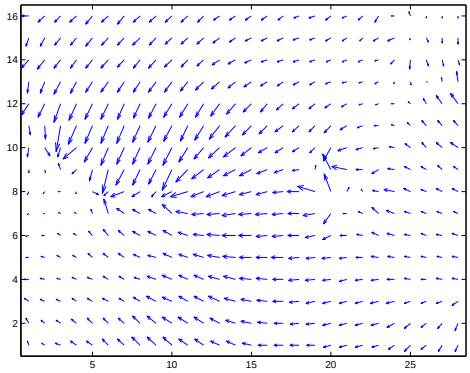
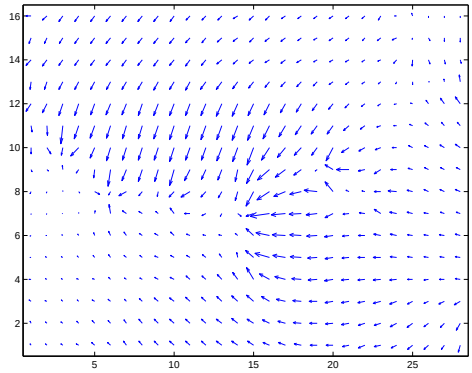


Figure 4.3.9: Flow profile in controlled case (`IFFCO`) for (28×28)–grid.



In Figures 4.3.10 and 4.3.11, the development of the plume is depicted at times $t = 5a$ and $t_1 = 20a$. These computations are done for the uncontrolled case on the grid with 42×42 nodes. It can be seen that the heat plume extends well beyond the extraction wells after an infiltration time of twenty years when no control is performed.

For the control computed by `IFFCO` for this grid, the controlled case is shown in Figures 4.3.12 and 4.3.13. After five years, the plume is visibly smaller than in the uncontrolled case, and not only is its overall extension smaller than before, but also its core. The same holds for the time span of twenty years.

Figure 4.3.10: Temperature plume after 5 years in uncontrolled case on (42×42)–grid.

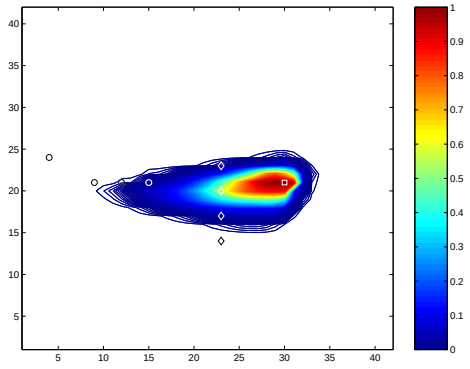


Figure 4.3.11: Temperature plume after 20 years in uncontrolled case on (42×42)–grid.

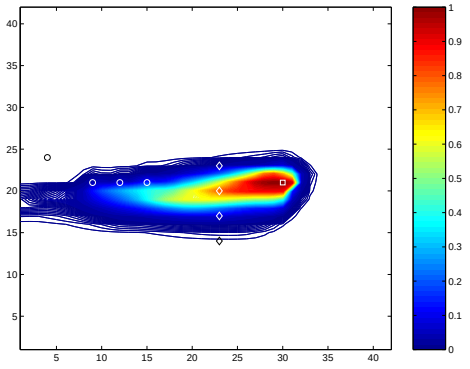


Figure 4.3.12: Temperature plume after 5 years with control computed by `IFFCO` for (42×42)–grid.

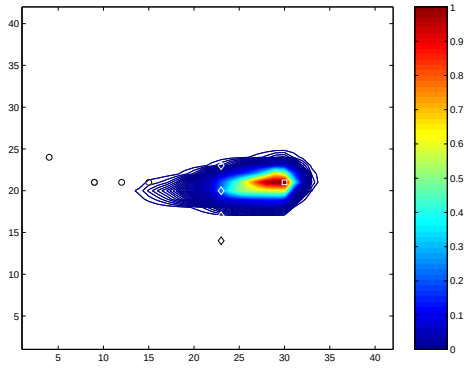
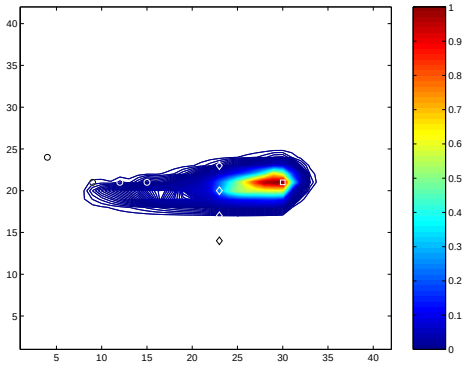


Figure 4.3.13: Temperature plume after 20 years with control computed by `IFFCO` for (42×42)–grid.



We now turn to the results for the equidistant well coordinates, where the absolute distances between the wells do not vary for different grids. The first observation is that the function values $J(0)$ at the origin do no longer exhibit the large jumps that are exhibited with the original well coordinates. Compare Tables 4.3.2 and 4.3.7. With the exception of the coarsest grid, the value $J(0)$ is consistently growing with mesh refinement.

The optimization with `IFFCO` also exhibits consistent results. The optimal function values computed for the different grids decrease with mesh refinement. This can be observed for both optimization routines, i.e., for the Nelder–Mead algorithm as well as for `IFFCO`. See Figures 4.3.14, 4.3.15, and Table 4.3.7. Despite this consistency

Table 4.3.7: Function values at origin and at solution computed by IFFCO and Nelder–Mead after 50 function evaluations for different grid sizes with equidistant well coordinates.

grid size	14	28	42	56	70
$J(0)$	1469.594	1353.404	1401.406	1430.984	1433.351
$J_{*,NM}$	232.985	201.948	174.289	150.561	138.461
$J_{*,IFFCO}$	208.355	195.360	172.446	150.541	140.298

with mesh refinement, the stress rates computed by the two routines are not considerably closer to each other than in the case where the well locations vary. An interesting result is furnished both by IFFCO and the Nelder–Mead algorithm for the (70×70)–grid, where both suggest to extract heated water from the northern wells (0.0136 and 0.0293 (NM), 0.0095 and 0.0289 (IFFCO)), and to infiltrate cold water into both southern barrier wells (0.00322 and 0.00429 (NM), 0.00182 and 0.00434 (IFFCO)). These values are computed after about 50 function evaluations. In all other computations with the upwind routine, both with the original and the equidistant well coordinates, it is consistently suggested by the optimization routines to extract water from three wells and to infiltrate a considerably smaller amount only at the southernmost well.

Figure 4.3.14: Development in IFFCO iterations over 50 function evaluations. Function values on logarithmic scale versus number of function evaluations for equidistant wells on different grids.

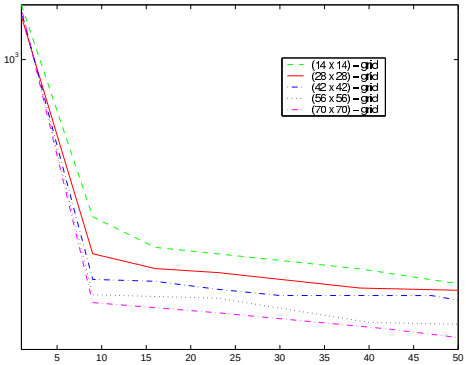
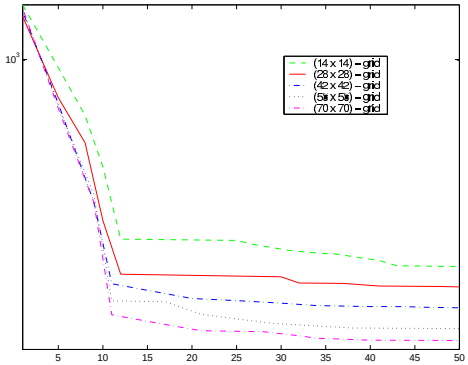


Figure 4.3.15: Development in NM iterations over 50 function evaluations. Function values on logarithmic scale versus number of function evaluations for equidistant wells on different grids.



The last set of results that we mention is furnished by employing the method of characteristics to solve the transport equation. Working with the original well coordinates, we observe again that the cost function value at the origin changes considerably with mesh refinement. These changes are even larger in absolute value than for the upwind scheme. See Table 4.3.8. Also, the function values for the computed solutions

differ widely and do not show a consistent development. Another observation is that `IFFCO`, which furnished in general better results than the Nelder–Mead algorithm, is outperformed by the latter when the method of characteristics is used. This is probably due to very poor gradient information. While derivative information is always poor in this application, it is considerably less reliable when the method of characteristics is used than when the upwind scheme is employed. In those cases where the Nelder–Mead algorithm is significantly better than `IFFCO`, gradient information can almost not be used. `IFFCO` is then trapped at one of those points that are used for the initial gradient approximation, typically in a corner of the hyperbox that is admitted for the optimization with `IFFCO`. In this case, and only in this case, one (or more) of the decision variables takes on the extreme value admitted. This numerical scheme did not furnish usable results for our application.

Table 4.3.8: Function values at origin and at solution computed by `IFFCO` and Nelder–Mead after 50 function evaluations for different grid sizes with method of characteristics (MOC).

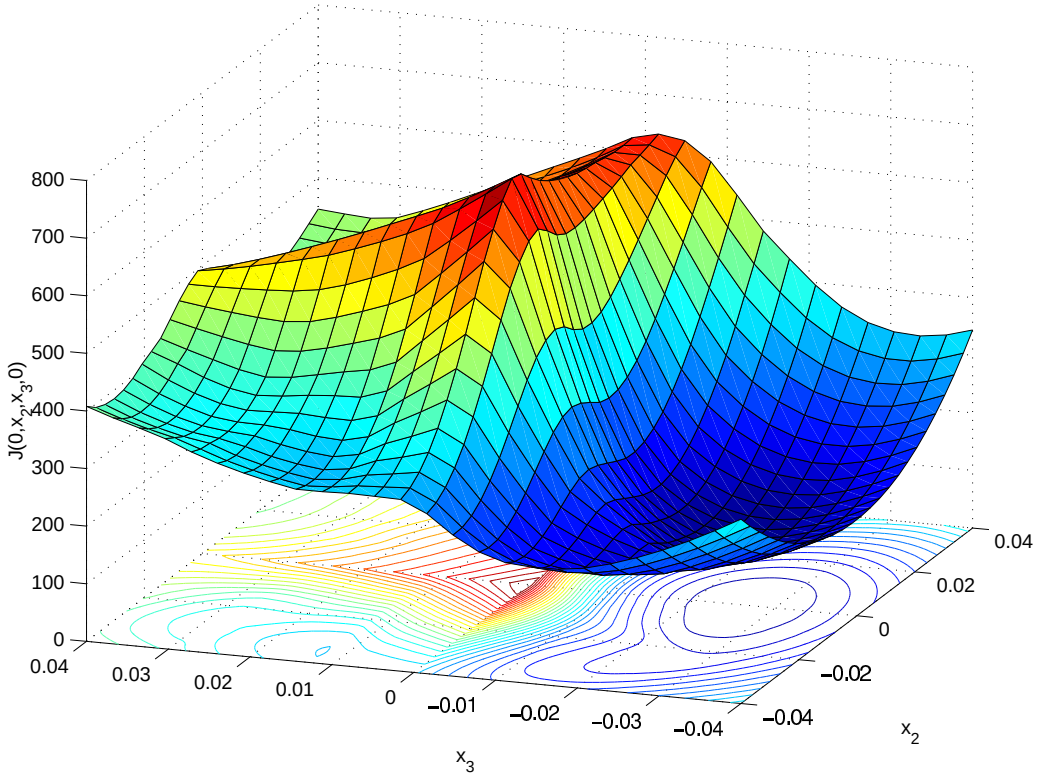
grid size	14	28	42	56	70
$J(0)_{upwind}$	1469.594	945.038	728.026	1484.668	1289.737
change		-35.69%	$+22.96\%$	$+103.93\%$	-13.13%
$J(0)_{MOC}$	563.361	833.926	482.313	1206.019	998.490
change		$+48.03\%$	-42.16%	$+150.05\%$	-17.21%
$J_{*,NM}$	25.131	45.772	44.952	61.513	51.703
$J_{*,IFFCO}$	47.164	50.507	164.893	81.278	53.523

In conclusion, specialized optimization routines are useful tools in the given setting. The optimization landscape with its nondifferentiable rim shows that optimization algorithms using first– and second–order information under the assumption of smoothness encounter difficulties. It is highly advisable in this setting to use methods that either take special care in the finite–differencing or forego derivative information.

The Nelder–Mead algorithm does the latter. It achieves good progress with few function evaluations. But the algorithm is sensitive to the starting point and can easily be trapped in one of the local minima which have large regions of attraction. The conception of the Nelder–Mead algorithm does not invite parallelization. Parallelization, however, holds promise for Implicit Filtering.

`IFFCO` gave good results for the application. With a low number of function evaluations, considerable progress was achieved. In all constellations which were considered in the numerical experiments, twenty function evaluations allowed good progress. The expense of fifty function evaluations might still be justifiable. This is true for any expensive application that exhibits the iteration history seen in this section, and even more in our application, where inaccuracy introduced through modeling, data, and software issues will not permit to capture the exact minimum.

Figure 4.3.16: Optimization landscape for the application on the (42×42)–grid. Function values for x_2, x_3 varying with x_1, x_4 fixed at origin.



Chapter 5

An Indefinite Preconditioner for KKT Systems

We now turn to our second example problem, motivated by an application in ground water modeling like the first. Likewise, it is formulated as an optimal control problem governed by a partial differential equation. Its discretization is a quadratic programming problem with linear constraints. Quadratic problems with linear constraints are frequent in optimization. In Chapter 1 we have seen that such problems have to be solved for example as subproblems within a sequential quadratic programming approach. The system matrices arising in the discretization of partial differential equations are generally sparse and tend to large dimensions so that generally iterative solvers are used. It is crucial for overall performance that the iterative solution is done efficiently, and so preconditioners as convergence–enhancing tools come into play.

In the present chapter we analyze an indefinite preconditioner which can be advantageously used within the iterative solution of large and sparse KKT systems. A number of other preconditioners for symmetric indefinite systems is reviewed in the following Chapter 6. Our application is derived in Chapter 7. The preconditioner of the present chapter is, together with related preconditioners, tested on this example problem of ground water hydraulic management. The linear systems under consideration are solved iteratively with the Krylov subspace methods MINRES [66] and GMRES [70]. Numerical results for the elliptic model problem are shown in Section 7.3. A condensed analysis of the indefinite preconditioner can be found in [7].

We start the present chapter with an introduction to preconditioning and a review of some Krylov subspace methods, Section 5.1. Convergence analysis is also addressed for those Krylov subspace methods that are employed in the numerical tests. The advocated preconditioner is presented in Section 5.2. There, some observations concerning the KKT system are related. Basic findings about the preconditioner are stated and its cost is discussed. The effects of the preconditioner on the eigenvalue distribution of the system, and, as a consequence, on the performance of iterative solvers, are studied in Sections 5.3 and 5.4.

5.1 Review of Preconditioning and of some Krylov Subspace Methods

In this part of the present work we are concerned with preconditioners for linear systems that frequently arise in optimization. The so called KKT matrices have been introduced in Chapter 1. The matrices under consideration exhibit the special structure

$$K = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix} \tag{5.1.1}$$

that we already encountered in the derivation of sequential quadratic programming methods. In each step of such methods, a linear system

$$K\,x = r \tag{5.1.2}$$

with $K \in I\!R^{N\times N}$ symmetric indefinite has to be solved, and it is crucial for overall performance to do that efficiently.

The system matrices K are generally sparse and tend to large dimensions. The use of iterative solvers is usually advocated under these circumstances. Iterative methods are advantageous in that they do not require to assemble the entire system K, but, instead, only a matrix–vector multiplication $K\,v$ per iteration. Common choices are so called Krylov subspace methods. These methods generate iterates

$$x_j \in \mathcal{K}_j(K, r_0) \equiv span\left\{K^l\, r_0\right\}_{l=0}^{j-1},$$

where $\mathcal{K}_j(K, r_0)$ denotes the Krylov subspace of order j generated by K and the initial residual r_0. A number of these methods is addressed in this section, specifically MINRES [66] and GMRES [70]. The methods differ e.g. in their applicability. While MINRES is reserved for symmetric, possibly indefinite, systems like e.g. (5.1.1), GMRES accepts general linear systems.

Several Krylov subspace methods, most prominently the conjugate gradient, minimum residual, and generalized minimum residual methods, are N–step procedures in exact arithmetic. This means that they stop with the exact solution of (5.1.2) after at most N iterations. But order of N iterations are in general judged a too high computational expense. It is customary to regard Krylov subspace methods as genuinely iterative procedures with termination based upon an iteration maximum and the residual norm $||r_j||$. Evaluation of such methods invariably focuses on how fast the iterates converge. Convergence analysis for Krylov subspace methods is well understood for normal matrices, compare e.g. [43]. For nonnormal matrices, it is an area of active research, see e.g. [14], [30], [43], [62]. Convergence analysis is addressed below in this section as well. Usually, convergence analysis is done for the "worst case", i.e., assertions usually hold for all right hand sides r in (5.1.2) and all starting guesses x_0. Thus, known bounds on the residual are not necessarily descriptive of actual convergence

behavior even in the normal, well–understood, case. But they can provide intuition of what constitutes 'good' and 'bad' eigenvalue distribution. As such, convergence analysis is the basis for the construction of preconditioners. Preconditioners are often that convergence–enhancing tool that renders iterative methods efficient.

Review of Preconditioning

The general issue in preconditioning is to construct a system that is equivalent to the original system and that is easier to solve. The ideal preconditioner can be — this is not a must — thought of to approximate the inverse of the original system matrix. Use of the exact inverse is in general prohibitively expensive. Clearly, cost and effectiveness of a preconditioner have to be weighted against each other.

When symmetric positive definite preconditioning is applied for symmetric K,

$$P^{-1}KP^{-T}\,P^{T}x = P^{-1}r, \tag{5.1.3}$$

the symmetry of the original K is maintained in the preconditioned matrix $P^{-1}KP^{-T}$. Of course, P is taken to be nonsingular, so that (5.1.2) and (5.1.3) are equivalent. Straightforward incorporation of preconditioning into Krylov subspace methods would replace the matrix–vector multiplication $K\,v$ by $P^{-1}KP^{-T}w$ so that a solve with P^T, the multiplication with K, and a solve with P would be performed. When symmetric positive definite preconditioning is incorporated into Krylov subspace methods, these can be rewritten such that in each iteration one solve with M must be done where $M = PP^T$. For a detailed derivation of how to use M instead of its factors within MINRES see e.g. [3]. It can be advantageous that the factors P, P^T need not be explicitly known. The matrix M as well as P may be referred to as the preconditioner.

Much of the research in preconditioning for indefinite linear systems is dedicated to symmetric positive definite preconditioning, see e.g. [5], [34], [69], [74], [75]. These works are reviewed in Section 6.1. For symmetric, but highly indefinite systems like the KKT system (5.1.1), "this restriction on a preconditioner is rather unnatural" [31]. Instead, one would like to be able to admit a preconditioner that is indefinite like K is. This is done e.g. in [31], [54]. In that instance, consider the generally preconditioned system

$$P_L^{-1}KP_R^{-T}\,P_R^{T}x = P_L^{-1}\,r \tag{5.1.4}$$

which is for nonsingular P_L, P_R equivalent to (5.1.2). The preconditioned system matrix $P_L^{-1}KP_R^{-T}$ is for $P_L \neq P_R$ in general nonsymmetric. In this case, methods for nonsymmetric matrices are needed, e.g. GMRES. Special cases of (5.1.4) are left– and right–sided preconditioning, obtained by setting $P_R = I$ and $P_L = I$, respectively.

Much of the general research in preconditioning is dedicated to linear algebra issues and to specific applications. For an overview see e.g. [43]. Classical preconditioning methods are for instance incomplete factorization methods. For a specific example in ground water modeling, the effects of several preconditioners derived from incomplete decompositions are studied in [56]. Another classical idea is to employ

splittings of the system matrix. We briefly review several important preconditioners of that type in Section 6.1. This is not our focus, however.

A main feature of our approach is that we are interested in constructing block preconditioners that maintain the structure of the original system and allow to exploit that structure to computational advantage. In general, the effectiveness of a preconditioner depends on the particular system the preconditioner is used for. There are some preconditioners, for example the preconditioner constructed by an incomplete LU–decomposition, or a truncated series approach, e.g. [63], that are designed without taking into account the structure of the matrix. Their usage can be highly effective, but this is not necessarily the case. Sometimes such preconditioners cannot be used because of the sheer size of the considered systems. If the matrices are very large, usually only the blocks are computed, and they are not really assembled into an entire system matrix. It is possible even that the blocks themselves are not explicitly known, and that only their action on vectors can be computed. These instances may exclude the use of general preconditioners. Another argument in favor of block preconditioners is that matrices that arise in the solution of optimal control problems are highly structured. For one, they exhibit the special (3×3)–structure of (5.1.1), and generally their blocks have special features, too. Matrices A arising from the discretization of partial differential equations have special structure, and their features have been studied by numerous authors. Therefore, we do not attempt to use preconditioners of such general design, but focus on preconditioners that take advantage of the special form and features.

The preconditioner proposed in this work is composed of blocks and, for suitable P_A and P_H, given as

$$\tilde{K} = \begin{pmatrix} 0 & 0 & P_A^T \\ 0 & P_H & B^T \\ P_A & B & 0 \end{pmatrix}.$$

In the ideal case, P_A and P_H are inverses for the submatrices of the original system or for products of submatrices. But it cannot in general be assumed that exact solves are affordable. We also account for the general case where the preconditioner for the full system is composed of preconditioners for the submatrices. Such block preconditioners, composed of preconditioners for submatrices, can generally be adapted to specific applications by incorporating known effective preconditioners for subsystems of the system matrix. This line of thinking can also be found in the works [3], [5], [34], [49], [69]. See Chapter 6.

Brief Review of Some Krylov Subspace Methods

As a basis for the topics that we focus on in Sections 5.3 and 5.4, we now review some Krylov subspace methods. In this section we do not use the notation (5.1.2), but the notation that is generally used for linear systems. We are interested in solving

$$A\,x = b \tag{5.1.5}$$

with $A \in \mathbb{R}^{n\times n}$ nonsingular. We also use $\|x\| = \|x\|_2 = \sqrt{x^T x}$ and $\langle x, y\rangle = x^T y$.

Iterative methods are especially suited for the solution of large sparse linear systems (5.1.5) because they do not require to assemble the entire system A, but, instead, only a matrix–vector multiplication $A\,v$ per iteration. Basic iterative methods are the Jacobi and Gauss–Seidel iterations, e.g. [39, Ch. 10.1]. A difficulty associated with these methods is their dependence on parameters that are sometimes hard to choose properly. This difficulty is overcome with the well–known conjugate gradient method, see e.g. [39, Ch. 10.2], which dates back to 1952. Numerous so called Krylov subspace methods have since evolved which can be viewed as generalizations of the conjugate gradient method.

By construction of the method, the conjugate gradient method really solves

$$A\,x = r_0, \tag{5.1.6}$$

where r_0 is the initial residual, $r_0 = b - A\,x_0$. Since we do not want to restrict ourselves to the starting vector $x_0 = 0$, we apply the conjugate gradient method to (5.1.5) with b replaced by r_0. The exact solution to (5.1.6) we denote by x_*. If no restarts are performed, the conjugate gradient and all other Krylov subspace methods generate iterates

$$x_j \in \mathcal{K}_j(A, r_0) \equiv span\left\{A^l\, r_0\right\}_{l=0}^{j-1},$$

the Krylov subspace of order j generated by A and the initial residual r_0. While the conjugate gradient method is well–defined only for symmetric positive definite matrices, MINRES and SYMMLQ [66] have been derived for symmetric, possibly indefinite, systems, and GMRES [70] solves general linear systems.

The derivation of the conjugate gradient method, e.g. [39, Ch. 9.3], can be based on the fact that for positive definite matrices A the problem to solve (5.1.6) is equivalent to the minimization problem

$$\min_{x\in\mathbb{R}^n} F(x) = \tfrac{1}{2}\langle x, Ax\rangle - \langle x, r_0\rangle\,. \tag{5.1.7}$$

The problem (5.1.7) has, in the positive definite case, the unique solution $x = A^{-1}r_0$.

In each step the conjugate gradient method computes the iterate x_j in the Krylov subspace $\mathcal{K}_j(A, r_0)$ which minimizes F over $\mathcal{K}_j(A, r_0)$, i.e., the iterate x_j solves

$$\min_{x\in\mathcal{K}_j(A,r_0)} F(x)\,. \tag{5.1.8}$$

This is equivalent to solving

$$\langle Ax_j - r_0, v\rangle = 0 \quad \forall v \in \mathcal{K}_j(A, r_0). \tag{5.1.9}$$

The conjugate gradient method thus minimizes the error $e = x_* - x$ in A–norm $\|.\|_A$ over the current Krylov space,

$$F(x_j) = \min_{x\in\mathcal{K}_j(A,r_0)} F(x) = \min_{x\in\mathcal{K}_j(A,r_0)} \tfrac{1}{2}\langle x_* - x, A(x_* - x)\rangle\,. \tag{5.1.10}$$

The A–norm is defined only for symmetric positive definite matrices. Also, if A is not positive definite, then (5.1.6) and (5.1.7) are not equivalent. In fact, (5.1.7) does not have a solution if A has negative eigenvalues, and it may not have a solution if A is only positive semidefinite. Even though in this case the foregoing derivation is not applicable, one can try to extend the conjugate gradient method by trying to compute the iterates x_j as a solution of (5.1.9). This leads to SYMMLQ. This method tries to compute the so called *Galerkin approximation* to the solution x_* of (5.1.5) over the Krylov subspace $\mathcal{K}_j(A, r_0)$ even for indefinite A, i.e., the solution x_j to (5.1.9). Unfortunately, (5.1.9) need not have a solution for indefinite A. This is why SYMMLQ uses a closely related iterate that coincides with the Galerkin approximation when an invariant subspace is encountered. We do not discuss implementational issues here, for that we refer the reader to the original works and others. A starting point is e.g. [43]. If A is positive definite, then (5.1.9) always has a unique solution. In that instance, SYMMLQ is equivalent to the conjugate gradient method.

An alternative to SYMMLQ is MINRES, this based on the *minimum residual approximation* to the exact solution. In each step, this method computes the unique solution $x_j \in \mathcal{K}_j(A, r_0)$ to

$$\min_{x \in \mathcal{K}_j(A, r_0)} \|r_0 - Ax\|. \tag{5.1.11}$$

MINRES and SYMMLQ are tailored to the symmetric, possibly indefinite, case. Their implementation relies on the Lanczos Tridiagonalization, see e.g. [39, Ch. 9.1], that effectively allows to compute the new iterate via a three–term recurrence. Its extension to the general, nonsymmetric case is the Arnoldi process, see e.g. [39, Ch. 9.3]. The method whose iterates satisfy the optimality condition (5.1.11) in each step $j = 0, 1, \ldots$ for general A is called GMRES.

Satisfying (5.1.11) or generating iterates with smallest possible error in a norm that is independent of the starting guess is for general A impossible via a simple three–term recurrence, [43, Ch. 6]. The recurrences of GMRES entail an increase in storage requirement throughout the iterations. If storage is exhausted, a remedy can be to restart GMRES. Also, additional methods have been developed for general A to overcome the long recurrences of GMRES, e.g. QMR [33]. These methods have the possibility of failure which can be alleviated by allowing for growth in work and storage within the iteration. For a discussion of numerical properties of methods for general A see e.g. [62].

In this work, see Section 7.3, MINRES and GMRES are used. For these we now discuss convergence.

Convergence Analysis

As Krylov subspace methods whose iterates satisfy the optimality condition (5.1.11), MINRES and GMRES are n–step procedures. Despite the finite termination property, rounding errors may lead to a loss of orthogonality among theoretically orthogonal vectors, and finite termination is not mathematically guaranteed. Moreover, when

these iterative solvers are applied, n is usually so large that $\mathcal{O}(n)$ iterations represent an unacceptable amount of work. As a consequence, it is customary to regard the methods as genuinely iterative techniques with termination based upon an iteration maximum and the residual norm. With this point of view, the rate of convergence becomes important. Before investigating more specialized convergence results, we state the result on the finite termination of these Krylov methods.

The convergence of Krylov subspace methods with optimal iterates is related to polynomial approximation problems. This relation is based on the special representation of Krylov subspaces as

$$\mathcal{K}_j(A, v) = \{p(A)v \,|\, p \in \Pi_{j-1}\}, \tag{5.1.12}$$

where Π_k denotes the space of all polynomials of degree k or less. The conjugate gradient method and its generalizations iterate on Krylov subspaces of increasing dimension that eventually are invariant subspaces of the system matrix A. A subspace $\mathcal{X}$ of $\mathbb{R}^n$ is an invariant subspace for A if and only if $AX = XB$ for some $B \in \mathbb{R}^{m\times m}$, where the m columns of $X \in \mathbb{R}^{n\times m}$ span $\mathcal{X}$. This means that the action of A on the m–dimensional subspace $\mathcal{X}$ is completely determined by B. If $r_0 \in \mathcal{X}$, then $r_0 = Xc$ holds for some $c \in \mathbb{R}^n$, so that $Ax = r_0$ can be solved by solving $By = c$ for $y \in \mathbb{R}^m$, and setting the solution to $x = Xy$. Thus the problem of dimension $n \times n$ is reduced to an $(m \times m)$–system which can result in quite a computational advantage. So the plan is to find the smallest invariant subspace containing r_0.

One of the features that the conjugate gradient, minimum residual, and generalized minimum residual methods have in common is their finite convergence. The finite termination of Krylov minimum residual methods follows from the optimality condition (5.1.11) and the fact that after at most n iterations the characteristic polynomial of A (of degree at most n) can furnish the zero residual. SYMMLQ computes the Galerkin approximation (5.1.9) and can be shown to also furnish a zero residual after at most n steps. This follows by observing that Krylov subspaces of maximal dimension are invariant subspaces for the generating matrix, i.e., that the implication

$$A^k v \in \mathcal{K}_k(A, v) \quad \Rightarrow \quad A^l v \in \mathcal{K}_k(A, v) \quad \forall l \geq k$$

holds. The proofs are elementary and can be found e.g. in [3].

Convergence Results for MINRES

Our interest in this section are symmetric indefinite system matrices for which we use the following notation. If $A \in \mathbb{R}^{n\times n}$ is nonsingular symmetric indefinite, then all eigenvalues of A are contained in two intervals on the real line, one on the positive, one on the negative part. The spectrum is denoted by $\Lambda(A) = \Lambda$. We set

$$\bar{\lambda} = \max_{\lambda\in\Lambda} |\lambda|\,, \quad \underline{\lambda} = \min_{\lambda\in\Lambda} |\lambda|\,, \quad \kappa = \frac{\bar{\lambda}}{\underline{\lambda}}\,, \tag{5.1.13}$$

and κ denotes the spectral condition number of A. In addition let λ_i denote the i–th largest eigenvalue, i.e.,

$$\lambda_1 \geq \ldots \geq \lambda_l > 0 > \lambda_{l+1} \geq \ldots \geq \lambda_n.$$

The minimization property (5.1.11) of MINRES and the representation (5.1.12) imply the following result, see e.g. [2], [79].

Theorem 5.1.1 *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ denote its spectrum. If x_j are minimum residual approximations to the solution of $Ax = r_0$ on a Krylov sequence, then the following estimate holds for the corresponding residuals,*

$$\|r_j\| \leq \min_{p\in\Pi_j^1} \max_{i=1,\ldots,n} |p(\lambda_i)|\, \|r_0\|, \tag{5.1.14}$$

where Π_j^1 stands for the space of all polynomials of degree j or less which are normalized to 1 at the origin.

The proof relies on the fact that for symmetric matrices A there exists a similarity transformation such that $A = VDV^T$ with V orthonormal and with D a diagonal matrix that contains the eigenvalues of A. Since V is orthonormal, and since $p(A) = Vp(D)V^T$ holds for every polynomial p,

$$\begin{aligned}
\|r_j\| &= \|r_0 - Ax_j\| \\
&= \min_{p\in\Pi_j^1} \|\,p(A)\,r_0\,\| \\
&= \min_{p\in\Pi_j^1} \|\,p(D)V^T r_0\| \\
&\leq \min_{p\in\Pi_j^1} \max_{i=1,\ldots,n} |p(\lambda_i)|\, \|r_0\|.
\end{aligned}$$

As a direct implication, it can be shown that if A has only l distinct eigenvalues, A symmetric indefinite, then MINRES terminates in l steps. This follows easily by properly normalizing the characteristic polynomial.

Theorem 5.1.2 *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular symmetric indefinite with l distinct eigenvalues. If $x_j \in \mathcal{K}_j(A, r_0)$ are minimum residual approximations of x_*, then $\|r_l\| = 0$.*

Theorem 5.1.2 shows that the iterative process stops after l steps if the system matrix has l distinct eigenvalues. If this number l is small compared to the dimension of the system, we have a large computational gain. This result on its own already motivates preconditioning to affect the eigenvalue distribution of the system matrix.

The convergence analysis of minimum residual approximations is closely related to the Chebyshev approximation problem, see e.g. [2]. It allows for instance the deduction of the following standard convergence estimate, e.g. [79].

Theorem 5.1.3 *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular symmetric indefinite. If $x_j \in \mathcal{K}_j(A, r_0)$ are minimum residual approximations of x_*, then the residuals $r_j = r_0 - A\,x_j$ obey*

$$\|r_j\| \leq 2\left(\frac{\kappa-1}{\kappa+1}\right)^{\lfloor j/2\rfloor}\|r_0\|,$$

where κ is the condition number of A. Here, $\lfloor j/2\rfloor$ denotes the largest integer less or equal to $j/2$.

If the intervals containing the eigenvalues of A are of equal size and if they have the same distance from the origin, and if the eigenvalues are equally distributed, then Theorem 5.1.3 gives a good description of the convergence behavior of MINRES. Distribution and clustering of the eigenvalues are important for the convergence of the method. If there are few well–separated clusters of eigenvalues, then the prediction will be pessimistic, and sharper results can be derived. If there are few negative eigenvalues, then the following result is of interest.

Theorem 5.1.4 *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular symmetric indefinite with eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_l > 0 > \lambda_{l+1} \geq \ldots \geq \lambda_n.$$

If x_j are minimum residual approximations of x_* on $\mathcal{K}_j(A, r_0)$, then

$$\|r_{j+n-l}\| \leq 2\left(\prod_{i=l+1}^{n}\frac{\lambda_1-\lambda_i}{|\lambda_i|}\right)\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{j}\|r_0\|$$

for $j \geq 0$, where $\kappa = \lambda_1/\lambda_l$.

The proof is performed via the construction of a specific polynomial that "singles out" the negative eigenvalues. This result is of interest if there are only few negative eigenvalues, so that the estimate can be established after a small number of iterations, and if the negative eigenvalues are not too small, because otherwise the factor $\prod(\lambda_1 - \lambda_i)/|\lambda_i|$ is large. This situation does not occur in our application. We have included this result because it gives the idea that isolating some eigenvalues can make sense in special situations in order to establish refined convergence results of this flavor. See e.g. [3, Th. 4.3.11]. In the following section we will see a refined analysis of this kind for the general case, Theorem 5.1.6.

Convergence Results for GMRES

We now turn to the general, not necessarily symmetric, case. The following paragraph can be easily reformulated to account for complex coefficient matrices. We restrict ourselves to the case with real entries. The eigenvalues of a real nonsingular matrix $A \in \mathbb{R}^{n\times n}$ can be considered to be ordered according to their absolute values,

$$|\lambda_1| \geq \ldots \geq |\lambda_n| > 0\,.$$

Bounds can be derived for normal matrices such that the residual norm of the GMRES iterates is completely determined by the eigenvalues of the matrix. The bounds are sharp in the sense that there always exists a right hand side r so that these bounds are attained, i.e., the bounds account for worst–case behavior. For nonnormal matrices, there seems at this time no known way to describe how good the approximation will be in terms of simple properties of the coefficient matrix. Compare e.g. [30]. The analysis of the convergence behavior of Krylov subspace methods is an area of active research, see e.g. [14], [30], [43], [58], [62].

While the bounds on the residual are as worst–case bounds not necessarily descriptive of actual convergence behavior even in the normal, well–understood, case, they can provide intuition of what constitutes good and bad eigenvalue distribution. As such, convergence analysis is the basis for the construction of preconditioners.

The property (5.1.11) is for general A the so called minimization property of the j–th GMRES iterate. As a consequence, compare (5.1.14) ,

$$||r_j||_2 = \min_{p\in\Pi_j^1} ||p(A)r_0||_2 \leq \min_{p\in\Pi_j^1} ||p(A)||\,||r_0||_2, \tag{5.1.15}$$

holds for general A. As before, Π_k^1 is the space of all polynomials of degree k or less that are normalized at the origin. The minimization property of the GMRES iterates and its implication in (5.1.15) can be used advantageously for convergence estimates. If A is diagonalizable, $A = VDV^{-1}$, (5.1.15) implies the bound

$$||r_j||_2 \leq \kappa(V) \min_{p\in\Pi_j^1} \max_{i=1,\ldots,n} ||p(\lambda_i)||\,||r_0||_2\,. \tag{5.1.16}$$

Here, $\kappa(V) = ||V||\cdot||V^{-1}||_2$ is the spectral condition number of the matrix V of eigenvectors of A. If A is normal, $\kappa(V) = 1$. It was mentioned above that the Chebyshev approximation problem can be exploited in the symmetric case. Likewise, if A is diagonalizable and if the eigenvalues are real, the minimization problem in (5.1.16) is an approximation problem on the real line. In fact, if A is normal with real eigenvalues, (5.1.16) reduces to the situation encountered in Theorem 5.1.1. The situation is considerably more difficult for nonsymmetric matrices as an approximation problem in the complex plane. Upper bounds for (5.1.16) have been derived for the diagonalizable and for the defective case. For references see [14].

In addition to the eigenvalue–based bounds there are two other prominent propositions on how to estimate GMRES convergence. See [30] for references. One is based on the field of values, $W(A) = \{x^TAx \mid x \in I\!R^n, ||x|| = 1\}$. The bound requires to solve an approximation problem on $W(A)$. The other proposition is based on the ϵ–pseudospectrum of a matrix, $\Lambda_\epsilon(A) = \{z \in \mathbb{C} \mid z \in \Lambda(A+E), ||E|| \leq \epsilon\}$. In this case, an approximation problem on $\Lambda_\epsilon(A)$ must be solved.

Embree [30] states that each of the three bounds can be accurate, but that each can be misleading also. All fail to accurately describe convergence if nonnormality is associated mainly with only part of the spectrum. This is the case for various constellations of the indefinite preconditioner as we see in Section 5.4. We have found that

in the ideal case a well–known result is descriptive of convergence behavior on the preconditioned system. This result gives an upper bound on the number of possible iterations. It is closely related to Theorem 5.1.2. In the original paper [70, Prop. 2], Saad and Schultz showed that in exact arithmetic GMRES requires at most d_{min} iterations, where d_{min} is the degree of the minimum polynomial of A. This means that the number of iterations is small if the spectrum of A consists of a small number of non–defective eigenvalues of high multiplicity. The assertion follows fast from (5.1.11) and the observation (5.1.15) with an appropriate standardization of the minimum polynomial.

Let λ_i be the distinct eigenvalues of A with multiplicity $m_i > 0$ ($i = 1, \ldots, k_d$) in the minimum polynomial. Then

$$d_{min} = \sum_{i=1}^{k_d} m_i$$

is the degree of the minimum polynomial of A, and the m_i are characterized as the smallest positive integer n such that

$$ker(A - \lambda_i I)^{n+1} = ker(A - \lambda_i I)^n.$$

This shows that the multiplicity m_i of the eigenvalue λ_i in the minimum polynomial is the geometric multiplicity of the eigenvalue λ_i, while its multiplicity in the characteristic polynomial is its algebraic multiplicity. The minimum polynomial of A is given by

$$\prod_{i=1}^{k_d} (A - \lambda_i I)^{m_i}.$$

The result by Saad and Schultz [70, Prop. 2] reads as follows.

Theorem 5.1.5 *Let d_{min} be the degree of the minimum polynomial of A. For any right hand side b and initial guess x_0, GMRES terminates in exact arithmetic the iteration on* (5.1.6) *in at most d_{min} iterations with the solution $x_{d_{min}} = x_*$.*

Theorem 5.1.5 states that in the given situation the Krylov subspace $\mathcal{K}_{d_{min}}$ already contains all necessary information. This is obvious from the following observation. If a polynomial p of degree d_{min} exists s.t. $p(A) = 0$, and if d_{min} is the minimum degree to furnish this zero, then the Krylov subspaces $\mathcal{K}_{d_{min}+1}$ and $\mathcal{K}_{d_{min}}$ coincide because linear dependency of the powers $A^l r_0$ is reached,

$$\mathcal{K}_{d_{min}+1} = span\left\{A^l r_0\right\}_{l=0}^{d_{min}} = span\left\{A^l r_0\right\}_{l=0}^{d_{min}-1} = \mathcal{K}_{d_{min}}.$$

If eigenvalues are clustered in intervals, convergence can be expected to be faster than in the case where they are spread out. By constructing polynomials that are related to the minimum polynomial, Campbell, Ipsen, Kelley, and Meyer set up a qualitative

model for the convergence behavior of GMRES in [14]. The model that is set up in [14] distinguishes between clusters of eigenvalues and outliers. It supposes that GMRES first processes the outliers. This may differ from actual residual reduction, which is why the model offers rather a qualitative than a quantitative description of actual GMRES convergence behavior. By accounting for outliers and clusters, this model can be of use in the case where a known eigenvalue distribution is disturbed.

Suppose that there exists a single cluster of $n-m$ eigenvalues, $m \leq n$, with center $z=1$ and radius ρ. The outliers are those m eigenvalues that are not associated with the cluster. Let d be the degree of the minimum polynomial associated with the outlying eigenvalues λ_j ($j=1,\ldots,m$). The model asserts that, once the outlying eigenvalues have been processed, residual reduction depends mainly on the cluster radius.

Theorem 5.1.6 (Prop. 4.1 in [14]) *Given $\rho > 0$, determine m, $0 \leq m \leq n$, s.t. $\lambda_{m+1},\ldots,\lambda_n$ are the clustered eigenvalues and the remaining m eigenvalues are the outliers, i.e.,*

$$\{\lambda_j\}_{j=m+1}^{n} \subset \{z : |z-1| < \rho\}\,, \quad \{\lambda_j\}_{j=1}^{m} \subset \{z : |z-1| > \rho\}\,.$$

Let $m_j > 0$ be the multiplicity of the outlier λ_j ($j=1,\ldots,m$) in the minimum polynomial of A, and let $d = \sum_{j=1}^{m} m_j$. Then for any right hand side b and initial guess x_0, the bound

$$\|r_{d+l}\| \leq c\,\rho^l\,\|r_0\|$$

holds with a constant c independent of l.

The assertion says that residual reduction is delayed in the model if there are many outliers or outliers with high multiplicity in the minimum polynomial. The number of additional iterations for processing the cluster depends mainly on the cluster radius ρ. Convergence requires that $\rho < 1$, this implying that the eigenvalue zero is always considered an outlier. The cluster radius ρ is interpreted as an asymptotic convergence factor by Campbell et al., and the constant c as an asymptotic error constant. This constant can be viewed as a product $c = \rho\,\delta^m\,c_0$, where c_0 reflects the nonnormality of A, while ρ stands for the cluster size as before, and δ for the (relative) distance of the outliers from the cluster,

$$\delta = \max_{|z-1|=\rho} \max_{1\leq j\leq m} \frac{|\lambda_j - z|}{|\lambda_j|}.$$

Thus, when A is normal, the choice $c \leq \delta^m$ is feasible, and when A is diagonalizable, the choice $c \leq \kappa(V)\,\delta^m$ is feasible, where $\kappa(V)$ is the spectral condition number of the matrix of eigenvectors of A.

5.2 The Preconditioner $\tilde{K}$

This part is concerned with an indefinite preconditioner for linear systems

$$K\,x = r \tag{5.2.1}$$

with a system matrix K of the specific block structure

$$K = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix}. \tag{5.2.2}$$

We have described in Chapter 1 the setting that we are interested in, the solution of discretized problems of optimal control by sequential quadratic programming methods. Each step of such a method requires the solution of a quadratic programming problem with linear constraints. See Equations (1.1.16), (1.1.17). The necessary and sufficient optimality conditions for this problem are given by the KKT conditions (1.1.14), involving a system K as given in (5.2.2). Since we are interested in optimal control problems that are governed by partial differential equations, the so called equation and adjoint solves must both be provided for in this approach. The matrix A and its transpose, accounting for equation and adjoint solve, are in this setting assumed to be the computationally most expensive parts of the system (5.2.2). In general, the systems (5.2.2) are large and sparse, this due to the fact that matrices arising from the discretization of partial differential equations tend to large dimensions and are highly structured.

For the blocks of such systems K that we are interested in we make a blanket assumption which is natural in the considered setting. This assumption, along with notation that is used throughout the remainder of this work, is given in Section 5.2.1.

The preconditioner that is proposed in this work is composed of blocks and, for suitable P_A and P_H, given by

$$\tilde{K} = \begin{pmatrix} 0 & 0 & P_A^T \\ 0 & P_H & B^T \\ P_A & B & 0 \end{pmatrix}.$$

The preconditioner mimics the structure of the KKT system (5.2.2), and, intuitively, is a good approximation to K for suitable P_A and P_H. At this point of the exposition we only require nonsingularity of the preconditioners P_A, P_H for the submatrices. All further requirements on P_A and P_H that are needed for special cases are framed accurately below.

The analysis of the preconditioner is done throughout Sections 5.2, 5.3, and 5.4. The eigenvalue distribution of the resulting system is addressed, and also the important question whether the application of $\tilde{K}$ is practicable. These two issues – effects and cost – in general represent opposing tasks. First facts about the preconditioner are stated in Section 5.2.2. Subsequently, the cost of the preconditioner is examined. This is related to implementational issues, see Section 5.2.3.

5.2.1 Preliminaries on the KKT System

This section is devoted to preliminary remarks concerning the KKT system K,

$$K = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix}, \tag{5.2.3}$$

that we are interested in. We also relate the blanket assumption on the blocks of K that is valid throughout the following treatment. We use the additional notation

$$C = (A \mid B), \quad L_{zz} = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} \\ L_{u\phi} & L_{uu} \end{pmatrix}, \quad W = \begin{pmatrix} -A^{-1}B \\ I_k \end{pmatrix}. \tag{5.2.4}$$

The matrix W is the null space representation for the constraints C which is canonical for problems of optimal control. Neither W nor $A^{-1}B$ are commonly computed or stored. With this notation, the reduced Hessian H of the underlying quadratic programming problem, compare Chapter 1, is readily seen to be

$$H = W^T L_{zz} W = B^T A^{-T} L_{\phi\phi} A^{-1} B - L_{u\phi} A^{-1} B - B^T A^{-T} L_{\phi u} + L_{uu}\,. \tag{5.2.5}$$

The question of nonsingularity of H is closely related to nonsingularity of the entire system K as we see below. At this point of the exposition we also recall the dimensions of the involved matrices,

$$A, L_{\phi\phi} \in I\!R^{m\times m},\ B, L_{\phi u} \in I\!R^{m\times k},\ L_{u\phi} \in I\!R^{k\times m},\ L_{uu} \in I\!R^{k\times k}. \tag{5.2.6}$$

Here, $m, k \in I\!N$, where typically $k \ll m$. We use $N = 2m + k$.

For the system K in (5.2.3) and its blocks we make the following assumption throughout. Compare Assumption 1.1.2.

Assumption 5.2.1 *Let for the system K in (5.2.3) and its submatrices be valid:*

1. *K is nonsingular.*
2. *K is symmetric.*
3. *A is nonsingular.*

As pointed out in Chapter 1, the second item of Assumption 5.2.1 is an immediate consequence of differentiability requirements on the underlying functions. If these are twice continuously differentiable, $L_{\phi\phi}$ and L_{uu} are symmetric and $L_{\phi u}^T = L_{u\phi}$. The third item states solvability of the state equation. If A is nonsingular, the matrix $C = (A \mid B)$ of constraints has full rank. The first item of Assumption 5.2.1, provided that C has full rank, is equivalent to requiring nonsingularity of the reduced Hessian H in (5.2.5). See Lemma 5.2.2 and [35, Cor. 3.1].

Note that we do not set the requirement corresponding to the second item of Assumption 1.1.1 throughout our treatment. The vast majority of our results need not require that L_{zz} is symmetric positive definite on the null space of the constraints C. This would be the symmetric positive definiteness of H. Neither do we assume that L_{zz} be invertible and that consequently the Schur complement S of K exist, $S = -CL_{zz}^{-1}C^T$. Even less we assume that the upper left block L_{zz} be symmetric positive definite throughout. If this can be assumed, however, it is easy to see that the system K is indefinite with $m+k$ positive eigenvalues and m negative eigenvalues,

$$\begin{pmatrix} L_{zz} & C^T \\ C & 0 \end{pmatrix} = \begin{pmatrix} L_{zz} & 0 \\ C & I \end{pmatrix} \begin{pmatrix} L_{zz}^{-1} & 0 \\ 0 & -CL_{zz}^{-1}C^T \end{pmatrix} \begin{pmatrix} L_{zz} & C^T \\ 0 & I \end{pmatrix}.$$

To find out about the eigenvalue distribution of K, assuming only full rank of the constraints, we apply congruence transformations. We see that the nonsingularity of K is in this case equivalent to nonsingularity of the reduced Hessian H. From

$$\begin{pmatrix} I & 0 & 0 \\ -(A^{-1}B)^T & I & 0 \\ 0 & 0 & A^{-1} \end{pmatrix} \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix} \begin{pmatrix} I & -A^{-1}B & 0 \\ 0 & I & 0 \\ 0 & 0 & A^{-T} \end{pmatrix}$$

$$= \begin{pmatrix} L_{\phi\phi} & (L_{\phi\phi} \,|\, L_{u\phi})\, W & I \\ W^T \binom{L_{\phi\phi}}{L_{\phi u}} & H & 0 \\ I & 0 & 0 \end{pmatrix} \tag{5.2.7}$$

one can see immediately the following statement.

Lemma 5.2.2 *Let A be nonsingular. The system K in (5.2.3), with dimensions as in (5.2.6), is invertible if and only if the reduced Hessian H in (5.2.5) is invertible.*

Additional transformations give information about the inertia of the KKT system. Compare [35, Th. 3.1]. Premultiplying (5.2.7) by

$$\begin{pmatrix} I_m & 0 & 0 \\ 0 & 0 & I_k \\ -\frac{1}{2}L_{\phi\phi} & I_m & -(L_{\phi\phi} \,|\, L_{u\phi})\, W \end{pmatrix}$$

and postmultiplying with the transpose, we obtain that K can be transformed into

$$\begin{pmatrix} 0 & I & 0 \\ I & 0 & 0 \\ 0 & 0 & H \end{pmatrix}. \tag{5.2.8}$$

The $2m+k$ eigenvalues of (5.2.8) are the k eigenvalues of the reduced Hessian H and $+1$ and -1, these with multiplicity m. Hence, by Sylvester's law of inertia, the matrix K has $m+n_+$ positive eigenvalues, $m+n_-$ negative eigenvalues and n_0 eigenvalues equal to zero, where n_+, n_-, n_0 are the numbers of positive, negative and zero eigenvalues of H, respectively. Assuming that H is symmetric positive definite, K has $m+k$ positive eigenvalues and m negative eigenvalues.

5.2.2 Inverse and Iteration Matrix

In this section we start by introducing the inverse of $\tilde{K}$, composed of preconditioners for submatrices. Also, the preconditioned system is explicitly denoted. The preconditioner, we recall, is

$$\tilde{K} = \begin{pmatrix} 0 & 0 & P_A^T \\ 0 & P_H & B^T \\ P_A & B & 0 \end{pmatrix}. \tag{5.2.9}$$

Lemma 5.2.3 *Let Assumption* 5.2.1 *hold. Let P_A and P_H be nonsingular. The inverse of the preconditioner $\tilde{K}$ is given by*

$$\tilde{K}^{-1} = \begin{pmatrix} P_A^{-1}BP_H^{-1}B^TP_A^{-T} & -P_A^{-1}BP_H^{-1} & P_A^{-1} \\ -P_H^{-1}B^TP_A^{-T} & P_H^{-1} & 0 \\ P_A^{-T} & 0 & 0 \end{pmatrix}.$$

Knowledge of $\tilde{K}^{-1}$ allows to explicitly denote the preconditioned system.

Lemma 5.2.4 *Let Assumption* 5.2.1 *hold. Let P_A and P_H be nonsingular. Denote*

$$\tilde{D} = -P_A^{-1}B,\ \tilde{I}_A = P_A^{-1}A,\ \tilde{I}_A^t = P_A^{-T}A^T,\ \tilde{0}_A^t = \tilde{I}_A^t - I_m.$$

The preconditioned system matrix $\tilde{K}^{-1}K$ is given by

$$\begin{pmatrix} \tilde{D}P_H^{-1}(\tilde{D}^TL_{\phi\phi}+L_{u\phi}) + \tilde{I}_A & \tilde{D}(P_H^{-1}(\tilde{D}^TL_{\phi u}+L_{uu}) - I_k) & -\tilde{D}P_H^{-1}B^T\tilde{0}_A^t \\ P_H^{-1}(\tilde{D}^TL_{\phi\phi}+L_{u\phi}) & P_H^{-1}(\tilde{D}^TL_{\phi u}+L_{uu}) & -P_H^{-1}B^T\tilde{0}_A^t \\ P_A^{-T}L_{\phi\phi} & P_A^{-T}L_{\phi u} & \tilde{I}_A^t \end{pmatrix}.$$

In the special case where the ideal preconditioner for A is chosen, i.e., $P_A = A$, the structure of the preconditioned system matrix becomes apparent. Other special cases, leading to an even simpler structure, are discussed in Section 5.3.3.

Lemma 5.2.5 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular, let $P_A = A$. Use the notation $D = -A^{-1}B$. The preconditioned system matrix $\tilde{K}^{-1}K$ is given by*

$$\begin{pmatrix} DP_H^{-1}(D^TL_{\phi\phi}+L_{u\phi}) + I_m & D(P_H^{-1}(D^TL_{\phi u}+L_{uu}) - I_k) & 0 \\ P_H^{-1}(D^TL_{\phi\phi}+L_{u\phi}) & P_H^{-1}(D^TL_{\phi u}+L_{uu}) & 0 \\ A^{-T}L_{\phi\phi} & A^{-T}L_{\phi u} & I_m \end{pmatrix}.$$

The right–preconditioned system $\tilde{K}^{-1}K$ is the transpose of the left–preconditioned system $K\tilde{K}^{-1}$, $(K\tilde{K}^{-1})^T = \tilde{K}^{-1}K$, in case P_H is symmetric. This is true in both cases $P_A = A$ and $P_A \approx A$. The preconditioned system is not normal in general,

$$K\tilde{K}^{-1}(K\tilde{K}^{-1})^T = K\tilde{K}^{-1}\tilde{K}^{-1}K \neq \tilde{K}^{-1}KK\tilde{K}^{-1} = (K\tilde{K}^{-1})^TK\tilde{K}^{-1}.$$

5.2.3 Preconditioner Solves

Preconditioning can only be useful if the gain due to a better eigenvalue distribution is not offset by computationally expensive matrix operations. In this section it is described how solves with the preconditioned system matrix $\tilde{K}^{-1}K$ can be done in an efficient way. This is first stated for the general case with preconditioners P_A and P_H and then also for a specific choice of the preconditioners.

In order to solve

$$\tilde{K}^{-1}K\,x = \tilde{r}$$

with an iterative solution method in each iteration the action of $\tilde{K}^{-1}K$ on a vector x has to be computed, see e.g. [2, p. 158ff.]. This can be achieved by the successive solution of three linear subsystems and is described in the following.

Lemma 5.2.6 *Let Assumption* 5.2.1 *hold. Let P_A, P_H be nonsingular. The matrix–vector product*

$$\tilde{K}^{-1}K\,x = x_+ \tag{5.2.10}$$

can be carried out at the cost of solving three linear subsystems, with matrices P_A, P_A^T, and P_H, respectively.

Proof: Let $z = (z_1, z_2, z_3)^T$ with $z_1 \in I\!R^m$, $z_2 \in I\!R^k$, $z_3 \in I\!R^m$, and let $x = (x_1, x_2, x_3)^T$ with $x_1 \in I\!R^m$, $x_2 \in I\!R^k$, $x_3 \in I\!R^m$.

Since Equation (5.2.10) is equivalent to $K\,x = \tilde{K}\,x_+$, and since $K\,x$ can be rewritten $K\,x = \tilde{K}\,x + (K - \tilde{K})\,x$, we can compute the action of the matrix $\tilde{K}^{-1}K$ on a vector x by solving

$$\tilde{K}\,(x_+ - x) = (K - \tilde{K})\,x \tag{5.2.11}$$

for $z = x_+ - x$ and then setting $x_+ = x + z$. The solution $z = x_+ - x$ of system (5.2.11) can be computed by successively solving the linear systems

$$\begin{aligned} P_A^T\,z_3 &= L_{\phi\phi}\,x_1 + L_{\phi u}\,x_2 + (A - P_A)^T\,x_3 \\ P_H\,z_2 &= L_{u\phi}\,x_1 + (L_{uu} - P_H)\,x_2 - B^T z_3 \\ P_A\,z_1 &= (A - P_A)\,x_1 - B\,z_2\,, \end{aligned} \tag{5.2.12}$$

for z_3, z_2, and z_1. □

We see that in order to have a practical algorithm, is is necessary not only to solve with the preconditioners P_A, P_A^T, and P_H at moderate cost, but also to be able to apply the preconditioners. In case the ideal preconditioner $P_A^{-1} = A^{-1}$ is chosen, and in case P_H is chosen as L_{uu} (this choice being discussed, along with others, in Section 5.3.3), system (5.2.12) simplifies to

$$A^T\,z_3 = L_{\phi\phi}\,x_1 + L_{\phi u}\,x_2\,, \quad L_{uu}\,z_2 = L_{u\phi}\,x_1 - B^T z_3\,, \quad A\,z_1 = -B\,z_2.$$

In the considered setting one can assume that solving with A, or with its preconditioner P_A, dominates the computations. Then the application of the preconditioner

$\tilde{K}$ requires essentially one solve with A, or with its preconditioner P_A, respectively, and one solve with A^T, or with its preconditioner P_A^T, respectively, in each step of an iterative solution method.

5.3 Eigenvalue Distribution

By left preconditioning with $\tilde{K}$, the linear system (5.2.1), $K\,x = r$, is modified to

$$\tilde{K}^{-1} K\,x = \tilde{r}\,, \tag{5.3.1}$$

where $\tilde{r} = \tilde{K}^{-1}\,r$. With P_A and P_H appropriately chosen in $\tilde{K}$, Equation (5.2.9), the system matrix favorably changes its eigenvalue distribution. If the ideal preconditioner $P_A = A$ is used, it can be shown that at least $2m$ of the eigenvalues of the preconditioned system $\tilde{K}^{-1}K$ are 1. The remaining k eigenvalues are those of the $(k \times k)$–system $P_H^{-1}H$, see Theorem 5.3.1. It can be shown that in case $P_A = A$ the preconditioned system has under natural assumptions only real eigenvalues. This aspect is treated in Lemma 5.3.2.

With the choice of the ideal preconditioner $P_A = A$, the $N \times N$–system $\tilde{K}^{-1}K$ has at most $k+2$ distinct eigenvalues. This is not necessarily maintained in case of an approximate preconditioner $P_A \approx A$. Thus, the clear–cut analysis in Theorem 5.3.1 for the ideal preconditioner $P_A = A$ is perturbed in the case of approximate constraints P_A. An auxiliary result in Lemma 5.3.3 allows the qualitative statement in Theorem 5.3.5.

5.3.1 Eigenvalue Distribution with Exact Equation Solve

In this section, the eigenvalue distribution of the preconditioned system $\tilde{K}^{-1}K$ is analyzed for the choice $P_A = A$.

Theorem 5.3.1 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular, let $P_A = A$. Let μ_i be the eigenvalues of the preconditioned system $\tilde{K}^{-1}K$ and let λ_j be the eigenvalues of $P_H^{-1}H - I_k$ $(i = 1, \ldots, N = 2m+k,\ j = 1, \ldots, k)$. For the μ_i we obtain*

$$\begin{array}{ll} \mu_i = \lambda_i + 1 & (i = 1, \ldots, k)\,, \\ \mu_i = 1 & (i = k+1, \ldots, N)\,. \end{array} \tag{5.3.2}$$

Proof: Write $\tilde{K}^{-1}K = F + I_N$ with the identity $I_N \in \mathbb{R}^{N\times N}$ $(N = 2m+k)$ and

$$F = \begin{pmatrix} DS & DR & 0 \\ S & R & 0 \\ T & U & 0 \end{pmatrix}. \tag{5.3.3}$$

Here, we use the notation

$$\begin{array}{l} D = -A^{-1}B\,,\ T = A^{-T}L_{\phi\phi}\,,\ U = A^{-T}L_{\phi u}\,, \\ S = -P_H^{-1}B^T A^{-T}L_{\phi\phi} + P_H^{-1}L_{u\phi}\,, \\ R = -P_H^{-1}B^T A^{-T}L_{\phi u} + P_H^{-1}L_{uu} - I_k\,. \end{array} \tag{5.3.4}$$

The nonzero eigenvalues of F are given by the solution to the set of equations

$$\begin{aligned} D\,S\,v + D\,R\,s &= \lambda\,v, && (5.3.5)\\ S\,v + R\,s &= \lambda\,s, && (5.3.6)\\ T\,v + U\,s &= \lambda\,w. && (5.3.7) \end{aligned}$$

From (5.3.6) we deduce $D\,S\,v + D\,R\,s = \lambda\,D\,s$, so that with $\lambda \neq 0$ it follows $v = D\,s$. From (5.3.6) we thus get $S\,D\,s + R\,s = \lambda\,s$, which shows that all nonzero eigenvalues λ of F are eigenvalues of $S\,D + R$. To see that all eigenvalues of $S\,D + R$ are eigenvalues of F, let λ be a nonzero eigenvalue of $S\,D + R$, i.e., $S\,D\,s + R\,s = \lambda\,s$. In this situation define $v := D\,s$ and $w := \frac{1}{\lambda}\,(T\,v + U\,s)$. Then Equations (5.3.5), (5.3.6), and (5.3.7) hold.

With the definition in (5.3.4), the matrix $S\,D + R$ coincides with $P_H^{-1}H - I_k$, see Equation (5.2.5), because

$$\begin{aligned} S\,D + R = &\; P_H^{-1}\left(B^T A^{-T} L_{\phi\phi} A^{-1} B - L_{u\phi} A^{-1} B - B^T A^{-T} L_{\phi u} + L_{uu}\right) - I_k \\ = &\; P_H^{-1}\,H - I_k\,. \end{aligned}$$

Then μ is an eigenvalue of $\tilde{K}^{-1}K$ if and only if for $x \in I\!R^N$ the relation

$$\tilde{K}^{-1}K\,x = (F + I_N)\,x = \mu x$$

holds, and so Assertion (5.3.2) is established. □

The matrix F that has just been introduced has interesting properties. It is easy to see from the structure of F in (5.3.3), defined via its entries in (5.3.4), that F has at most $m + k$ nonzero eigenvalues. The zero eigenvalue has an algebraic multiplicity of at least m. In the case that we choose $P_H = H$ we obtain that F has only the zero eigenvalue. Its algebraic multiplicity is then $2m + k$. Its geometric multiplicity is only 2. See Section 5.4. The nonnormality of F is easily checked, $FF^T \neq F^T F$.

To supplement the facts that have been found out about the eigenvalues of the preconditioned system $\tilde{K}^{-1}K$, we now relate a result stating that the preconditioned system $\tilde{K}^{-1}K$ in case $P_A = A$ has under additional assumptions only real eigenvalues. The proof follows easily from the findings of Theorem 5.3.1. In case of approximate constraints $P_A \approx A$, the preconditioned system will not in general have real eigenvalues.

Lemma 5.3.2 *Let Assumption 5.2.1 hold. Let P_H be nonsingular and symmetric, let $P_A = A$. Assume that either P_H or H or both are symmetric positive definite. Then the preconditioned system $\tilde{K}^{-1}K$ has real eigenvalues.*

Proof: In this situation it follows from Theorem 5.3.1 that the eigenvalues μ_j of $\tilde{K}^{-1}K$ are given by 1 and $1 + \lambda_i$ ($i = 1, \ldots, k; j = 1, \ldots, 2m + k$), where λ_i are the eigenvalues of $P_H^{-1}H - I_k$. If either P_H or H or both are symmetric positive definite,

then $P_H^{-1}H - I_k$ is similar to a symmetric matrix which is necessarily diagonalizable and has real eigenvalues. This proves the assertion. □

The assumption of symmetric positive definiteness of the reduced Hessian H is in the optimization setting closely related to assuming that one is already close to a minimizer. This is often not satisfied. But it is not farfetched to require a preconditioner P_H to be symmetric positive definite. For instance, the cheap variant $P_H = I$, discussed below, already constitutes a symmetric positive definite preconditioner.

5.3.2 Eigenvalue Distribution with Approximate Equation Solve

One expects for an appropriately chosen preconditioner P_A of A that the eigenvalues of the generally preconditioned system matrix in Lemma 5.2.4 behave similarly to those of the ideally preconditioned system in Theorem 5.3.1. This statement in Theorem 5.3.5 is anticipated in the following lemma.

Lemma 5.3.3 *Let Assumption 5.2.1 hold. Let P_A and P_H be nonsingular. Let $\tilde{F}$ be defined by* (5.3.3), (5.3.4) *with A replaced by P_A. Let $\bar{\mu}_i$ $(i = 1, \ldots, N = 2m + k)$ be the eigenvalues of $\tilde{F} + I_N$, let $\tilde{\lambda}_j$ $(j = 1, \ldots, k)$ be the eigenvalues of $P_H^{-1}\tilde{H} - I_k$, where the perturbed reduced Hessian $\tilde{H}$ is defined to be*

$$\tilde{H} = B^T P_A^{-T} L_{\phi\phi} P_A^{-1} B - L_{u\phi} P_A^{-1} B - B^T P_A^{-T} L_{\phi u} + L_{uu}\,. \tag{5.3.8}$$

The eigenvalues $\bar{\mu}_i$ obey

$$\begin{aligned} \bar{\mu}_i &= \tilde{\lambda}_i + 1 \qquad (i = 1, \ldots, k)\,, \\ \bar{\mu}_i &= 1 \qquad\quad\;\; (i = k+1, \ldots, N)\,. \end{aligned}$$

Proof: In analogy to Equation (5.3.4), the perturbed quantities $\tilde{D}, \tilde{S}, \tilde{R}, \tilde{T}$, and $\tilde{U}$ are

$$\begin{aligned} &\tilde{D} = -P_A^{-1}B\,,\ \tilde{T} = P_A^{-T} L_{\phi\phi}\,,\ \tilde{U} = P_A^{-T} L_{\phi u}\,, \\ &\tilde{S} = -P_H^{-1}(B^T P_A^{-T} L_{\phi\phi} - L_{u\phi}), \\ &\tilde{R} = -P_H^{-1} B^T P_A^{-T} L_{\phi u} + P_H^{-1} L_{uu} - I_k. \end{aligned}$$

The matrix $\tilde{F}$ is given by

$$\tilde{F} = \begin{pmatrix} \tilde{D}\tilde{S} & \tilde{D}\tilde{R} & 0 \\ \tilde{S} & \tilde{R} & 0 \\ \tilde{T} & \tilde{U} & 0 \end{pmatrix}. \tag{5.3.9}$$

Similar to the preceding proof to Theorem 5.3.1 it can be shown that there are at most k nonzero eigenvalues of $\tilde{F}$ and that these k coincide with the eigenvalues of $P_H^{-1}\tilde{H} - I_k$. The assertion follows. □

The statements that have been made following the proof to Theorem 5.3.1 concerning F in (5.3.3) carry over to the newly defined $\tilde{F}$.

All statements that have been made so far solely relied on linear algebra arguments, and it was only required that the preconditioners be nonsingular. For the statement below that relates the eigenvalues of the generally preconditioned system ($P_A \approx A, P_A \neq A$) to those of the ideally preconditioned system ($P_A = A$) we pose more stringent assumptions on the preconditioners.

To start with, we fix notation that is consistently used within this paragraph and has been partly introduced so far.

Notation 5.3.4 *The spectrum of a matrix $G \in \mathbb{R}^{l\times l}$ ($l \in \mathbb{N}$) is denoted by $\Lambda(G)$. The letters μ_i and λ_i may be used for the eigenvalues ($i = 1, \ldots, l$). The eigenvalues are ordered according to their absolute values, i.e., $|\lambda_1| \geq \ldots \geq |\lambda_l|$. The spectral condition number of G, measured with respect to the 2–norm $\|.\|_2 := \|.\|$, is denoted by $\kappa(G)$. We term $\tilde{G}$ or E a* perturbation *of G, where consistently we write*

$$\tilde{G} = G + E$$

for some "small" matrix $E \in \mathbb{R}^{l\times l}$. If λ_i are the eigenvalues of G, we call $\tilde{\lambda}_i$ the eigenvalues of $\tilde{G}$. The identity matrix in $\mathbb{R}^{l\times l}$ is denoted by I_l. We additionally define for a preconditioner P_G of G the perturbed identities $\tilde{I}_G$ and approximate zeros $\tilde{0}_G$,

$$\begin{aligned} \tilde{I}_G &:= P_G^{-1}G, & \tilde{I}_G^t &:= P_G^{-T}G^T, \\ \tilde{0}_G &:= P_G^{-1}G - I_l, & \tilde{0}_G^t &:= P_G^{-T}G^T - I_l. \end{aligned}$$

We can now phrase the relation between the eigenvalues of the preconditioned system matrix $\tilde{K}^{-1}K$ and the eigenvalues of the low–dimensional convergence–governing system $P_H^{-1}\tilde{H} - I_k$ for the general case.

Theorem 5.3.5 *Let Assumption 5.2.1 hold. Let P_H be nonsingular, let P_A be a preconditioner for A. Let $\tilde{\lambda}_j$ ($j = 1, \ldots, k$) be the eigenvalues of $P_H^{-1}\tilde{H} - I_k$, where $\tilde{H}$ is given in* (5.3.8)*. Then for the eigenvalues $\tilde{\mu}_i$ ($i = 1, \ldots, N = 2m + k$) of the preconditioned system $\tilde{K}^{-1}K$ we have the following bound. For all $\epsilon_A > 0$ there exists a scalar δ_A such that if $\|A - P_A\| < \delta_A$ then it holds*

$$\begin{aligned} -\epsilon_A \;&\leq |\tilde{\mu}_i - \tilde{\lambda}_i - 1| \leq\; \epsilon_A \qquad (i = 1, \ldots, k)\,, \\ -\epsilon_A \;&\leq |\tilde{\mu}_i - 1| \leq\; \epsilon_A \qquad (i = k+1, \ldots, N)\,. \end{aligned}$$

Proof: Interpret $\tilde{K}^{-1}K$ in Lemma 5.2.4 as a perturbation of the matrix $\tilde{F} + I_N$,

$$\tilde{K}^{-1}K = \tilde{F} + I_N + E,$$

where $\tilde{F}$ is defined in (5.3.9), and where E is, with $\tilde{0}_A$, $\tilde{0}_A^t$ as in Notation 5.3.4,

$$E = \begin{pmatrix} \tilde{0}_A & 0 & P_A^{-1}BP_H^{-1}B^T\,\tilde{0}_A^t \\ 0 & 0 & -P_H^{-1}B^T\,\tilde{0}_A^t \\ 0 & 0 & \tilde{0}_A^t \end{pmatrix}. \tag{5.3.10}$$

If $A - P_A$ is sufficiently small in norm, then the norm of $\tilde{0}_A$ and the norm of $\tilde{0}_A^t$ are small also because these continuously depend on the matrix entries. Using the structure of E in (5.3.10), we can thus estimate $\|E\|$ by some constant $\delta > 0$ which continuously depends on δ_A. The assertion follows with the fact that the eigenvalues of the full preconditioned system are continuous functions of the entries of E. This is phrased as a qualitative perturbation theorem e.g. in [78, Th. IV.1.1]: *Let λ be an eigenvalue of G of algebraic multiplicity r. Then for any norm $\|.\|$ and all sufficiently small $\epsilon > 0$ there is a $\delta > 0$ s.t. if $\|E\| < \delta$, the disk $\mathcal{D}(\lambda, \epsilon) = \{\zeta \in \mathbb{C} : |\zeta - \lambda| \leq \epsilon\}$ contains exactly r eigenvalues of the perturbed matrix $\tilde{G} = G + E$.* □

Theorem 5.3.5 is the generalization of Theorem 5.3.1 to the case with an approximation P_A to A. The main assertion is that the eigenvalues of the generally preconditioned system ($P_A \approx A$) behave quantitatively similar to those of the ideally preconditioned system ($P_A = A$). Note that this has been formulated without a requirement on P_H except nonsingularity, this due to the fact that the product $P_H^{-1}\tilde{H}$ does not turn up in the error matrix E in (5.3.10). Often, the small number k ($k \ll m$) of eigenvalues of the low–dimensional system $P_H^{-1}\tilde{H}$ will already lead to significant improvement in the preconditioned iteration versus the original case. If this is not yet satisfying, more stringent requirements on P_H might have to be met.

If we choose P_H to be the exact preconditioner, then this means, allowing a general preconditioning for A through P_A, that $P_H^{-1} = \tilde{H}^{-1}$ for $\tilde{H}$ in (5.3.8).

Theorem 5.3.6 *Let Assumption 5.2.1 hold. Let $P_H = \tilde{H}$, and let P_A be a preconditioner for A. Then for the eigenvalues $\tilde{\mu}_i$ of $\tilde{K}^{-1}K$ we have the following bound. For all $\epsilon_A > 0$ there exists a scalar δ_A such that if $\|A - P_A\|_2 < \delta_A$ then*

$$-\epsilon_A \leq |\tilde{\mu}_i - 1| \leq \epsilon_A \qquad (i = 1, \ldots, 2m + k)\,.$$

Proof: If $P_H = \tilde{H}$, then the eigenvalues $\tilde{\lambda}_j$ of $P_H^{-1}\tilde{H} - I_k$ vanish. Theorem 5.3.5 yields the simplified estimate that is stated above. □

Naturally, the question of a quantitative bound on the perturbation in the eigenvalues arises in this context. However, any general perturbation bound on the eigenvalues of a matrix has to be pessimistic because it must account for ill–conditioned behavior. See the theorem by Elsner, e.g. [29], [78, Th. IV.1.3]. Possible are changes with the N–th root of the error, and so are changes that are proportional to the error itself. The latter is true in the normal case, compare Section 5.1, but can be valid in the nonnormal case as well. A bound by Henrici, see [78, Th. IV.1.9], phrased in terms of departure

from normality, provides a transition between these cases. For a discussion of the effects of nonnormality see also [15], [43]. We have not found general quantitative bounds to provide useful information for the systems that we consider.

As the last point in this section, we relate an easy extension to a result in the case of exact constraints, $P_A = A$. The following corollary to Lemma 5.3.2 is immediate by considering Lemma 5.3.3.

Corollary 5.3.7 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular and symmetric, Let P_A be nonsingular. Let $\tilde{H}$ be defined by* (5.3.8). *Let $\tilde{F}$ be defined by* (5.3.9). *Assume that either P_H or $\tilde{H}$ or both are symmetric positive definite. Then the matrix $\tilde{F} + I$ has real eigenvalues.*

5.3.3 Eigenvalue Distribution for Special Choices of $\mathbf{P_H}$

The case with the ideal preconditioner $P_A = A$ was treated in depth already in Section 5.3.1. The overall result in this situation is that the eigenvalues of the preconditioned system are, with high multiplicity, the value 1, and the eigenvalues of $P_H^{-1}H$. We now consider important special cases of this situation.

$\mathbf{P_A = A}$ and $\mathbf{P_H = H}$

The case where both preconditioners P_A and P_H are chosen as the ideal preconditioners for A and H, respectively, has mostly theoretical character. For completeness we state the result in this case too.

Theorem 5.3.8 *Let Assumption* 5.2.1 *hold. Let $P_A = A$ and $P_H = H$. Then all the eigenvalues μ_i $(i = 1, \ldots, 2m + k)$ of $\tilde{K}^{-1}K$ equal* 1.

Proof: This result follows from Theorem 5.3.1 using the fact that for $P_H = H$ all eigenvalues of $P_H^{-1}H - I$ vanish. Therefore all eigenvalues μ_i of $\tilde{K}^{-1}K$ are of size 1. □

In this special case all eigenvalues of the preconditioned system are 1 — although $\tilde{K}^{-1}K$, compare Lemma 5.2.5, is not the identity or anywhere near to it component–wise. Although a preconditioner can often be perceived as an approximate inverse, this need not be the case. In the case considered here we also have that $E = 0$ and $\tilde{K}^{-1}K = F + I$, where all eigenvalues of F vanish. The zero eigenvalue of F has algebraic multiplicity $2m + k$ whereas its geometric multiplicity is considerably smaller. We will see in Section 5.4, specifically in Lemma 5.4.4, that the minimum polynomial of the preconditioned system is in this case of degree 3 only. In this case, GMRES can – in exact arithmetic – be shown to encounter an invariant subspace of the system matrix after three steps which is confirmed in the numerical tests. See Section 7.3.

The drawback that accompanies the nice result that all eigenvalues of the preconditioned system are 1 and that only three steps are necessary for GMRES is the fact that this choice of preconditioner is very expensive. Although H is "small" and the

solve with it should not be prohibitive in case H can be computed explicitly, it is just this assembly of the matrix that might not be possible. In case it is only possible to solve with the submatrices involved or to apply them, e.g. if the matrices A and B are not readily available, H cannot be computed explicitly. Instead, a solve with H then requires solves with A and A^T and is likely to be very expensive.

Another approach is to use $P_A = A$ and a preconditioner $P_H \approx H$. This could for example be a preconditioner for the reduced Hessian from a quasi–Newton approach. The cost might also be reduced by choosing $P_H = \tilde{H}$ in case a good preconditioner P_A for A is available. Compare Theorem 5.3.6. Alternatively, preconditioners $P_H \approx H$ and $P_A \approx A$ could be chosen independently from each other.

$\mathbf{P_A = A}$ and $\mathbf{P_H = L_{uu}}$

Instead of the expensive choice $P_H = H$, a cheap variant, if applicable, can be to take $P_H = L_{uu}$. Often, L_{uu} can be derived from the user–defined cost function, when the constraints depend in a nonlinear way only on the controls u but not the states ϕ. In this case we also have $L_{\phi u} = 0$. Then we obtain the statement of Theorem 5.3.10. First we give the corresponding result for the case where $P_H = L_{uu}$ and where the off–diagonal entries of the upper left block are nonzero, $L_{\phi u} \neq 0$, $L_{u\phi} \neq 0$.

Theorem 5.3.9 *Let Assumption* 5.2.1 *hold. Set $P_H = L_{uu}$ and $P_A = A$. Let λ_j ($j = 1, \ldots, k$) be the eigenvalues of $L_{uu}^{-1}H - I_k$. For the eigenvalues μ_i of the preconditioned system $\tilde{K}^{-1}K$ we obtain*

$$\begin{aligned} \mu_i &= \lambda_i + 1 \qquad (i = 1, \ldots, k)\,, \\ \mu_i &= 1 \qquad (i = k+1, \ldots, 2m+k)\,. \end{aligned}$$

Proof: The situation is a special case of Theorem 5.3.1. Under the assumptions stated we obtain for the quantities in (5.3.4) that

$$\begin{aligned} &D = -A^{-1}B\,,\ T = A^{-T}L_{\phi\phi}\,,\ U = A^{-T}L_{\phi u}\,, \\ &S = -L_{uu}^{-1}(B^TA^{-T}L_{\phi\phi} - L_{u\phi}), \\ &R = -L_{uu}^{-1}B^TA^{-T}L_{\phi u}\,. \end{aligned}$$

This results in

$$L_{uu}^{-1}H - I_k = SD + R = L_{uu}^{-1}(B^TA^{-T}L_{\phi\phi}A^{-1}B - B^TA^{-T}L_{\phi u} - L_{u\phi}A^{-1}B),$$

which yields the statement of the theorem. □

Note that in this case the matrix representation for $\tilde{K}^{-1}K = F + I_N + E$ with $E = 0$ and F given in (5.3.3) has the special form

$$\begin{pmatrix} DL_{uu}^{-1}(D^TL_{\phi\phi}+L_{u\phi}) + I_m & DL_{uu}^{-1}D^TL_{\phi u} & 0 \\ L_{uu}^{-1}(D^TL_{\phi\phi}+L_{u\phi}) & L_{uu}^{-1}D^TL_{\phi u}+I_k & 0 \\ A^{-T}L_{\phi\phi} & A^{-T}L_{\phi u} & I_m \end{pmatrix}.$$

We now turn to the case where in addition the mixed entries $L_{\phi u}, L_{u\phi}$ are zero. Then the eigenvalues of the preconditioned system are immediately deduced from the matrix representation.

Theorem 5.3.10 *Let Assumption* 5.2.1 *hold. Assume in addition that* $L_{\phi u} = 0$ *and* $L_{u\phi} = 0$. *Set* $P_H = L_{uu}$ *and* $P_A = A$. *Let* λ_j $(j = 1, \ldots, k)$ *be the eigenvalues of* $L_{uu}^{-1} B^T A^{-T} L_{\phi\phi} A^{-1} B$. *For the eigenvalues* μ_i *of the preconditioned system* $\tilde{K}^{-1}K$ *we obtain*

$$\begin{aligned} \mu_i &= \lambda_i + 1 \qquad (i = 1, \ldots, k)\,, \\ \mu_i &= 1 \qquad\quad (i = k+1, \ldots, 2m+k)\,. \end{aligned}$$

Proof: The situation is a special case of Theorem 5.3.1 and of the immediately preceding Theorem 5.3.9. Under the assumptions stated we obtain for the quantities in (5.3.4)

$$D = -A^{-1}B,\ \ T = A^{-T}L_{\phi\phi},\ \ U = 0,\ \ S = -L_{uu}^{-1}B^T A^{-T} L_{\phi\phi},\ \ R = 0.$$

This results in

$$P_H^{-1}H - I_k = SD + R = L_{uu}^{-1}B^T A^{-T} L_{\phi\phi} A^{-1} B,$$

which yields the statement of the theorem. □

Note that in this case the matrix representation for $\tilde{K}^{-1}K = \tilde{F} + I_N + E$ is even simpler than in the situation stated above. As before, $E = 0$ and $\tilde{F} = F$. Moreover, F in (5.3.3) reduces to

$$F = \begin{pmatrix} DS & 0 & 0 \\ S & 0 & 0 \\ T & 0 & 0 \end{pmatrix}.$$

Thus, the preconditioned system is

$$\tilde{K}^{-1}K = \begin{pmatrix} DL_{uu}^{-1}D^T L_{\phi\phi} + I_m & 0 & 0 \\ L_{uu}^{-1}D^T L_{\phi\phi} & I_k & 0 \\ A^{-T}L_{\phi\phi} & 0 & I_m \end{pmatrix}.$$

Its spectrum is obviously composed of the eigenvalue 1 (with algebraic multiplicity of at least $m + k$ instead of m as previously noted for the other cases), and of the eigenvalues of $DL_{uu}^{-1}D^T L_{\phi\phi} + I_m$. For completeness we add the following lemma which states that the nonzero eigenvalues of $DL_{uu}^{-1}D^T L_{\phi\phi}$ and $L_{uu}^{-1}D^T L_{\phi\phi}D$ are identical.

Lemma 5.3.11 *Let* $P_A \in \mathbb{R}^{m\times m}$ *and* $P_H \in \mathbb{R}^{k\times k}$ *be nonsingular. Let* $B \in \mathbb{R}^{m\times k}$, *let* $L_{\phi\phi} \in \mathbb{R}^{m\times m}$, *let* $k < m$ *for* $k, m \in \mathbb{N}$. *In this situation,*

$$\Lambda(P_H^{-1}B^T P_A^{-T} L_{\phi\phi} P_A^{-1} B) \cup \{0\} = \Lambda(P_A^{-1}BP_H^{-1}B^T P_A^{-T} L_{\phi\phi})\,.$$

Proof: The matrix $P_A^{-1}BP_H^{-1}B^TP_A^{-T}L_{\phi\phi} \in \mathbb{R}^{m\times m}$ has, due to the dimension of its factor P_H, at most k possibly nonzero eigenvalues. We use the notation $\tilde{D} = P_A^{-1}B$ again. The k possibly nonzero eigenvalues of $\tilde{D}P_H^{-1}\tilde{D}^TL_{\phi\phi}$ are identical to the nonzero eigenvalues of $P_H^{-1}\tilde{D}^TL_{\phi\phi}\tilde{D} \in \mathbb{R}^{k\times k}$. The nonzero eigenvalues of $P_H^{-1}\tilde{D}^TL_{\phi\phi}\tilde{D}$ are, with $x \in \mathbb{R}^k$, determined by the equation $P_H^{-1}\tilde{D}^TL_{\phi\phi}\tilde{D}\,x = \lambda\,x$. Premultiplying this equation by $\tilde{D}$, one obtains $\tilde{D}P_H^{-1}\tilde{D}^TL_{\phi\phi}\,y = \lambda\,y$, where $y \in \mathbb{R}^m$ is defined to be $y = \tilde{D}\,x$. Thus, all nonzero eigenvalues of $P_H^{-1}\tilde{D}^TL_{\phi\phi}\tilde{D}$ are nonzero eigenvalues of the larger matrix $\tilde{D}P_H^{-1}\tilde{D}^TL_{\phi\phi}$. □

$\mathbf{P_A = A}$ and $\mathbf{P_H = I}$

A variant even cheaper than the one considered in the immediately preceding section is to take $P_H = I_k$. Again, the situation is a special case of Theorem 5.3.1.

Theorem 5.3.12 *Let Assumption* 5.2.1 *hold. Set $P_H = I_k$ and $P_A = A$. Let λ_j be the eigenvalues of H ($j = 1,\ldots,k$). For the eigenvalues μ_i of the preconditioned system $\tilde{K}^{-1}K$ we obtain*

$$\begin{aligned} \mu_i &= \lambda_i \qquad (i = 1,\ldots,k)\,, \\ \mu_i &= 1 \qquad (i = k+1,\ldots,2m+k)\,. \end{aligned}$$

At first sight it is not clear why the identity could be a sensible preconditioner for the reduced Hessian. However, it is cheap, as already noted, and readily available. Also, in terms of expected performance, the number of eigenvalues distinct from 1 is small as before. See also Section 6.3.

5.4 Expected Performance

In this section we address the benefits that can be achieved for iterative methods, specifically for GMRES [70], by preconditioning a system K of the structure (5.2.2) with the indefinite preconditioner $\tilde{K}$ that we investigate. To this end recall the results assembled in Section 5.1, specifically Theorem 5.1.5. In the original paper [70, Prop. 2], Saad and Schultz showed that in exact arithmetic GMRES requires at most d_{min} iterations, where d_{min} is the degree of the minimum polynomial of the system matrix. Thus, an upper limit on the iteration number is provided which can be considerably smaller than N. The result says that the number of iterations is small if the spectrum consists of a small number of non–defective eigenvalues of high multiplicity. We will now apply this theorem to the system preconditioned with $\tilde{K}$. See Equation (5.2.9) for the preconditioner $\tilde{K}$. In case the constraints are maintained accurately in the preconditioner, the preconditioned system has at most $k+1$ distinct eigenvalues. Specifically it is shown in our Theorem 5.3.1 that in case $P_A = A$ the preconditioned system $\tilde{K}^{-1}K$ has only k eigenvalues distinct from 1. These k values are the eigenvalues of $P_H^{-1}H - I_k$. This leads us to closely inspect the minimum

polynomial of the preconditioned system. It is seen that the eigenvalue 1 has algebraic multiplicity $2m$, but geometric multiplicity 2. Thus the assertion of Theorem 5.4.1.

Theorem 5.4.1 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular, let $P_A = A$. GMRES will in exact arithmetic terminate the iteration on the preconditioned system $\tilde{K}^{-1}K\,x = \tilde{K}^{-1}\,r$ after at most $k+2$ iterations.*

Proof: The eigenvalue 1 of the matrix $\tilde{K}^{-1}K \in \mathbb{R}^{N\times N}$, $N = 2m+k$, has the multiplicity $2m$ in the characteristic polynomial. To get insight into the degree of the minimum polynomial of the preconditioned system $\tilde{K}^{-1}K$, we investigate the matrix polynomial

$$(\tilde{K}^{-1}K - I_N)\cdot\prod_{i=1}^{k}(\tilde{K}^{-1}K - (\lambda_i+1)I_N) = F\cdot\prod_{i=1}^{k}(F-\lambda_i I_N)\,. \tag{5.4.1}$$

Here, the λ_i ($i=1,\ldots,k$) are the eigenvalues of the matrix $P_H^{-1}H - I_k = \tilde{0}_H$. We define for $l = 1,2,\ldots,k$ ($k\in\mathbb{N}$) the matrix polynomial

$$p_{\lambda,l}(\tilde{0}_H) = \prod_{i=1}^{l}(\tilde{0}_H - \lambda_i I_k)\,, \tag{5.4.2}$$

so that for $l=k$ we have the characteristic polynomial $p_{\lambda,k}$ of the matrix $\tilde{0}_H$. By the theorem of Hamilton–Cayley it follows

$$p_{\lambda,k}(\tilde{0}_H) = \prod_{i=1}^{k}(\tilde{0}_H - \lambda_i I_k) = 0_{k\times k}\,. \tag{5.4.3}$$

The following equality holds for the matrix polynomial in (5.4.1) with matrices T_k, U_k as defined below in (5.4.5),

$$F\cdot\prod_{i=1}^{k}(F-\lambda_i I) = \begin{pmatrix} D\cdot p_{\lambda,k}(\tilde{0}_H)\cdot S & D\cdot p_{\lambda,k}(\tilde{0}_H)\cdot R & 0\\ p_{\lambda,k}(\tilde{0}_H)\cdot S & p_{\lambda,k}(\tilde{0}_H)\cdot R & 0\\ T_k & U_k & 0\end{pmatrix}. \tag{5.4.4}$$

This leads with (5.4.3) to the result

$$\begin{aligned} & (\tilde{K}^{-1}K - I_N)^2\cdot\textstyle\prod_{i=1}^{k}(\tilde{K}^{-1}K-(\lambda_i+1)I_N)\\ = \;& F^2\cdot\textstyle\prod_{i=1}^{k}(F-\lambda_i I_N)\\ = \;& \begin{pmatrix} DS & DR & 0\\ S & R & 0\\ T & U & 0\end{pmatrix}\cdot\begin{pmatrix} 0 & 0 & 0\\ 0 & 0 & 0\\ T_k & U_k & 0\end{pmatrix} = 0_{N\times N}\,. \end{aligned}$$

Since this matrix polynomial has degree $k+2$ and annihilates $\tilde{K}^{-1}K$, and since the minimum polynomial is the polynomial of minimum degree to annihilate $\tilde{K}^{-1}K$,

the minimum polynomial of $\tilde{K}^{-1}K$ also is of degree $k+2$ or less. With Theorem 5.1.5 the assertion follows.

The equality in (5.4.4) can be deduced by induction as given in the remainder of the proof. We define additional matrices T_l and U_l ($l = 1, \ldots, k$) via the recursion

$$\begin{aligned} T_0 &= T, \\ U_0 &= U, \\ T_l &= T_{l-1}(DS - \lambda_l I_k) + U_{l-1}\, S, \\ U_l &= T_{l-1}\, DR + U_{l-1}(R - \lambda_l I_k). \end{aligned} \tag{5.4.5}$$

The assertion to be proven by induction is

$$\begin{pmatrix} D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot S & D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot R & 0 \\ p_{\lambda,l-1}(\tilde{0}_H) \cdot S & p_{\lambda,l-1}(\tilde{0}_H) \cdot R & 0 \\ T_{l-1} & U_{l-1} & 0 \end{pmatrix} \cdot \prod_{i=l}^{k} (F - \lambda_i I_N)$$

$$= \begin{pmatrix} D \cdot p_{\lambda,l}(\tilde{0}_H) \cdot S & D \cdot p_{\lambda,l}(\tilde{0}_H) \cdot R & 0 \\ p_{\lambda,l}(\tilde{0}_H) \cdot S & p_{\lambda,l}(\tilde{0}_H) \cdot R & 0 \\ T_l & U_l & 0 \end{pmatrix} \cdot \prod_{i=l+1}^{k} (F - \lambda_i I_N)\,.$$

For $l = 1$ one finds the identity

$$F \cdot \prod_{i=1}^{k} (F - \lambda_i I_N)$$

$$= \begin{pmatrix} DS & DR & 0 \\ S & R & 0 \\ T_0 & U_0 & 0 \end{pmatrix} \begin{pmatrix} DS - \lambda_1 I_m & DR & 0 \\ S & R - \lambda_1 I_k & 0 \\ T & U & -\lambda_1 I_m \end{pmatrix} \cdot \prod_{i=2}^{k} (F - \lambda_i I)$$

$$= \begin{pmatrix} D(SD + R - \lambda_1 I_m)S & D(SD + R - \lambda_1 I_m)R & 0 \\ (SD + R - \lambda_1 I_m)S & (SD + R - \lambda_1 I_m)R & 0 \\ (TD + U)S - \lambda_1 T & (TD + U)R - \lambda_1 U & 0 \end{pmatrix} \cdot \prod_{i=2}^{k} (F - \lambda_i I_N)$$

$$= \begin{pmatrix} D \cdot p_{\lambda,1}(\tilde{0}_H) \cdot S & D \cdot p_{\lambda,1}(\tilde{0}_H) \cdot R & 0 \\ p_{\lambda,1}(\tilde{0}_H) \cdot S & p_{\lambda,1}(\tilde{0}_H) \cdot R & 0 \\ T_1 & U_1 & 0 \end{pmatrix} \cdot \prod_{i=2}^{k} (F - \lambda_i I_N)\,.$$

The step from $l-1$ to l,

$$\begin{pmatrix} D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot S & D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot R & 0 \\ p_{\lambda,l-1}(\tilde{0}_H) \cdot S & p_{\lambda,l-1}(\tilde{0}_H) \cdot R & 0 \\ T_{l-1} & U_{l-1} & 0 \end{pmatrix} \begin{pmatrix} DS - \lambda_l I_m & DR & 0 \\ S & R - \lambda_l I_k & 0 \\ T & U & -\lambda_l I_m \end{pmatrix}$$

$$= \begin{pmatrix} D \cdot p_{\lambda,l}(\tilde{0}_H) \cdot S & D \cdot p_{\lambda,l}(\tilde{0}_H) \cdot R & 0 \\ p_{\lambda,l}(\tilde{0}_H) \cdot S & p_{\lambda,l}(\tilde{0}_H) \cdot R & 0 \\ T_l & U_l & 0 \end{pmatrix},$$

follows due to the identity

$$\begin{aligned} & D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot S \cdot DS - \lambda_l \cdot D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot S + D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot R \cdot S \\ = \; & D \cdot p_{\lambda,l-1}(\tilde{0}_H) \cdot (SD + R - \lambda_l I_k) \cdot S \\ = \; & D \cdot p_{\lambda,l}(\tilde{0}_H) \cdot S \,. \end{aligned}$$

in the $(1,1)$–entry. Similar manipulations apply in the $(1,2)$, $(2,1)$, and $(2,2)$–entries. □

The straightforward extension of the preceding theorem to "GMRES stops after k_d iterations on the preconditioned system" for the case where $P_H^{-1}H - I_k$ has $k_d \leq k$ distinct eigenvalues can only be done under additional assumptions. This statement presupposes that either P_H or H or both are symmetric positive definite. See Theorem 5.4.3. However, the following immediate statement can be made.

Corollary 5.4.2 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular, let $P_A = A$. Let λ_i ($i = 1, \ldots, k_d; k_d \leq k$) denote all distinct eigenvalues of $P_H^{-1}H - I_k$. Let $m_i > 0$ ($i = 1, \ldots, k_d$) be their respective multiplicity in the minimum polynomial, and let $\sum_{i=1}^{k_d} m_i = d_{min} \leq k$. GMRES will in exact arithmetic terminate the iteration on the preconditioned system $\tilde{K}^{-1}K\,x = \tilde{K}^{-1}\,r$ after at most $d_{min} + 2$ iterations.*

Proof: Again, we consider the polynomial (5.4.1). Since m_i ($i = 1, \ldots, k_d$) are the multiplicities in the minimum polynomial of the distinct eigenvalues λ_i, it follows already $p_{\lambda,d_{min}}(\tilde{0}_H) = 0_k$ for the polynomial defined in (5.4.2), so that

$$F \cdot \prod_{i=1}^{k_d} (F - \lambda_i I)^{m_i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ T_k & U_k & 0 \end{pmatrix}.$$

With the arguments used in the proof to Theorem 5.4.1 the assertion follows. □

In case the convergence–governing system is diagonalizable, the number of distinct eigenvalues already tells the whole story.

Theorem 5.4.3 *Let Assumption* 5.2.1 *hold. Let P_H be nonsingular and symmetric, let $P_A = A$. Let either P_H or H or both be symmetric positive definite. Let λ_i denote the distinct eigenvalues of $P_H^{-1}H - I_k$ ($i = 1, \ldots, k_d; k_d \leq k$). In exact arithmetic, GMRES will terminate the iteration on the preconditioned system $\tilde{K}^{-1}K\,x = \tilde{K}^{-1}\,r$ after at most $k_d + 2$ iterations.*

Proof: Since it is assumed that either P_H or the reduced Hessian H is symmetric positive definite, $P_H^{-1}H$ is diagonalizable as the product of a symmetric and a symmetric

positive definite matrix. Thus we are in the diagonalizable case, and the minimum polynomial of $\tilde{0}_H = P_H^{-1}H - I_k$ is already given by

$$p_{\lambda,k_d}(\tilde{0}_H) = \prod_{i=1}^{k_d}(\tilde{0}_H - \lambda_i I_k).$$

The assertion follows. □

The first situation discussed in Section 5.3.3 is where both ideal preconditioners, $P_A = A$, $P_H = H$, are used. Then, $\tilde{0}_H$ is the zero matrix and thus has the minimum polynomial identical to zero with the single eigenvalue 0. Then, three steps are sufficient to build the solution in the appropriate Krylov spaces.

Lemma 5.4.4 *Let Assumption* 5.2.1 *hold. Let $P_H = H$, let $P_A = A$. In exact arithmetic, GMRES will terminate the iteration on the preconditioned system $\tilde{K}^{-1}K\,x = \tilde{K}^{-1}r$ after at most* 3 *iterations.*

The assumptions of Corollary 5.4.2 are not sufficient to obtain the result that the Krylov subspace generated by the preconditioned system and the preconditioned initial residual, $\mathcal{K}(\tilde{K}^{-1}K, \tilde{K}^{-1}r_0)$, has at most dimension $k_d + 2$, where k_d is the number of distinct eigenvalues of the preconditioned system. Also, symmetry alone of P_H is not sufficient. This is discussed in detail in Section 6.3.

The foregoing discussion has shown that in exact arithmetic GMRES will take no more than $d_{min} + 2$ steps to solve the preconditioned system (5.3.1) when the exact subpreconditioner $P_A = A$ is employed. The degree d_{min} of the minimum polynomial of the $(k \times k)$–system $P_H^{-1}H$ is crucial. Of course, d_{min} is usually not known. If P_H or H are symmetric positive definite, the number of distinct eigenvalues of $P_H^{-1}H$ is sufficient to know. However, this is usually not known either. In any of the considered cases, no more than $k + 2$ steps are taken, where k is the number of control variables and thus considerably smaller than the system's dimension $N = 2m + k$.

We have seen in Section 5.3 that the situation is considerably more involved when the exact constraints are replaced by approximations, $P_A \neq A$, $P_A \approx A$. The few convergence results that are known for the general case are, because they account for the general case, likely to be very pessimistic.

Chapter 6

Additional Preconditioning Strategies

In this chapter we are concerned with issues that are closely related to the preconditioner investigated in the preceding chapter. We start with a review of several preconditioners, Section 6.1. Some of these are general–purpose preconditioners, and others are designed within specific settings that are related to our background. Three symmetric positive definite block preconditioners are reviewed in Section 6.2. These are derived and analyzed in [5] before a background almost identical to ours. These preconditioners prove in their ideal versions less effective for our problem than the indefinite $\tilde{K}$. See Chapter 7. Nevertheless, they can be judged advantageous in that they maintain the symmetry of the system. Next, in Section 6.3, we are concerned with the indefinite preconditioner that has been analyzed in [49].

6.1 Brief Review of Some Preconditioners

In this section and in the following, several of the presented preconditioners differ from the preconditioner $\tilde{K}$ of the preceding chapter in the way they are applied. Again, compare also Section 5.1, we give a brief review of preconditioning which also serves to fix notation.

In the numerical solution of large sparse linear systems

$$K\,x = r, \tag{6.1.1}$$

$K \in I\!R^{N\times N}$, an efficient preconditioner can be the "most important part of an iterative algorithm" as states e.g. [1]. The general issue in preconditioning is to construct a system that is equivalent to the original system and at the same time easier to solve. Equivalent to (6.1.1) with nonsingular P_L and P_R, the generally preconditioned system is usually written as

$$P_L^{-1} K P_R^{-T}\, P_R^T\, x = P_L^{-1}\, r. \tag{6.1.2}$$

Even for a symmetric original system K, the preconditioned system matrix $P_L^{-1} K P_R^{-T}$ is in the general case nonsymmetric. We have so far focused our attention on left–sided

preconditioning, i.e., we set $P_R = I$ in the previous chapter and considered

$$P_L^{-1} K\, x = P_L^{-1}\, r.$$

A common choice, however, especially when dealing with an originally symmetric system matrix K, is two–sided preconditioning, e.g. [5]. Instead of the original system (6.1.1), the equivalent system

$$P^{-1} K P^{-T}\, P^T\, x = P^{-1}\, r \tag{6.1.3}$$

is considered which is obtained from (6.1.2) with the choice $P_L = P_R$. This preconditioned system is still symmetric which is often considered advantageous. The approach (6.1.3) is also known as positive definite preconditioning because often the matrix $M = PP^T$, positive definite by construction, is then referred to as the preconditioner instead of P. This is justified because the matrices $P^{-1} K P^{-T}$ and $M^{-1} K$ are similar, i.e., have identical spectra.

Classical preconditioning methods are for instance incomplete factorization methods. Sparsity of a matrix is easily lost within a factorization process, see e.g. [2, Ch. 1.4]. The idea of incomplete factorization methods is to reject those "fill–in" entries which either occur in positions outside an accepted sparsity pattern or which are small relative to some chosen entries. This is referred to as incomplete factorization by position or by value, respectively, and related e.g. in [2, Ch. 7].

Decompositions, whether complete or incomplete, can be used as a preconditioner in various ways. Consider for example the Bunch–Parlett factorization, see e.g. [39, Ch. 4.4], of a symmetric indefinite matrix K. This factorization gives

$$K = \Pi L D L^T \Pi^T, \tag{6.1.4}$$

with Π a permutation matrix, L lower triangular, and D block–diagonal with blocks of size 1×1 and 2×2. One possibility is to have that decomposition computed for K for a chosen level of fill–in. Numerical results are reported e.g. in [31]. Another possibility is to have that decomposition computed for an approximation to K, e.g. [34]. Based on a factorization (6.1.4), an incomplete decomposition, modified to fit the requirements of a multigrid framework, is derived in [72].

Another classical idea is to employ splittings of the system matrix. Splittings based on the diagonal and triangular parts of K are used to derive basic iterative methods (and their block versions), for instance Jacobi, Gauss–Seidel, and SOR [2, Chs. 5, 6.5]. A related idea is pursued in [32] to construct a block SOR preconditioner for the indefinite KKT matrix arising within the considered interior–point method.

Nash and Sofer [63] are concerned with preconditioning reduced matrices by a truncated series approach. For a numerical experience with this approach see [18].

It was already pointed out in the preceding chapter, Section 5.1, that we do not attempt to use preconditioners of such general design, but focus on preconditioners

that take advantage of the special form and features of the systems. We are interested in constructing block preconditioners that maintain the structure of the original system and allow to exploit that structure to computational advantage. This line of thinking can also be found in the works [3], [5], [34], [49], [54], [69], [73], [74] which we review in the sequel.

The relation between the eigenvalues of the entire system K and the eigenvalues of its upper left block and the singular values of the constraints C is investigated by Rusten and Winther [69]. They are concerned with the solution of a saddle point problem. This gives rise to a system of similar structure as the KKT matrix. For the solution of that system with MINRES, Rusten and Winther use block–diagonal preconditioners. Silvester and Wathen [74], [75], are also motivated by saddle point problems. Their subpreconditioners are tailored to their specific application. A similar approach is pursued in [73] within a multigrid framework.

Gill, Murray, Ponceleón, and Saunders [34] follow the idea to use preconditioners composed of blocks, which allows to specifically address problematic blocks in the original system. Gill et al. are motivated by interior–point methods. We briefly point out where the obvious focus is in the design of preconditioners for such methods.

In the presence of bound constraints

$$\phi \geq 0,\ u \geq 0 \tag{6.1.5}$$

for the linearly equality–constrained quadratic programming problem in standard form, (1.1.16) subject to (1.1.17), interior–point methods are attractive solution methods, especially for large–scale problems. Unlike active set methods which usually move along the boundary of the set $\{(\phi, u) \mid \phi \geq 0, u \geq 0\}$, interior–point methods, as suggested by the name, generate iterates (ϕ, u) that are in the interior, i.e., satisfy $\phi > 0,\ u > 0$. See e.g. [82] for an overview of interior–point methods. We briefly sketch the KKT systems that are set up in primal–dual Newton interior–point methods and barrier methods for the solution of the quadratic programming problem (1.1.16) s.t. (1.1.17) and (6.1.5).

The construction of a primal–dual Newton interior–point methods is based on so called perturbed KKT conditions which give rise to a nonsymmetric system. The nonsymmetric system, however, can be reduced to a symmetric system

$$\begin{pmatrix} L_{zz} + D & C^T \\ C & 0 \end{pmatrix} \tag{6.1.6}$$

by variable elimination. See e.g. [32] for a stable reduction. The matrix D is a diagonal matrix which, as variables ϕ_j or u_i approach the bound, has increasingly larger entries.

Similar systems (6.1.6) are constructed when using e.g. logarithmic barrier functions for the solution of the quadratic programming problem (1.1.16) s.t. (1.1.17) and (6.1.5). During the iteration, the barrier parameter is adjusted and causes the entries in D to grow, because, as before, large quantities are added to the diagonal of L_{zz} as variables ϕ_j or u_i approach the bound. For details on the barrier method see [82].

Growth in the entries of the upper diagonal of (6.1.6) causes computational problems which can be overcome by appropriately addressing the large entries. This leads to major improvement if the constraints C are simple. But this is not sufficient if the constraints are not simple which is usually the case in the control context. Compare the preceding chapter. Preconditioners are derived in [34] which first try to overcome the problems induced by the large entries, and only then can try to take the constraints into account. See also the work [5] which distinguishes several cases for the bound constraints (6.1.5), namely "state and control constraints", "only state constraints", "only control constraints", and "no constraints". The symmetric positive definite preconditioners derived in [5], analyzed in detail in [3], are described in Section 6.2.

Also motivated by interior–point methods are Oliveira and Sorensen [65]. They choose to partition the matrix according to the size of the problematic entries to derive a preconditioner.

All block preconditioners that we reviewed so far are symmetric positive definite. For symmetric, but highly indefinite systems like the KKT system that is constantly considered in this work, "this restriction on a preconditioner is rather unnatural" [31]. Instead, one would like to be able to admit a preconditioner that is indefinite like K is. This is done e.g. in [31], [49], [54]. Freund [31] advocates indefinite preconditioning over positive definite preconditioning. Incomplete decompositions are used in this work like, as mentioned above, by Freund and Jarre in [32]. See also [72].

Klawonn [54] is interested in saddle point problems with a penalty term. For such problems, an indefinite preconditioner is presented in [54]. This is either a left or a right preconditioner, and the subpreconditioners on the diagonal are complemented with the exact constraint matrix. For a similar situation, a block–diagonal symmetric positive definite preconditioner is analyzed in [53].

Another indefinite preconditioner, termed "constraint preconditioner", is analyzed in [49]. See Section 6.3. This section will conclude our review of preconditioners. Before turning to the detailed reviews of Sections 6.2 and 6.3, we comment briefly on two additional works, [47], [61], that are concerned with preconditioners.

The ideal preconditioner is often thought of to approximate the inverse of the original system matrix. This is not a must, however, as we already noted. Murphy, Golub, and Wathen [61] show that for matrices

$$\begin{pmatrix} G & C_1^T \\ C_2 & 0 \end{pmatrix} \tag{6.1.7}$$

of KKT form with $G \in \mathbb{R}^{n\times n}$ nonsingular, $C_1, C_2 \in \mathbb{R}^{m\times n}$ ($n \geq m$), the blockdiagonal preconditioner

$$\begin{pmatrix} G & 0 \\ 0 & -S \end{pmatrix}, \quad S = -C_2 G^{-1} C_1^T, \tag{6.1.8}$$

furnishes a preconditioned system that has at most four distinct eigenvalues, three if nonsingular. Furthermore, if the system is nonsingular, the eigenvalues are of simple multiplicity in the minimum polynomial so that Krylov subspaces can (in exact arith-

metic) capture the exact solution to a linear program with this preconditioned system matrix within three steps.

The note [61] is extended to the case with nonzero lower entry C_3 by Ipsen [47]. It is shown that the preconditioner (6.1.8), with the Schur complement S appropriately modified to $S = C_3 - C_2G^{-1}C_1^T$, can be derived from a (scaled block) LU–decomposition of the system.

The proposed preconditioner can be of use when inexpensive approximations of G and of the (negative) Schur complement S are available. This might be the case in some applications. Then it is still to check how the preconditioner performs if it is not used in its above ideal version, but with approximate entries. No experiences are related in the cited papers [47], [61]. Also, the assumption that G be nonsingular excludes the use of the proposed preconditioner for numerous applications arising in optimization. There, $C_1 = C_2$ is common, and often $C_3 = 0$. It is often assumed that G be positive definite on the null space of the constraints, but considerably less often that G be positive definite on the entire space. In our application, for instance, see Chapter 7, a boundary control problem, the upper left block is not invertible. Nevertheless, interesting insight is provided by the notes [47], [61]. Compare our analysis for the special case that we investigate for our preconditioner in Lemma 5.4.4.

6.2 Three Positive Definite Preconditioners

We consider three symmetric positive definite preconditioners in this section. These preconditioners were derived and analyzed in [3] and [5]. The preconditioners are taken up again in [6], where new results on the eigenvalue distribution of the ideally preconditioned systems accompany the numerical comparison to the indefinite preconditioner $\tilde{K}$ of Chapter 5.

We refer to the preconditioners as P_1, P_2, and P_3 like it is done in these works. The three preconditioners are composed of submatrices, in the general case of preconditioners for the submatrices only, and in this sense closely related to the preconditioner $\tilde{K}$. These preconditioners are designed to work on systems of the structure

$$K = \begin{pmatrix} L_{\phi\phi} & 0 & A^T \\ 0 & L_{uu} & B^T \\ A & B & 0 \end{pmatrix}. \tag{6.2.1}$$

We are in this work effectively interested in the case

$$K = \begin{pmatrix} L_{\phi\phi} & L_{\phi u} & A^T \\ L_{u\phi} & L_{uu} & B^T \\ A & B & 0 \end{pmatrix}, \tag{6.2.2}$$

however. One of the preconditioners considered in this section, P_3, can be extended to the case with nonzero mixed entries $L_{\phi u}$, $L_{u\phi}$ in a straightforward way. The other

two, P_1 and P_2, are considered in their original versions. We also investigate the consequences of nonzero mixed entries $L_{\phi u}$, $L_{u\phi}$.

The blocks in (6.2.1) and (6.2.2) are taken to obey Assumption 5.2.1. In addition, we assume throughout the sections dealing with P_1 and P_2 that $L_{\phi\phi}$ and L_{uu} be symmetric positive definite. Should that not be true, adaptations can be thought of. (It is not true for the boundary control problem that we consider in Chapter 7. There, $L_{\phi\phi}$ is only symmetric positive semidefinite. But $L_{\phi\phi}$ is very simple in that application; it is a scaled identity in its nonsingular part. Thus, its preconditioner is taken to be $L_{\phi\phi}$ with the zeros on the diagonal changed to ones.) We repeat the dimensioning of the quantities involved, Equation (5.2.6),

$$L_{\phi\phi} \in I\!R^{m\times m},\ L_{uu} \in I\!R^{k\times k},\ L_{\phi u}^T = L_{u\phi} \in I\!R^{k\times m},\ A \in I\!R^{m\times m},\ B \in I\!R^{m\times k}\,.$$

As was defined in (1.1.19), the reduced Hessian H corresponding to the full system in (6.2.1) is given by

$$H = B^T A^{-T} L_{\phi\phi} A^{-1} B - L_{u\phi} A^{-1} B - B^T A^{-T} L_{\phi u} + L_{uu}\,.$$

Obviously, the reduced Hessian that corresponds to the system (6.2.2) is

$$H = B^T A^{-T} L_{\phi\phi} A^{-1} B + L_{uu}\,.$$

6.2.1 The First Preconditioner

The first preconditioner is given by its inverse $(P_1^*)^{-1}$ as

$$(P_1^*)^{-1} = \begin{pmatrix} L_{\phi\phi}^{-1/2} & 0 & 0 \\ 0 & L_{uu}^{-1/2} & 0 \\ 0 & 0 & L_{\phi\phi}^{1/2} A^{-1} \end{pmatrix}.$$

This leads in the case of blockdiagonal upper left part described in (6.2.1) to the preconditioned system

$$\begin{pmatrix} I_m & 0 & I_m \\ 0 & I_n & L_{uu}^{-1/2} B^T A^{-T} L_{\phi\phi}^{1/2} \\ I_m & L_{\phi\phi}^{1/2} A^{-1} B L_{uu}^{-1/2} & 0 \end{pmatrix}.$$

This system has at most $2k+3$ distinct eigenvalues, [6, Th. 2.3]. In exact arithmetic, this constitutes an upper limit to the number of steps MINRES may take. Compare Section 5.1.

In the case of nonzero mixed entries, (6.2.2) is preconditioned to

$$\begin{pmatrix} I_m & L_{\phi\phi}^{-1/2} L_{\phi u} L_{uu}^{-1/2} & I_m \\ L_{uu}^{-1/2} L_{u\phi} L_{\phi\phi}^{-1/2} & I_n & L_{uu}^{-1/2} B^T A^{-T} L_{\phi\phi}^{1/2} \\ I_m & L_{\phi\phi}^{1/2} A^{-1} B L_{uu}^{-1/2} & 0 \end{pmatrix}.$$

In general, $L_{\phi\phi}^{-1/2}, L_{uu}^{-1/2}$, and A^{-1}, A^{-T} can not be computed exactly. To derive a practicable preconditioner, it is assumed that preconditioners P_ϕ and P_u of $L_{\phi\phi}$ and L_{uu}, respectively, are available and that an approximate inverse P_A^{-1} of A is known. This leads to the first preconditioner in a general form which is given by

$$P_1^{-1} = \begin{pmatrix} P_\phi^{-1} & 0 & 0 \\ 0 & P_u^{-1} & 0 \\ 0 & 0 & P_\phi^T P_A^{-1} \end{pmatrix}.$$

The preconditioned KKT matrix is then for the case (6.2.2) given by

$$\begin{pmatrix} P_\phi^{-1} L_{\phi\phi} P_\phi^{-T} & P_\phi^{-1} L_{\phi u} P_u^{-1} & P_\phi^{-1} P_A^{-T} A^T P_\phi \\ P_u^{-1} L_{u\phi} P_\phi^{-1} & P_u^{-1} L_{uu} P_u^{-T} & P_u^{-1} B^T P_A^{-T} P_\phi \\ P_\phi^T P_A^{-1} A P_\phi^{-T} & P_\phi^T P_A^{-1} B P_u^{-T} & 0 \end{pmatrix}.$$

Benefits can be expected from the preconditioner if the singular values of the matrix

$$G = L_{\phi\phi}^{1/2} A^{-1} B L_{uu}^{-1/2} \tag{6.2.3}$$

in the ideal case, or of its approximate counterpart $\tilde{G} = P_\phi^T P_A^{-1} B P_u^{-T}$ in the general case, are of moderate size. This can be shown to hold under certain assumptions on the submatrices of K. See [5].

Application of the First Preconditioner The application of the preconditioner P_1 can be done as follows. Consider a vector $z = (z_1, z_2, z_3)^T$ with entries $z_1 \in \mathbb{R}^m$, $z_2 \in \mathbb{R}^k$, $z_3 \in \mathbb{R}^m$, and denote $x = (x_1, x_2, x_3)^T$ with entries $x_1 \in \mathbb{R}^m$, $x_2 \in \mathbb{R}^k$, $x_3 \in \mathbb{R}^m$. The transformed vector $x = P_1^{-1} z$ can be computed by solving the linear systems

$$x_1 = P_\phi^{-1} z_1\,, \quad x_2 = P_u^{-1} z_2,\,, \quad x_3 = P_\phi^T P_A^{-1} z_3.$$

Likewise, $w = P_1^{-T} x$, where $w = (w_1, w_2, w_3)^T$ with entries $w_1 \in \mathbb{R}^m$, $w_2 \in \mathbb{R}^k$, $w_3 \in \mathbb{R}^m$, can be computed by solving the linear systems

$$w_1 = P_\phi^{-T} x_1\,, \quad w_2 = P_u^{-T} x_2,\,, \quad w_3 = P_A^{-T} P_\phi^T x_3.$$

Of course, we never compute the inverses of matrices, but solve the corresponding systems.

6.2.2 The Second Preconditioner

The second preconditioner P_2 is in the ideal version given by its inverse as

$$(P_2^*)^{-1} = \begin{pmatrix} L_{\phi\phi}^{-1/2} & 0 & 0 \\ 0 & L_{uu}^{-1/2} & 0 \\ -L_{\phi\phi}^{-1/2} & -L_{\phi\phi}^{1/2} A^{-1} B L_{uu}^{-1} & L_{\phi\phi}^{1/2} A^{-1} \end{pmatrix}$$

and transforms system (6.2.1) into the blockdiagonal matrix

$$\begin{pmatrix} I_m & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & -(I_m + GG^T) \end{pmatrix},$$

where G on the lower diagonal is given in (6.2.3). As was mentioned for P_1, benefits can be expected if the singular values of G are of moderate size. In the case of the ideal preconditioner P_2^*, the preconditioned system has at most $k+2$ distinct eigenvalues, [6, Th. 2.4].

A general version of this preconditioner would read

$$\begin{pmatrix} P_\phi^{-1} & 0 & 0 \\ 0 & P_u^{-1} & 0 \\ -P_\phi^{-1} & -P_\phi P_A^{-1} B P_u^{-2} & P_\phi P_A^{-1} \end{pmatrix}.$$

For sake of a brief presentation we will not further pursue this general form with preconditioners for the submatrices. Even for K in (6.2.1) with blockdiagonal upper part, the generally preconditioned system is not blockdiagonal any more. Nonzero off–diagonal entries in the preconditioned system have a contaminating effect on the eigenvalue distribution compared to the ideally preconditioned case. Note also that the application of the second preconditioner in its ideal version to the system K in (6.2.2), i.e., with nonzero $L_{u\phi}$, $L_{\phi u}$, already leads to the full system

$$\begin{pmatrix} I_m & K_{2,12} & K_{2,13} \\ K_{2,12}^T & I_k & K_{2,23} \\ K_{2,13}^T & K_{2,23}^T & K_{2,33} \end{pmatrix}$$

with G from (6.2.3) in the entries

$$K_{2,12} = L_{\phi\phi}^{-1/2} L_{\phi u} L_{uu}^{-1/2}, \quad K_{2,13} = -L_{\phi\phi}^{-1/2} L_{\phi u} G^T, \quad K_{2,23} = -K_{2,12}^T,$$
$$K_{2,33} = -(I_m + (GG^T)) + G\, K_{2,12}^T + K_{2,12}\, G^T.$$

Application of the Second Preconditioner The application of the preconditioner P_2 can be done as follows. Denote by z the vector $z = (z_1, z_2, z_3)^T$ with $z_1 \in \mathbb{R}^m$, $z_2 \in \mathbb{R}^k$, $z_3 \in \mathbb{R}^m$, let $x = (x_1, x_2, x_3)^T$ with $x_1 \in \mathbb{R}^m$, $x_2 \in \mathbb{R}^k$, $x_3 \in \mathbb{R}^m$. The transformed vector $x = P_2^{-1} z$ can be computed as follows.

$$x_1 = P_\phi^{-1} z_1, \quad x_2 = P_u^{-1} z_2, \quad x_3 = P_\phi P_A^{-1}\Big(z_3 - B\, P_u^{-2} x_2\Big) - x_1.$$

We can compute $w = P_2^{-T} x$ for $w = (w_1, w_2, w_3)^T$ with $w_1 \in \mathbb{R}^m$, $w_2 \in \mathbb{R}^k$, and $w_3 \in \mathbb{R}^m$, by solving the linear systems

$$w_1 = P_\phi^{-T}\Big(x_1 - x_3\Big), \quad w_3 = P_A^{-T} P_\phi^T x_3, \quad w_2 = P_u^{-T}\Big(x_2 - P_u^{-T} B^T w_3\Big).$$

Assuming that the preconditioners for L_{uu} and $L_{\phi\phi}$ can be applied efficiently and that, therefore, the cost of solving with P_A and its transpose dominates the other computations, we can see that the application of P_2^{-1} is essentially not costlier than the application of P_1^{-1}.

6.2.3 The Third Preconditioner

A third preconditioner has been considered in [5], essentially transforming the spectrum of K into the spectrum of the reduced Hessian H of the underlying quadratic programming problem. Compare Section 5.2.1 for the corresponding transformations, specifically Lemma 5.2.2. This preconditioner was designed for blockdiagonal upper part in K just as P_1 and P_2, compare [5], but has a straightforward extension to the case of nonzero $L_{u\phi}$, $L_{\phi u}$. The preconditioner P_3 is, with this additional modification, given in its ideal version by its inverse as

$$(P_3^*)^{-1} = \begin{pmatrix} -B^T A^{-T} & I_k & (B^T A^{-T} L_{\phi\phi} - L_{u\phi}) A^{-1} \\ 0 & 0 & A^{-1} \\ I_m & 0 & -1/2\, L_{\phi\phi} A^{-1} \end{pmatrix} \tag{6.2.4}$$

and transforms K in (6.2.2) such that the blockdiagonal system

$$\begin{pmatrix} H & 0 & 0 \\ 0 & 0 & I_m \\ 0 & I_m & 0 \end{pmatrix} \tag{6.2.5}$$

is obtained. The eigenvalues of this preconditioned system are m 1's, m -1's, and the eigenvalues of H, [3, Section 6.4.2], [6, Th. 2.4]. A general version P_3 of (6.2.4) is

$$\begin{pmatrix} -B^T P_A^{-T} & I_k & (B^T P_A^{-T} L_{\phi\phi} - L_{u\phi}) P_A^{-1} \\ 0 & 0 & P_A^{-1} \\ I_m & 0 & -1/2\, L_{\phi\phi}\, P_A^{-1} \end{pmatrix}.$$

Additional preconditioning of the preconditioned system (6.2.5) involving the reduced Hessian H is possible. Obviously, when $M_H = P_H P_H{}^T$ is a preconditioner for H, one can transform (6.2.5) into the equivalent system

$$\begin{pmatrix} P_H^{-1} H P_H^{-T} & 0 & 0 \\ 0 & 0 & I_m \\ 0 & I_m & 0 \end{pmatrix}.$$

See [63] for a discussion of how to precondition the reduced Hessian and [18] for numerical experiments based on this. Compare also [11], where the additional preconditioning step is inspired by a quasi–Newton framework.

Application of the Third Preconditioner The application of the preconditioner P_3 can be done in the following way. Note that the vector x is partitioned differently from z and w. Let $z = (z_1, z_2, z_3)^T$ with $z_1 \in I\!R^m$, $z_2 \in I\!R^k$, $z_3 \in I\!R^m$, let $x = (x_1, x_2, x_3)^T$ with $x_1 \in I\!R^k, x_2 \in I\!R^m, x_3 \in I\!R^m$. The transformed vector $x = P_3^{-1} z$ can be computed as follows,

$$x_2 = P_A^{-1} z_3\,, \quad x_3 = z_1 - \frac{1}{2}\, L_{\phi\phi} x_2\,, \quad x_1 = z_2 + B^T P_A^{-T}\Big(L_{\phi\phi} x_2 - z_1\Big) - L_{u\phi}\, x_2.$$

Using one additional array t in the implementation, we need to compute the product with $L_{\phi\phi}$ only once. After solving for x_2 as before, set $t = L_{\phi\phi}\, x_2$ to obtain

$$x_1 = z_1 - \frac{1}{2}\, t\,, \quad x_3 = z_2 + B^T P_A^{-T}\big(t - z_1\big) - L_{u\phi}\, x_2.$$

Since the components in z_3 are no longer needed after solving the system $z_3 = P_A x_2$, an additional array t is not really needed; we can overwrite z_3 with $L_{\phi\phi} P_A^{-1} z_3$.

The application of the transpose of the third preconditioner can be done in a similar way. We can compute $w = P_3^{-T} x$, where $w = (w_1, w_2, w_3)^T$ with $w_1 \in I\!R^m$, $w_2 \in I\!R^k$, $w_3 \in I\!R^m$, by setting $w_2 = x_1$ and then solving the linear systems

$$t = P_A^{-1} B\, x_1\,, \quad w_1 = x_3 - t\,, \quad w_3 = P_A^{-T}\big(x_2 + L_{\phi\phi}(t - \frac{1}{2}\, x_3) - L_{\phi u} x_1\big).$$

In this case an additional array t for the implementation *is* necessary.

Note that a single application of this preconditioner requires essentially the double amount of work of the preconditioners P_1, P_2, and $\tilde{K}$. This is due to the fact that the application of P_3^{-1} involves both A^{-1} and A^{-T}, and that the same is true for its transpose P_3^{-T}. Assuming that the cost of solving with A or with its preconditioner P_A dominates the computations, we can see that the cost of the application of P_3 is essentially twice the cost of applying the other preconditioners that have been considered. This is true in the case where each solve with A or its approximation and A^T requires the call of a routine (or an entire software package) and where this routine dominates the computations. This case must often be assumed for practical applications. If, however, it is such that matrices are set up and a decomposition of A is computed, the situation is changed. Then the (expensive) decomposition is done once, and the ensuing solves with the factors of A will in general not dominate. This is what we see in the numerical results for our application in Section 7.3, where this disadvantage of P_3 is not very pronounced.

From the above presentation of the solve with the preconditioner $M_3 = P_3 P_3^T$ one already concludes that two solves with P_A and two solves with its its transpose are the expense one has to pay. This is confirmed by the matrix representation of $M_3^{-1} = (P_3^*)^{-T}(P_3^*)^{-1}$,

$$M_3^{-1} = \begin{pmatrix} DD^T & D & M_{3,13} \\ D^T & 0 & M_{3,23} \\ M_{3,13}^T & M_{3,23}^T & M_{3,33} \end{pmatrix} + \begin{pmatrix} I_m & 0 & 0 \\ 0 & I_k & 0 \\ 0 & 0 & I_m \end{pmatrix},$$

with entries

$$\begin{aligned} M_{3,13} &= -(D(D^T L_{\phi\phi} - L_{u\phi}) + 1/2\, L_{\phi\phi})A^{-1}, \\ M_{3,23} &= (D^T L_{\phi\phi} - L_{u\phi})A^{-1}, \\ M_{3,33} &= A^{-T}\Big((L_{\phi\phi} D - L_{\phi u})(D^T L_{\phi\phi} - L_{u\phi}) + 1/4 L_{\phi\phi}^2\Big)A^{-1}, \end{aligned}$$

where $D = A^{-1}B$ as before. This representation of M_3 shows that the cost of M_3 can never be less than two solves with P_A (or A) and two solves with P_A^T (or A^T) because

in the entry $M_{3,33}$ these four factors are present. When $z = (z_1, z_2, z_3)^T$ with $z_1 \in I\!R^m$, $z_2 \in I\!R^k$, $z_3 \in I\!R^m$, and $w = (w_1, w_2, w_3)^T$ with $w_1 \in I\!R^m$, $w_2 \in I\!R^k$, $w_3 \in I\!R^m$, then the solve of $M_3\, w = z$ for w will require that these four solves are performed sequentially, starting with w_3.

6.3 The Constraint Preconditioner

An indefinite preconditioner for KKT systems has been suggested by Keller, Gould and Wathen in [49]. This preconditioner is termed "constraint preconditioner" in [49] because the constraints are retained without modification in the preconditioner.

The situation in [49] is similar to the situation underlying Chapter 5. See Chapter 1. For the quadratic programming problem

$$\min q(x) = \frac{1}{2} x^T G x + g^T x \tag{6.3.1}$$

subject to

$$Cx = b, \tag{6.3.2}$$

the associated KKT conditions

$$\begin{pmatrix} G & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix} = \begin{pmatrix} -g \\ b \end{pmatrix} \tag{6.3.3}$$

have to be solved, i.e., a linear system

$$\mathcal{G} s = r, \tag{6.3.4}$$

with a system matrix $\mathcal{G}$ of the structure given in (6.3.3). The dimensioning of the quantities involved is similar to that in Section 5.2. See specifically (5.2.6). Since we are effectively considering a (2×2)–block matrix here instead of the (3×3)–system of Section 5.2, $m + k$ is abbreviated as n in this section,

$$G \in I\!R^{n \times n},\ C \in I\!R^{m \times n},\ x \in I\!R^n,\ g \in I\!R^n.$$

Note that thus $k = n - m$. It is assumed that k is a nonnegative integer, so that $n \geq m$.

In analogy to Assumption 5.2.1 it is throughout this section required that the following holds for the blocks of (6.3.3). These are the assumptions of [49].

Assumption 6.3.1 *For the submatrices of $\mathcal{G}$ in* (6.3.3) *let be valid:*

1. *$G \in I\!R^{n \times n}$ is symmetric.*

2. *$C \in I\!R^{m \times n}$ has full rank.*

The preconditioner suggested in [49] for linear systems (6.3.3) is

$$\mathcal{M} = \begin{pmatrix} M & C^T \\ C & 0 \end{pmatrix}, \tag{6.3.5}$$

where C is as above, and the symmetric matrix M is supposed to approximate G. Note that the constraints C of the original system (6.3.3) are preserved in the preconditioner without changes. This corresponds to the choice $P_A = A$ in our preconditioner $\tilde{K}$ which makes our treatment more general in that respect. We have seen in Chapter 5 that major difficulties arise when the preconditioner is only approximate ($P_A \approx A$). Our results in Theorem 5.3.1 are recovered for the constraint preconditioner. See below.

The blanket assumption for M is only symmetry. For most results of interest in [49], however, it is necessary that M be symmetric positive definite on the null space of the constraints. This is the symmetric positive definiteness of the reduced Hessian. Eigenvalue bounds for the preconditioned system are stated effectively in [49] only for the case where M is symmetric positive definite even on the entire space $\mathbb{R}^n$.

The structure of $\mathcal{M}$ in (6.3.5) is near to that of our preconditioner $\tilde{K}$. In considering (3×3)–block systems instead of 2×2, we allow for more degrees of freedom. In the notation of [49], our subpreconditioner M necessarily has the structure

$$M = \begin{pmatrix} 0 & 0 \\ 0 & P_H \end{pmatrix}. \tag{6.3.6}$$

This appears less general at first sight. However, we consider different choices for P_H, e.g. $P_H = H$ and $P_H = L_{uu}$. Compare Section 5.3.3. The choices considered in that section are more appropriate for some settings than $M = \mathrm{diag}(G)$ or $M = I_n$, these being the only choices which are treated in [49]. If G is diagonally dominant, for instance, $M = \mathrm{diag}(G)$ might already give an excellent preconditioner. Likewise, $M = I_n$ can be the source of significant improvement within the preconditioned solve, merely due to the fact that at most $k + 1$ distinct eigenvalues (in exact arithmetic, of course) are left. However, there are situations, e.g. when $L_{\phi u}$ is not negligible, where these choices are not good enough, motivating us to pursue different options in our preconditioner. Also, using a preconditioner of the structure (6.3.6) is the key to the efficient implementation of $\tilde{K}$. Compare Section 5.2.3. The application of the constraint preconditioner requires different techniques. In terms of the iterative method, the work [49] has precursors in the studies [20] and [42]. The so called conjugate gradient iteration on the reduced system is derived below.

Discussion of the Constraint Preconditioner

We use this section to go through results for the eigenvalue distribution of the preconditioned system $\mathcal{M}^{-1}\mathcal{G}$ with $\mathcal{G}$ in (6.3.3) obeying Assumption 6.3.1 and $\mathcal{M}$ given in (6.3.5). These results in [49] have been derived with an orthogonal decomposition QR of the null space of the constraints. This factorization is not being used in actual computations, however.

Let

$$QR = (Y \mid Z) \begin{pmatrix} R \\ 0 \end{pmatrix}$$

be an orthogonal factorization of C^T, where $R \in \mathbb{R}^{m \times m}$ is upper triangular, where $Y \in \mathbb{R}^{n \times m}$, and where $Z \in \mathbb{R}^{n \times k}$ is a basis for the null space $\mathcal{N}(C)$ of C. Thus, $CZ = 0$ and $Y^T Y = I_k$, $YY^T = ZZ^T = I_n$, $Z^T Z = I_m$, $Z^T Y = 0$. Also, $R^T = CY$. With the help of such an orthogonal factorization the following is observed in [49], an analogue to Theorem 5.3.1.

Theorem 6.3.2 (Th. 2.1 in [49]) *Let $\mathcal{G}$ obey Assumption* 6.3.1*, let Z be a basis for the null space of the constraints. If $\mathcal{G}$ is preconditioned by $\mathcal{M}$ in* (6.3.5) *with M symmetric, then the preconditioned system $\mathcal{M}^{-1}\mathcal{G}$ has an eigenvalue* 1 *with multiplicity* $2m$*, and the remaining eigenvalues are given by the solution to the generalized eigenvalue problem $Z^T GZx = \lambda Z^T MZx$.*

Keller, Gould and Wathen [49] also find that these remaining k eigenvalues are real in case that either the reduced Hessian, $Z^T GZ$, or the projection of the preconditioner onto the null space of the constraints, $Z^T MZ$, or both, are symmetric positive definite. This is in analogy to Lemma 5.3.2. Due to the special structure (6.3.6) of the subpreconditioner in the upper left of $\tilde{K}$, we require in Lemma 5.3.2 only that P_H be symmetric positive definite. This is natural in our setting. In general, we work with the null space representation

$$W = \begin{pmatrix} -A^{-1}B \\ I_k \end{pmatrix},$$

compare (1.1.18), that is canonical for problems of optimal control. With this representation, the requirement that $W^T MW$ symmetric positive definite for M in (6.3.6) is equivalent to P_H being symmetric positive definite.

These results were achieved under quite general assumptions. Eigenvalue bounds, however, are only given for the case where M is not only symmetric positive definite on the null space of the constraints, but on the entire $\mathbb{R}^n$.

Consequentially, convergence of iterative methods on the preconditioned system is scrutinized in the work [49]. Statements are expressed with the help of Krylov subspaces

$$\mathcal{K}(\mathcal{M}^{-1}\mathcal{G}, \mathcal{M}^{-1}r_0) \tag{6.3.7}$$

generated by the preconditioned system $\mathcal{M}^{-1}\mathcal{G}$ and the initial preconditioned residual $\mathcal{M}^{-1}r_0$, where $r_0 = r - \mathcal{G}s_0$ is the initial residual determined by the starting guess s_0. This is based on the original finding by Saad and Schultz [70], reformulated in Theorem 5.1.5. Compare Section 5.4. We now state the results in [49] and point out the analogies in our work.

Theorem 6.3.3 (Th. 3.2 in [49]) *Let Assumption* 6.3.1 *hold, let $m = n$. If $\mathcal{G}$ is preconditioned by $\mathcal{M}$, then the Krylov subspace* (6.3.7) *is of dimension at most* 2*.*

This is understood to hold for any right hand side r and initial guess s_0. The assertion is easily seen to be true upon realizing that with $m = n$, $k = 0$ in Theorem 5.4.1.

Theorem 6.3.4 (Th. 3.5 in [49]) *Let Assumption* 6.3.1 *hold, let $m < n$. Let Z be a basis for the null space of C, let M be symmetric and $Z^T MZ$ symmetric positive definite. If $\mathcal{G}$ is preconditioned by $\mathcal{M}$, then the Krylov subspace* (6.3.7) *is of dimension at most $k + 2$.*

This is the assertion of our Theorem 5.4.1, proven in [49] only under the additional assumption that M be symmetric and that $Z^T MZ$ be symmetric positive definite. This additional assumption is not necessary to get the desired result. Under this additional assumption, however, the convergence–governing system $(Z^T MZ)^{-1} Z^T GZ$ is normal, and it can be shown that the dimension of the Krylov subspace (6.3.7) is effectively determined by the number of distinct eigenvalues of $(Z^T MZ)^{-1} Z^T GZ$. See Theorem 5.4.3. This is not yet true without the assumption that $Z^T GZ$ or $Z^T MZ$ or both be symmetric positive definite. Thus, Theorem 3.7 in [49] which lacks this assumption has to be complemented with it. It is shown in the following example that $Z^T MZ$ must be symmetric positive definite in order to determine the dimension of (6.3.7) via the number of distinct eigenvalues of $(Z^T MZ)^{-1} Z^T GZ$.

With the admissible choices $n = 4$, $m = 1$,

$$G = \begin{pmatrix} 42 & -42 & 35 & 0 \\ -42 & 39 & -30 & 0 \\ 35 & -30 & 20 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \quad C^T = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \tag{6.3.8}$$

$$M = \begin{pmatrix} 7 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{and } Z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \tag{6.3.9}$$

convergence of GMRES on the preconditioned system $\mathcal{M}^{-1}\mathcal{G}$ is governed by the eigenvalues of the matrix

$$S = (Z^T MZ)^{-1}(Z^T GZ) = \begin{pmatrix} 6 & -6 & 5 \\ 14 & -13 & 10 \\ 7 & -6 & 4 \end{pmatrix}. \tag{6.3.10}$$

The matrix S has the single eigenvalue $\lambda = -1$ with multiplicity $\mu = 3$. Theorem 3.7 in [49] asserts that the dimension of the Krylov subspace $\mathcal{K}(\mathcal{M}^{-1}\mathcal{G}, b)$ is at most 3, which is the number of distinct eigenvalues of S, i.e., 1, augmented by 2. One finds, however, that with $\mathcal{M}^{-1} r_0 = b = (1, 1, 1, 1, 1)^T$, the associated Krylov subspace of order 4,

$$\mathcal{K}_4(\mathcal{M}^{-1}\mathcal{G}, b) = \left\{ b, (\mathcal{M}^{-1}\mathcal{G})b, (\mathcal{M}^{-1}\mathcal{G})^2 b, (\mathcal{M}^{-1}\mathcal{G})^3 b \right\}, \tag{6.3.11}$$

is spanned by b and $(5, 11, 5, 1, -1)^T$, $(-11, -23, -11, 1, -3)^T$, $(17, 35, 17, 1, -5)^T$, and has dimension 4.

Preconditioned Conjugate Gradients on the Reduced System

Upon use of the constraint preconditioner, one has to deal with

$$\mathcal{M}^{-1}\mathcal{G}s = \mathcal{M}^{-1}r, \tag{6.3.12}$$

where $\mathcal{G}$ is given in (6.3.3) and obeys Assumption 6.3.1, and where the preconditioner $\mathcal{M}$ with M symmetric, $M \approx G$, is defined in (6.3.5). Since both the original system $\mathcal{G}$ and the preconditioner $\mathcal{M}$ are symmetric indefinite, it is not obvious that there is an iterative solution method other than those designed for general systems, e.g. GMRES, to solve the preconditioned linear system. Interestingly, the solution of the preconditioned linear system can be done with a conjugate gradient routine. The treatment in [42] of the so called conjugate gradient iteration on the reduced system is the basis for the work in [49]. Elements of the work in [42] can already be found in [20] and [21]. It will be subsequently pointed out what the focus of the works [20] and [21] is. We now go through the derivation of the conjugate gradient iteration on the reduced system, recalling that the goal is to solve the quadratic programming problem (6.3.1) s.t. (6.3.2). This can be accomplished via the solution of the associated KKT system (6.3.3).

Let W denote a matrix that spans the null space $\mathcal{N}(C)$ of the constraints C, i.e., W satisfies $CW = 0$. Thus, the columns of C^T together with the columns of W span the entire $\mathbb{R}^{n\times n}$. It is well known that both

$$W(W^TW)^{-1}W^T, \quad I_n - C^T(CC^T)^{-1}C \tag{6.3.13}$$

are projections onto the null space of C. The representation (1.1.18) for the null space of C is often encountered in problems of optimal control, where a partitioning of the variables into state and control variables is natural. Other choices for the null space representation are possible, compare [42]. A common strategy is an orthonormal factorization, requiring the computation of (sparse) LQ factors of C. We will not explore these issues in depth. We mean W in this section to be some null space representation which is not necessarily the natural one for control problems.

Employing a null space representation W, any solution x^* to (6.3.2) can be written as the sum of normal component $C^Tx_C^*$ and tangential component Wx_W^*,

$$x^* = C^Tx_C^* + Wx_W^* \tag{6.3.14}$$

for some $x_C^* \in \mathbb{R}^m$, $x_W^* \in \mathbb{R}^k$. The constraints (6.3.2) yield

$$CC^Tx_C^* + CWx_W^* = CC^Tx_C^* = b,$$

and these normal equations determine x_C^*. Substituting x^* into the objective function (6.3.1), omitting the now constant term x_C^*, the remaining x_W^* solves the reduced problem

$$\min_{x_W} \frac{1}{2}x_W^THx_W + g_W^Tx_W. \tag{6.3.15}$$

In (6.3.15), $H \in \mathbb{R}^{k\times k}$ is the reduced Hessian, and $g_W \in \mathbb{R}^k$ is the reduced gradient,

$$H = W^T G W, \quad g_W = W^T(GC^T x_C^* - g).$$

Assuming that G of the quadratic program (6.3.1) is positive definite on the null space of the constraints, i.e., that H is positive definite, (6.3.15) is equivalent to

$$Hx_W = -g_W. \tag{6.3.16}$$

Figure 6.3.1: Preconditioned conjugate gradients on reduced system.

Preconditioned conjugate gradients on reduced system.

---

applied for the solution of the quadratic program (6.3.1) s.t. (6.3.2) via the reduced system (6.3.16).

Preprocessing Step:
Choose initial point x_W.
Compute initial reduced residual $r_W = Hx_W + g_W$.
Compute initial preconditioned residual as solution of $P_H y_W = r_W$.
Set $v_W = -y_W$.
Set termination tolerance `tol`.

Algorithm:
WHILE $r_W^T y_W \geq$ `tol` DO
Compute $\alpha = \frac{r_W^T y_W}{v_W^T H v_W}$.
Set $x_W \leftarrow x_W + \alpha v_W$.
Compute $r_W^+ = r_W + \alpha H v_W$.
Solve $P_H y_W^+ = r_W^+$.
Compute $\beta = \frac{(r_W^+)^T y_W^+}{r_W^T y_W}$.
Set $v_W \leftarrow -y_W^+ + \beta v_W$.
Set $y_W \leftarrow y_W^+$.
Set $r_W \leftarrow r_W^+$.
END WHILE

Postprocessing Step:
Compute Wx_W.
Set $x = C^T x_C + W x_W$.

The linear system (6.3.16) can be solved with the conjugate gradient routine. Substitution of the solution into (6.3.14) yields an (approximate) solution of the quadratic program (6.3.1) s.t. (6.3.2). This strategy corresponds to computing the normal component $C^T x_C^*$ of the solution x^* exactly and the tangential component $W x_W^*$ approximately.

The conjugate gradient routine for the linear system (6.3.16) is called conjugate gradient routine for reduced systems or projected conjugate gradient routine. This name stems from the fact that the iteration can be performed fully with the "reduced" quantities, i.e., fully on the projection space $\mathcal{N}(C)$. Often, the practical application of the conjugate gradient routine requires the use of a preconditioner. We therefore assume at this point that a preconditioner $P_H \in \mathbb{R}^{k \times k}$ for the reduced Hessian H is available. In order to be applicable within the conjugate gradient routine, P_H must be symmetric positive definite. The algorithm in its preconditioned form is given in Figure 6.3.1. The original version is recovered with the choice $P_H = I_k$.

The conjugate gradient routine can also be written in the so called expanded form. Compare Figure 6.3.2. In that case, the original quantities are used within the iteration, e.g. g and G instead of the reduced gradient g_W and the reduced Hessian H. Straightforward rewriting of the quantities in that routine leads to a preconditioner $W P_H^{-1} W^T \in \mathbb{R}^{n \times n}$ for the full residual r instead of P_H^{-1} for the reduced residual r_W.

The discussion of the preconditioner P_H for the reduced Hessian $H = W^T G W$ in [42] assumes that the preconditioner is formed as

$$P_H^{-1} = (W^T M W)^{-1}$$

with M chosen s.t. the product $W^T M W$ is symmetric positive definite. This preconditioner P_H^{-1} is supposed to approximate the ideal preconditioner H^{-1} whose usage is typically out of question due to cost considerations.

The choice of a preconditioner for reduced systems is discussed in detail from a numerical linear algebra point of view in [63]. In such way, the choices $M = I_k$ and $M = \mathrm{diag}(G)$ are addressed in [42]. Even with these simple choices, due to the need of the null space basis W, the application of the preconditioner can easily dominate overall computational cost. The preconditioner is applied once in each step of the conjugate gradient routine. In the notation of Figure 6.3.2, in each iteration the preconditioned residual y^+ has to be computed as

$$y^+ = W P_H^{-1} W^T r^+. \tag{6.3.17}$$

It is stated in [42] that the conjugate gradient method for reduced systems can be considered an effective method for computing the solution to the quadratic programming problem (6.3.1) s.t. (6.3.2). The need for a null space matrix W, however, constitutes a main drawback. An alternative is derived in [42] to avoid this null space representation.

In the case $G = I_k$, the preconditioned residual y^+ in (6.3.17) is given as, compare (6.3.13),

$$y^+ = W(W^T W)^{-1} W^T r^+ = (I_n - C^T (C C^T)^{-1} C) r^+. \tag{6.3.18}$$

Figure 6.3.2: Preconditioned conjugate gradients on reduced system. Expanded form.

Preconditioned conjugate gradients on reduced system. Expanded form.

---

applied for the solution of the quadratic program (6.3.1) s.t. (6.3.2).

Preprocessing Step:
Choose initial point x that satisfies the constraints $Cx = b$.
Compute initial residual $r = Gx - g$.
Compute initial preconditioned residual $y = W^T P_H^{-1} W r$.
Set $v = -y$.
Set termination tolerance `tol`.

Algorithm:
WHILE $r^T y \geq$ `tol` DO
Compute $\alpha = \frac{r^T y}{v^T G v}$.
Set $x \leftarrow x + \alpha v$.
Compute $r^+ = r + \alpha G v$.
Compute $y^+ = W P_H^{-1} W^T r^+$.
Compute $\beta = \frac{(r^+)^T y^+}{r^T y}$.
Set $v \leftarrow -y^+ + \beta v$.
Set $y \leftarrow y^+$.
Set $r \leftarrow r^+$.
END WHILE

This can be expressed as

$$y^+ = r^+ - C^T z^+, \tag{6.3.19}$$

where the vector z^+ is the solution of

$$CC^T z^+ = Cr^+. \tag{6.3.20}$$

Noting that (6.3.20) are normal equations, it follows that z^+ is the solution of the least squares problem

$$\min_z \|r^+ - C^T z^+\|,$$

and that the desired residual y^+ is the corresponding residual. This approach can be implemented using a Cholesky factorization of CC^T.

Equivalent to (6.3.19) in combination with (6.3.20), and without the use of a null space basis W, y^+ and the auxiliary quantity z^+ can be determined as the solution of the augmented system

$$\begin{pmatrix} I & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} y^+ \\ z^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{6.3.21}$$

If the preconditioner M is chosen s.t. M and $W^T MW$ are symmetric positive definite, then formula (6.3.18) for the preconditioned residual becomes

$$y^+ = W(W^T MW)^{-1} W^T r^+ = (M^{-1} - C^T (CMC^T)^{-1} C) r^+,$$

and the generalization of (6.3.21) is

$$\begin{pmatrix} M & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} y^+ \\ z^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{6.3.22}$$

Note that (6.3.22) does not make use of a null space representation W of $\mathcal{N}(C)$. However, the system (6.3.22) has to be solved in each step of the preconditioned conjugate gradient routine in order to perform the preconditioning step. This can be cost effective only if solves with the constraints $C = (A \mid B)$ are affordable and if M is simple. If the solve with M is not considerably simpler that the solve with the original G, one obviously has not gained much. For instance, if M is the ideal preconditioner, the preconditioned system $\mathcal{M}^{-1}\mathcal{G} = I$ has (in exact arithmetic) only the single desired eigenvalue 1. This eigenvalue has algebraic multiplicity $m + n$ and geometric multiplicity 1, and the iterative solver will terminate after one step. Unfortunately, the preconditioning step within the iterative solver requires the solution of a linear system with $M = G$, and nothing is won. This shows very nicely that preconditioning is no free lunch.

The computation of the preconditioned residual y^+ in Equation (6.3.17) via the system (6.3.22) can give rise to significant round–off error, particularly as the conjugate gradient iterates approach the solution. This is studied intensively in [42], and as a remedy iterative refinement techniques are proposed. These techniques lead to a residual update strategy for the augmented system approach. See [42] for details.

The approach in [42] — preconditioned conjugate gradients on the reduced system with safeguards to prevent significant numerical round–off — is the basis for the work by Keller, Gould and Wathen in [49]. The work [49] is concerned with the solution of systems (6.3.4) with a system matrix $\mathcal{G}$ of the structure (6.3.3) that obeys Assumption 6.3.1. The solution of these systems is done in [49] with the preconditioned conjugate gradient routine described in [42]. As was stated earlier, the computation of the preconditioned residual y^+ in (6.3.17), necessary in each iteration, is often the most expensive computational factor within the algorithm. Like it is suggested in [42], Keller, Gould and Wathen [49] avoid the explicit use of a null space representation W of $\mathcal{N}(C)$. Instead, they compute y^+ by means of a symmetric indefinite factorization of the system matrix of (6.3.22). This can be performed using the MA27 package of

the Harwell Subroutine Library. System (6.3.22) can often, as is stated in [49], be factored efficiently by using MA27 when M is a simple matrix block, whereas the direct application of the routine to the original system (6.3.3) may be limited by space requirements as well as time for large enough systems. As a way out, Keller, Gould and Wathen are investigating a new implementation approach based on an adaptation of MA27 such that an incomplete factorization of the upper left block M or G is possible.

It was already mentioned that parts of the work sketched above can also be found in [20] and [21]. Coleman [20] considers the application of preconditioned conjugate gradients, projected onto the null space of the constraints, in the setting of linearly constrained optimization. In this setting, especially in large dimensions, it might be prohibitive both to search for a null space basis and to explicitly form the reduced Hessian. This renders the search for a preconditioner difficult. See [63] for considerations of this problem. It is observed in [20] that the preconditioning step for the reduced system (6.3.16) can be performed via the solution of an augmented system (6.3.22) with M a preconditioner for G. Coleman presents three basic solution techniques for (6.3.22), the so called full space, range space, and null space approaches, each involving a different factorization of the system matrix in (6.3.22).

Coleman and Verma [21] further pursue the idea to use preconditioned conjugate gradients on the reduced system in the linearly constrained optimization setting. In order to facilitate the preconditioning step in (6.3.22), in their framework the factorization of the system matrix in (6.3.22), Coleman and Verma investigate numerically the consequences of using approximations to both G and C. This is performed by setting "small" matrix entries to zero, based on a dropping tolerance. It is observed in the numerical experiments that computational efficiency, space and time requirements all initially improve as the dropping tolerance is increased from zero up to an optimal point. From this point on overall computing time begins to rise because the quality of the approximation degrades. The optimal point was seen as dependent on the specific problem.

Chapter 7

A Ground Water Modeling Problem

In this chapter, we treat our second example problem which is motivated by problems of ground water modeling. We consider an optimal control problem that is governed by a partial differential equation. We derive the example problem and state the equations relevant to the approach, Section 7.1. In the subsequent section, Section 7.2, we sketch the discretization and then turn to the finite–dimensional solution of the optimization problem. The discretized problem is a quadratic programming problem with linear constraints. The finite–dimensional solution is found via the solution of the so called KKT system. The solution of this system is done with iterative solution methods. The Krylov subspace methods GMRES and MINRES are employed. However, the solution of the original system with these methods is highly impractical, which is why we turn to preconditioning. The numerical performance of the preconditioner $\tilde{K}$ analyzed in Chapter 5 is for a number of choices for the subpreconditioners tested on this example problem. It is additionally compared to the performance of the preconditioners of Section 6.2. See Section 7.3 for the assembled numerical results.

In the example problem, we assume homogeneous and isotropic conditions in the porous medium domain for which the continuum approach is employed. The domain is considered to be fully saturated. Flow is governed by Darcy's law and a continuity equation. For details see [8], [9]. A combination of these two equations with appropriate boundary conditions leads to the stationary partial differential equation that is the governing state equation of this problem. Compare Section 2.1 and the Appendix.

7.1 The Optimal Control Problem

We consider a boundary control problem defined on a rectangular domain. It occurs as a mixing problem when extracting ground water that originates from two different water bodies. The problem can be formulated as the linear quadratic problem

$$\begin{aligned}\min_{(\phi,u)} \quad & \Big\{ \tfrac{\eta_1}{2} \int_{\Gamma_2} u^2(x)\,dx + \eta_2 \int_{\Gamma_2} u(x)\,\phi(x,\bar{z})\,dx + \tfrac{\eta_3}{2} \int_{\Gamma_2} \phi^2(x,\bar{z})\,dx \\ & + \eta_4 \int_{\Gamma_6} \tfrac{\partial}{\partial n}\phi(0,z)\,dz + \eta_5 \int_{\Gamma_2} u(x)\,u_v\,dx \Big\}\end{aligned} \tag{7.1.1}$$

subject to the state equation

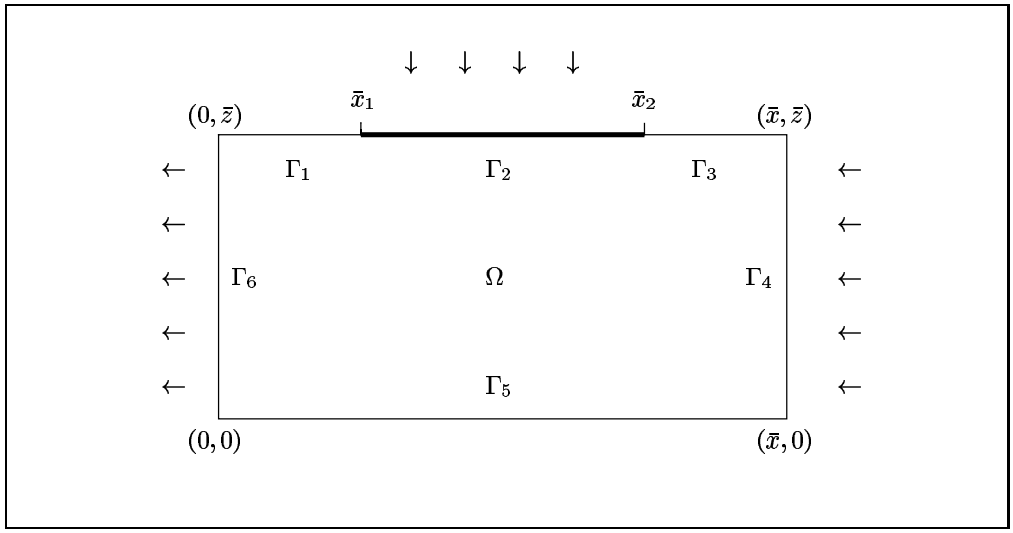
$$\begin{aligned}
\Delta\phi(x,z) &= 0 && \text{in } \Omega\,,\\
\tfrac{\partial}{\partial n}\phi(x,z) &= 0 && \text{on } \Gamma_1\cup\Gamma_3\cup\Gamma_5\,,\\
\phi(\bar{x},z) &= \phi_4 && \text{on } \Gamma_4\,,\\
\phi(0,z) &= \phi_6 && \text{on } \Gamma_6\,,\\
\tfrac{\partial}{\partial n}\phi(x,\bar{z}) &= \tfrac{1}{d}\left(u(x)-\phi(x,\bar{z})\right) && \text{on } \Gamma_2\,.
\end{aligned} \tag{7.1.2}$$

Here, the domain Ω and its boundary parts Γ_i $(i=1,\ldots,6)$ are defined by

$$\begin{aligned}
\Omega &= (0,\bar{x})\times(0,\bar{z})\,,\\
\Gamma_1 &= \{(x,\bar{z}) : x\in[0,\bar{x}_1]\}\,, & \Gamma_2 &= \{(x,\bar{z}) : x\in(\bar{x}_1,\bar{x}_2]\}\,,\\
\Gamma_3 &= \{(x,\bar{z}) : x\in(\bar{x}_2,\bar{x}]\}\,, & \Gamma_4 &= \{(\bar{x},z) : z\in[0,\bar{z}]\}\,,\\
\Gamma_5 &= \{(x,0) : x\in[0,\bar{x}]\}\,, & \Gamma_6 &= \{(0,z) : z\in[0,\bar{z}]\}\,.
\end{aligned}$$

The geography of the domain is shown in Figure 7.1.1. The variable ϕ denotes the state, and u is the control variable. The domain is two–dimensional and presents a vertical cut. Thus, $\phi=\phi(x,z)$. The control u acts on the boundary part Γ_2 only, and is accordingly given as $u=u(x)$.

Figure 7.1.1: The geography of the domain Ω.



We now explain the derivation of the problem and the different terms in the cost function (7.1.1). This is due to private communications with engineers from TGU GmbH, our industrial partner. As stated above, we are dealing with a mixing problem upon the extraction of ground water that originates from two different water bodies.

One is a stream and the other a surface water body in reclaimed land. The motivation behind the problem is the controlled flooding of Γ_2 for use as a relief basin when the water level is high in the stream. Thus the height of water on Γ_2 is considered as the control. The stream is, averaged over time, represented by the constant potential ϕ_4 on Γ_4. Leakage from the overlying surface water on Γ_2 occurs inversely proportional to the layer thickness d and the difference in piezometric head in the overlying surface water, u, and on the boundary Γ_2 of Ω, ϕ. This leads to mixed boundary conditions. Ground water is extracted from a perfect well on Γ_6. The potential on Γ_6 is held on a fixed value ϕ_6, $\phi_6 < \phi_4$. No–flow conditions are imposed on the boundary parts Γ_1, Γ_3, and Γ_5.

We assume the water coming in from Γ_2 to be of worse quality than that coming in from Γ_4, thus requiring treatment. Treatment cost for the water extracted via the well is assumed to rise quadratically with inflow from Γ_2. In addition, pumping cost arises if the water level u deviates from a fixed level u_v on Γ_2. This is cost of exercising the control. Pumping cost for the well, proportional to the amount of water extracted on Γ_6, and pumping height ϕ_f, is relevant as well. The goal is to minimize total cost while holding the state on Γ_6 on a fixed potential, ϕ_6, through pumping out of the well. This is described, with appropriate choices of the cost parameters β_i $(i = 1, 2, 3)$, by

$$\begin{aligned} \min_{(\phi,u)} \quad & \Big\{ \beta_1 \int_{\Gamma_2} Q_2^2(\phi(x,\bar{z}), u(x))\, dx + \beta_2 \int_{\Gamma_2} (u(x) - u_v)^2\, dx \\ & \qquad + \beta_3 \int_{\Gamma_6} Q_6(\phi(0,z))\,(\phi_f - \phi_6)\, dz \Big\} . \end{aligned} \tag{7.1.3}$$

The cost function (7.1.3) is dominated by the treatment cost for the water extracted on Γ_6. It is mainly the water flux infiltrating on Γ_2, Q_2, that determines this cost. Pumping cost arises if the water level u deviates heavily from a fixed level u_v. Pumping cost for the well, proportional to the amount of extracted water, Q_6, and pumping height ϕ_f, is given in the last term of the cost function (7.1.3).

The cost function (7.1.3) can be written entirely in terms of state, ϕ, and control, u. To do this, we refer to the size of the domain in x–, y– and z–direction by $\bar{x}$, $\bar{y}$, and $\bar{z}$, respectively. The width of Γ_2 is denoted by $\bar{x}_2 - \bar{x}_1$. We refer to the height d of the bounding layer only through a factor α, given as $\alpha = k_2/d$, where k_2 is the hydraulic conductivity on Γ_2. Hydraulic conductivity in the domain Ω is denoted by k_f. The flow Q_2 from Γ_2 into Ω is in a point $(x, \bar{z})$ given by

$$\begin{aligned} Q_2(\phi(x,\bar{z}), u(x)) &= k_g \cdot \tfrac{\partial}{\partial n}\phi(x,\bar{z}) \cdot (\bar{x}_2 - \bar{x}_1) \cdot \bar{y} \\ &= \alpha\,(u(x) - \phi(x,\bar{z})) \cdot (\bar{x}_2 - \bar{x}_1) \cdot \bar{y} \qquad x \in (\bar{x}_1, \bar{x}_2] \end{aligned}$$

and flow from Ω into Γ_6 in a point $(0, z)$ is given by

$$Q_6(\phi(0,z)) = k_f \cdot \frac{\partial}{\partial n}\phi(0,z) \cdot \bar{z} \cdot \bar{y} \qquad z \in [0, \bar{z}]\,.$$

Using the abbreviations $\tilde{\alpha} = \alpha \cdot (\bar{x}_2 - \bar{x}_1) \cdot \bar{y}$ and $\tilde{k}_f = k_f \cdot \bar{z} \cdot \bar{y}$, the objective

function (7.1.3) is equivalent to

$$\min_{(\phi,u)} \quad \Big\{ \beta_1 \int_{\Gamma_2} \tilde{\alpha}^2 \, (u(x) - \phi(x,\bar{z}))^2 \, dx + \beta_2 \int_{\Gamma_2} (u(x) - u_v)^2 \, dx \\ + \beta_3 \int_{\Gamma_6} \tilde{k}_f \, (\phi_f - \phi_6) \, \tfrac{\partial}{\partial n} \phi(0,\bar{z}) \, dz \Big\} \, . \tag{7.1.4}$$

The formulation (7.1.4), in turn, allows to state the problem in a standard formulation for linear quadratic control problems

$$\min_{(\phi,u)} \quad \Big\{ \tfrac{\eta_1}{2} \int_{\Gamma_2} u^2(x) \, dx + \eta_2 \int_{\Gamma_2} u(x) \, \phi(x,\bar{z}) \, dx + \tfrac{\eta_3}{2} \int_{\Gamma_2} \phi^2(x,\bar{z}) \, dx \\ + \eta_4 \int_{\Gamma_6} \tfrac{\partial}{\partial n} \phi(0,z) \, dz + \eta_5 \int_{\Gamma_2} u(x) \, u_v \, dx \Big\}$$

subject to (7.1.2). In this formulation of the objective, which has already been stated in (7.1.1), the constant term $\beta_2 \int_{\Gamma_2} u_v^2 \, dx$ has been omitted because it is irrelevant for the optimization. The parameters η_i $(i = 1, \ldots, 5)$ are

$$\eta_1 = 2(\beta_1 \, \tilde{\alpha}^2 + \beta_2) \, , \eta_2 = -2\beta_1 \, \tilde{\alpha}^2 \, , \eta_3 = 2\beta_1 \, \tilde{\alpha}^2 \, , \eta_4 = \beta_3 \, \tilde{k}_f \, (\phi_f - \phi_6) \, , \eta_5 = -2\beta_2 \, .$$

The constants used in the computations, as proposed by engineers from TGU GmbH, are

$$\begin{aligned} &\beta_1 = 2 \, \$ \, s^2/m^7 \, , \beta_2 = 0.1 \, \$/m^3 \, , \beta_3 = 0.2 \, \$ \, s/m^5 \, , \\ &k_f = 0.5 \, m/s \, , \alpha = 5 \cdot 10^{-3} \, 1/s \, , \\ &\phi_4 = 20m \, , \phi_6 = 10m \, , \phi_f = 30m \, , u_v = 21m \, , \\ &\bar{z} = 10m \, , \bar{x} = s_f \cdot \bar{z} \, , \bar{y} = 1m \, , \end{aligned}$$

so that, e.g. for $s_f = 2$,

$$\eta_1 = 0.45 \frac{\$}{m^3}, \eta_2 = -0.25 \frac{\$}{m^3}, \eta_3 = 0.25 \frac{\$}{m^3}, \eta_4 = 20.0 \frac{\$}{m}, \eta_5 = -0.2 \frac{\$}{m^3}.$$

7.2 The Discretized Problem

We define a uniform grid on the domain Ω, containing m grid points. The piezometric head, ϕ, is defined on all grid vertices, i.e., $\phi \in I\!R^m$. There are n grid points, $n \ll m$, for the boundary Γ_2. The control u is defined on k of the grid vertices of the boundary Γ_2, i.e., $u \in I\!R^k$ with $k \leq n$. We apply a second order finite difference discretization to the problem (7.1.1) s.t. (7.1.2). This is derived in detail for the case $k = n$ in the following Section 7.2.1. The case $k = 1$ is considered in Section 7.2.2, and the cases $k = 2$ and $k = 4$ are treated in Section 7.2.3.

The resulting finite dimensional problem is to minimize a quadratic function F under linear constraints on state and control,

$$\min_{(\phi_h,u_h)} \; F(\phi_h, u_h) \text{ s.t. } A\phi_h + Bu_h = b. \tag{7.2.1}$$

The discrete objective function can be written as

$$F(\phi_h, u_h) = \frac{\eta_1}{2} \, u_h^T H_{uu} \, u_h + \eta_2 \, \phi_h^T H_{\phi u} \, u_h + \frac{\eta_3}{2} \, \phi_h^T H_{\phi\phi} \, \phi_h + \eta_4 \, c^T \phi_h + \eta_5 \, d^T u_h \, . \tag{7.2.2}$$

The discrete Lagrangian is given by

$$L(\phi_h, u_h, \lambda_h) = F(\phi_h, u_h) + \lambda_h^T (A\phi_h + Bu_h - b).$$

Note that for this quadratic problem the second partial derivatives $L_{\phi\phi}$, $L_{\phi u}$ and L_{uu} of the Lagrangian coincide with $H_{\phi\phi}$, $H_{\phi u}$ and H_{uu}, respectively.

Applying the first order optimality conditions (Karush–Kuhn–Tucker conditions, see e.g. [36, Ch. 3]), the system

$$\begin{pmatrix} \eta_3 L_{\phi\phi} & \eta_2 L_{\phi u} & A^T \\ \eta_2 L_{u\phi} & \eta_1 L_{uu} & B^T \\ A & B & 0 \end{pmatrix} \begin{pmatrix} \phi_h \\ u_h \\ \lambda_h \end{pmatrix} = \begin{pmatrix} -\eta_4 c \\ -\eta_5 d \\ b \end{pmatrix} \tag{7.2.3}$$

has to be solved, where

$$\begin{gathered} \phi_h \in I\!R^m,\ u_h \in I\!R^k,\ \lambda_h \in I\!R^m,\ c \in I\!R^m,\ d \in I\!R^k,\ b \in I\!R^m, \\ L_{\phi\phi} \in I\!R^{m\times m},\ L_{uu} \in I\!R^{k\times k},\ L_{\phi u} \in I\!R^{m\times k},\ A \in I\!R^{m\times m},\ B \in I\!R^{m\times k}. \end{gathered}$$

If A is invertible, the discrete state equation $A\phi_h + Bu_h = b$ can be solved for ϕ_h, $\phi_h(u_h) = A^{-1}(b - Bu_h)$. Thus, one can define the equivalent unconstrained problem

$$\min_{u_h} \ J(u_h) = F(\phi_h(u_h), u_h)\,. \tag{7.2.4}$$

In order to get the gradient, g_h, of the unconstrained problem (7.2.4), the state and the costate equations must be solved exactly. This means that for a given control u_h the following set of equations has to be satisfied,

$$\begin{pmatrix} \eta_3 L_{\phi\phi} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \phi_h \\ \lambda_h \end{pmatrix} = \begin{pmatrix} -\eta_4 c - \eta_2 L_{\phi u} u_h \\ b - Bu_h \end{pmatrix}.$$

Then the gradient of (7.2.4), also referred to as the reduced gradient of (7.2.1), can be computed as

$$g_h = B^T \lambda_h + \eta_1 L_{uu} u_h + \eta_2 L_{\phi u} \phi_h + \eta_5 d.$$

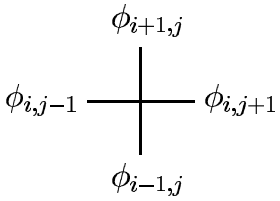
This quantity can be used to monitor progress of the iteration. The Hessian of the discrete unconstrained problem (7.2.4), the reduced Hessian of (7.2.1), is given by

$$H = \eta_3 B^T A^{-T} L_{\phi\phi} A^{-1} B - \eta_2 L_{u\phi} A^{-1} B - \eta_2 B^T A^{-T} L_{\phi u} + \eta_1 L_{uu}.$$

7.2.1 The Finite Difference Discretization

We now consider the finite difference scheme for the discrete problem (7.2.1). The variables in the discretization grid are organized like it is given in Figure 7.2.1, with the reference point $\phi_{i,j}$ being located in the center. The first point, ϕ_{11}, is located in the lower left corner of the discretization grid, and the bottom points are thus numbered $\phi_{11}, \ldots, \phi_{1,n_x}$. The last variable is ϕ_{n_z,n_x}, located in the upper right corner of the grid.

Figure 7.2.1: The organization of the variables.



The dimensions are given as follows. The number of subintervals in z–direction and in x–direction is s_z and s_x, respectively. Here, $s_x = s_f \cdot s_z$ for a stretching factor $s_f \in I\!N$. Similarly, the number of points in z–direction and in x–direction is n_z, $n_z = s_z + 1$, and n_x. The number of state variables is m, $m = n_z \cdot n_x$, and the number of control variables is k. The number of discretization points for Γ_1 is g_1, and the number of points for Γ_2 is n, $n = g_2 - g_1$.

In this formulation, the number n of points for Γ_2 is equal to the number k of control variables. The state variable ϕ_h, adjoint variable λ_h, and control variable u_h are organized in vectors in the following way,

$$\phi_h = \begin{pmatrix} \phi_{1,1} \\ \phi_{1,2} \\ \vdots \\ \phi_{1,n_x} \\ \hline \phi_{2,1} \\ \vdots \\ \hline \phi_{n_z,1} \\ \vdots \\ \phi_{n_z,n_x} \end{pmatrix}, \quad \lambda_h = \begin{pmatrix} \lambda_{1,1} \\ \lambda_{1,2} \\ \vdots \\ \lambda_{1,n_x} \\ \hline \lambda_{2,1} \\ \vdots \\ \hline \lambda_{n_z,1} \\ \vdots \\ \lambda_{n_z,n_x} \end{pmatrix}, \quad u_h = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}.$$

The mesh size h is determined by the number s_z of subintervals in z–direction and thus given by $h = 1/nz$. In the domain Ω we use for the state equation $\Delta\phi = 0$ the five–point discretization of the Laplacian of order $\mathcal{O}(h^2)$,

$$-\frac{4}{h^2}\,\phi_{ij} + \frac{1}{h^2}\,(\phi_{i,j+1} + \phi_{i,j-1} + \phi_{i-1,j} + \phi_{i+1,j}) = 0\,. \tag{7.2.5}$$

On the boundary parts Γ_4 and Γ_6, the piezometric head is fixed to

$$\phi_{i,n_x} = \phi_4, \quad \phi_{i,1} = \phi_6 \quad (i = 1, \ldots, n_z). \tag{7.2.6}$$

On top and bottom of the domain, auxiliary points ϕ_{0j} and $\phi_{n_z+1,j}$ are introduced for the discretization of the normal derivative. On Γ_5, the difference formula

$$\frac{1}{2h}\,(\phi_{0j} - \phi_{2j}) = 0 \quad (j = 2, \ldots, n_x - 1),$$

leads to the discrete boundary condition

$$-\frac{4}{h^2}\,\phi_{1,j}+\frac{1}{h^2}\,(\phi_{1,j+1}+\phi_{1,j-1}+2\,\phi_{2,j})=0\,. \tag{7.2.7}$$

On Γ_1 and on Γ_3, the condition $\frac{\partial}{\partial n}\phi=0$ leads in a similar way to

$$\frac{1}{2h}\,(\phi_{n_z+1,j}-\phi_{n_z-1,j})=0 \quad (j=2,\ldots,g_1, \text{ and } j=g_2+1,\ldots,n_x-1),$$

which is used to arrive at

$$-\frac{4}{h^2}\,\phi_{n_z,j}+\frac{1}{h^2}\,(\phi_{n_z,j+1}+\phi_{n_z,j-1}+2\,\phi_{n_z-1,j})=0\,. \tag{7.2.8}$$

Finally, on Γ_2 the control has to be considered in the boundary condition

$$\frac{1}{2h}\,(\phi_{n_z+1,j}-\phi_{n_z-1,j})=\alpha\,(u_{j-g_1}-\phi_{n_z,j}) \quad (j=g_1+1,\ldots,g_2). \tag{7.2.9}$$

The resulting equation is

$$-\frac{4+2\alpha h}{h^2}\,\phi_{n_z,j}+\frac{1}{h^2}\,(\phi_{n_z,j+1}+\phi_{n_z,j-1}+2\,\phi_{n_z-1,j})+\frac{2\alpha}{h}\,u_{j-g_1}=0\,. \tag{7.2.10}$$

Equations (7.2.5), (7.2.6), (7.2.7), (7.2.8), and (7.2.10) allow to formulate the discretized state equation

$$A\,\phi_h+B\,u_h=b.$$

The matrix A is given in Figure 7.2.2. Its entries p, q, r, s are

$$p=\frac{-4}{h^2}, \quad q=\frac{-4+2\alpha h}{h^2}, \quad r=\frac{1}{h^2}, \quad \text{and} \quad s=\frac{-1}{h^2}.$$

In the lowest diagonal block, q occupies the places ii for $i=g_1+1,\ldots,g_2$, i.e., the n places corresponding to the boundary Γ_2.

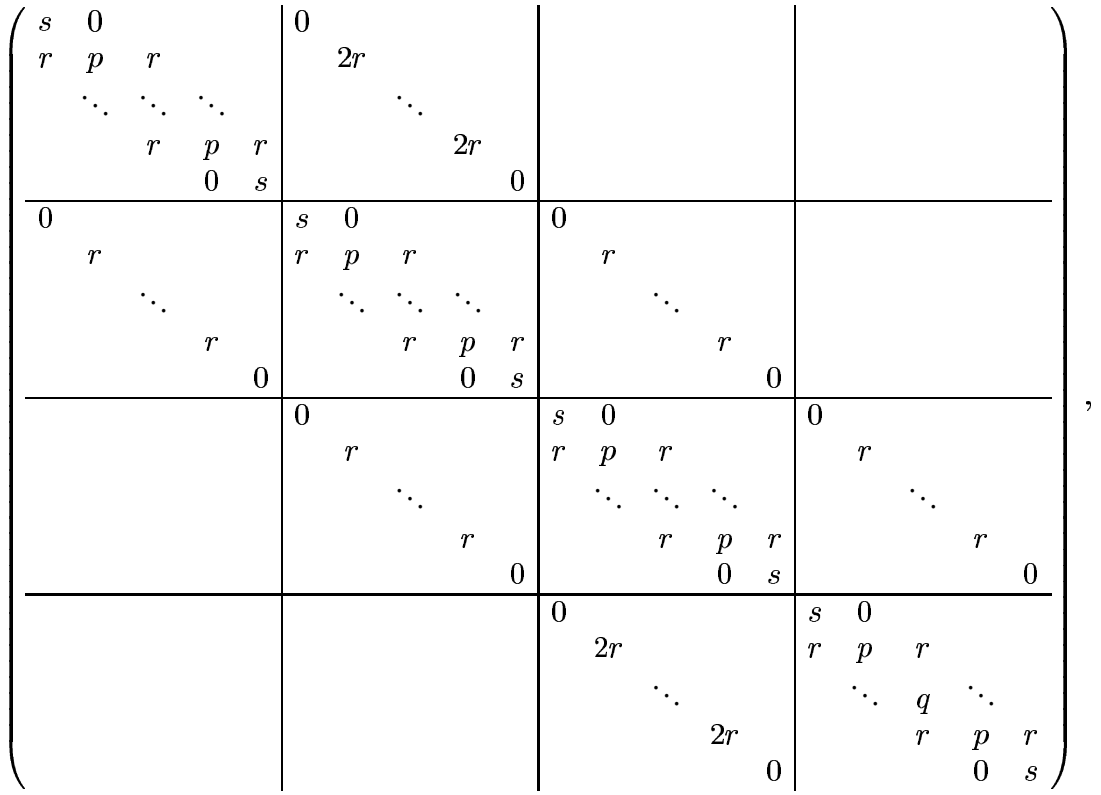
The matrix B is thus

$$B=\frac{2\alpha}{h}\begin{pmatrix} 0_{(n_x-1)n_z+g_1,n} \\ I_{n,n} \\ 0_{n_x-g_2,n} \end{pmatrix},$$

and the right hand side b is, according to the definition of A and B, given in (7.2.11). The term $\eta_4\,\int_{\Gamma_6}\phi_n\,ds$ in the cost functional is represented on the discrete level by the dot product $\eta_4\,c^T\phi_h$. The stencil

$$\frac{1}{h}\,(\phi_{i1}-\phi_{i2})=0 \quad (i=1,\ldots,n_z)$$

Figure 7.2.2: The matrix A of the discretization.



is used to arrive at the vector c, also given in (7.2.11). The term $\eta_5 \int_{\Gamma_2} u\, u_v\, ds$ in the cost functional is represented on the discrete level by the dot product $\eta_5\, d^T u_h$. Thus, d is the vector of ones multiplied by the scalar $\eta_5\, u_v$. The right hand side of the KKT system is given by

$$b = \frac{-1}{h^2} \left(\begin{array}{c} \phi_4 \\ 0 \\ \vdots \\ 0 \\ \phi_6 \\ \hline \vdots \\ \hline \phi_4 \\ 0 \\ \vdots \\ 0 \\ \phi_6 \end{array} \right), \quad c = h \frac{1}{h} \left(\begin{array}{c} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \\ \hline \vdots \\ \hline 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{array} \right), \quad d = \frac{1}{n} \begin{pmatrix} \eta_5\, u_v \\ \vdots \\ \eta_5\, u_v \end{pmatrix}. \tag{7.2.11}$$

The matrices L_{uu}, $L_{\phi u}$, and $L_{\phi\phi}$, all representing integration on the boundary Γ_2, are

$$L_{uu} = \frac{1}{n} I_{n,n}\,, \quad L_{\phi u} = \frac{1}{n} \begin{pmatrix} 0_{(n_x-1)n_z+g_1,n} \\ I_{n,n} \\ 0_{n_x-g_2,n} \end{pmatrix},$$

and

$$L_{\phi\phi} = \frac{1}{n} \begin{pmatrix} 0_{(n_x-1)n_z+g_1,(n_x-1)n_z+g_1} & 0_{(n_x-1)n_z+g_1,n} & 0_{(n_x-1)n_z+g_1,n_x-g_2} \\ 0_{n,(n_x-1)n_z+g_1} & I_{n,n} & 0_{n,n_x-g_2} \\ 0_{n_x-g_2,(n_x-1)n_z+g_1} & 0_{n_x-g_2,n} & 0_{n_x-g_2,n_x-g_2} \end{pmatrix}.$$

7.2.2 The Case of a Uniform Control

With the problem formulation derived above, a non–uniform control is computed, i.e., the piezometric head is not constant over the boundary Γ_2. Although the computed inclination of the surface water body is small this solution is not physical. We thus turn to another formulation where we only compute one single control, which nevertheless acts on all of Γ_2. The analytic problem is still (7.1.3) subject to (7.1.2), but supplemented with the additional constraint

$$u(x,\bar{z}) \equiv u \quad (x,\bar{z}) \in \Gamma_2\,.$$

Again, we define a uniform grid on the domain Ω, containing m grid points. The piezometric head, ϕ, is defined on all grid vertices, i.e., $\phi \in I\!R^m$, and the single–valued control $u_1 \in I\!R$ acts on all piezometric heads, $\phi_{n_z,j-g_1}$ $(j = g_1+1,\ldots,g_2)$, on the boundary Γ_2. That is, we now treat the case $k = 1$. A second order finite difference discretization, basically identical to the discretization in Section 7.2.1, is applied. The main change occurs in (7.2.9) and (7.2.10), where u_{j-g_1} $(j = g_1+1,\ldots,g_2)$ is replaced by the single scalar u_1. Note that although n is still the number k of discretization points on Γ_2, the number of control variables is now 1. Thus, the dimension of the system K is $2m+1$, with

$$\begin{gathered} \phi_h \in I\!R^m,\ u_h \in I\!R,\ \lambda_h \in I\!R^m,\ c \in I\!R^m,\ d \in I\!R,\ b \in I\!R^m, \\ L_{\phi\phi} \in I\!R^{m\times m},\ L_{uu} \in I\!R,\ L_{\phi u} \in I\!R^m,\ A \in I\!R^{m\times m},\ B \in I\!R^m. \end{gathered}$$

Only the matrices A and $L_{\phi\phi}$ are unchanged by the transition to a single control. The matrix B is replaced by

$$B = \frac{2\alpha}{h} \begin{pmatrix} 0_{(n_x-1)n_z+g_1} \\ 1\!\!1_n \\ 0_{n_x-g_2} \end{pmatrix},$$

where $\mathbb{1}_n$ denotes the vector of ones of length n. Accordingly,

$$L_{uu} = 1\,, \quad \text{and} \quad L_{\phi u} = \frac{1}{n} \begin{pmatrix} 0_{(n_x-1)n_z+g_1} \\ \mathbb{1}_n \\ 0_{n_x-g_2} \end{pmatrix}.$$

Note that although $d \in \mathbb{R}$, the linear term $\eta_5\, d^T u_h$ in the discrete cost functional (7.2.2) stays the same. The transition to a single control u requires a second look on the weighting of the terms in the cost functional. Because the line integrals on Γ_2 and Γ_6 require the factors $1/n$ and $1/n_z$, respectively, we have to omit the factor $1/n$ in c now to reflect the integration. This is analogous to the treatment of L_{uu} and $L_{\phi u}$.

7.2.3 The Case of a Fixed Number of Controls

Apart from the problem formulations in Sections 7.2.1 and 7.2.2, another formulation is sensible. One can think of a fixed number k of different cells for the surface water body in reclaimed land. We consider the cases $k = 2$ and $k = 4$. This leads to a fixed number k of control variables. The analytic problem is given by the objective function (7.1.3) subject to the state equation (7.1.2), which has to be supplemented with the additional constraints

$$u(x, \bar{z}) \equiv u_l \quad (l = 1, \ldots, k)$$

for $x \in [\bar{x}_1, \bar{x}_2]$, i.e., $(x, \bar{z}) \in \Gamma_2$.

As before, a uniform grid with m grid points is defined, and the piezometric head ϕ is in $\mathbb{R}^m$ The k–valued control $u \in \mathbb{R}^k$ acts on all piezometric heads, $\phi_{n_z, j-g_1}$ $(j = g_1 + 1, \ldots, g_1 + n)$ on the boundary Γ_2. Recall that $g_1 + n = g_2$. A second order finite difference discretization is applied. It is basically identical to the discretization Section 7.2.1. The main change occurs in (7.2.9) and (7.2.10), where u_{j-g_1} $(j = g_1 + 1, \ldots, g_2)$ is replaced by u_{j-g_1} $(j = g_1 + (l-1) \cdot \frac{n}{k} + 1, \ldots, g_1 + l \cdot \frac{n}{k})$ for $l = 1, \ldots, k$. Note that n is still the number of discretization points on Γ_2, and that the number of control variables is k. Thus, the dimension of the system K is $2m + k$. The matrices A and $L_{\phi\phi}$ are unchanged by the transition to a control $u \in \mathbb{R}^k$. The former $L_{\phi u}$ is for $k = 2$ replaced by

$$L_{\phi u} = \frac{1}{n} \begin{pmatrix} 0_{(n_x-1)n_z+g_1,2} & \\ \mathbb{1}_{n/2} & 0_{n/2} \\ 0_{n/2} & \mathbb{1}_{n/2} \\ 0_{n_x-g_2,2} & \end{pmatrix}.$$

where $\mathbb{1}_n$ denotes the vector of ones of length n. Accordingly,

$$L_{uu} = \frac{1}{k}\, I_{k\times k}\,, \quad \text{and} \quad B = \frac{2\alpha}{h}\, n\, L_{\phi u}\,.$$

If $k = 4$,

$$L_{\phi u} = \frac{1}{n} \begin{pmatrix} & 0_{(n_x-1)n_z+g_1,4} & & \\ \mathbb{1}_{n/4} & 0_{n/4} & 0_{n/4} & 0_{n/4} \\ 0_{n/4} & \mathbb{1}_{n/4} & 0_{n/4} & 0_{n/4} \\ 0_{n/4} & 0_{n/4} & \mathbb{1}_{n/4} & 0_{n/4} \\ 0_{n/4} & 0_{n/4} & 0_{n/4} & \mathbb{1}_{n/4} \\ & 0_{n_x-g_2,4} & & \end{pmatrix}.$$

7.3 Numerical Results

We now turn to the numerical results obtained for the example problem treated in Section 7.1. We investigate mainly the situation $k = n$. There, the number of controls increases as the grid is refined. In terms of the numerics, this is the most interesting case, although the cases $k = 1, 2$, and 4 are not easy to handle either. We add the results for these cases after a detailed discussion of the case $k = n$. For this situation, the preconditioner $\tilde{K}$ is in various constellations tested intensively. Different subpreconditioners are used. The indefinite preconditioner is also compared to the symmetric positive definite preconditioners reviewed in Section 6.2.

As laid down in Section 7.2, a uniform grid is defined on the domain Ω, containing m grid points. The piezometric head, ϕ, is defined on all grid vertices, i.e., $\phi \in \mathbb{R}^m$. There are n grid points, $n \ll m$, for the boundary Γ_2. The control u is defined on k of the grid vertices of the boundary Γ_2, i.e., $u \in \mathbb{R}^k$ with $k \leq n$. A second order finite–difference discretization is applied to the problem (7.1.1) s.t. (7.1.2). The discretization is described in Section 7.2.1. The resulting finite dimensional problem is to minimize a quadratic function F under linear constraints on state and control. Compare Equations (7.2.1) and (7.2.2). The parameters η_i $(i = 1, \ldots, 5)$ are incorporated into the matrices and vectors, respectively, as denoted in (7.2.3).

With grid refinement, the linear systems grow fast. In our numerical experiments we can currently consider $n_z = 4$, 8, 16, 32, 64, and 128, discretization points in vertical direction. The dimension N of the entire system is given as $2m + k$. For $n_x = 8, 16, \ldots, 256$ discretization points in the horizontal and $k = n$ controls, this corresponds to systems K of dimension 94×94 up to 66434×66434. Thus, the systems grow fast to large dimensions, and it is in general advantageous, if not crucial, to exploit their sparsity structure. The nonzero structure of the full system K is depicted in Figure 7.3.1.

Compare Table 7.3.1 for the dimension of K and its nonzero entries nnz in absolute and relative figures. The relative number of nonzero entries in the system matrix decreases by a factor of 4 with each doubling of the number of grid points in one spatial direction.

Figure 7.3.1: Sparsity structure of K ($k = n$).

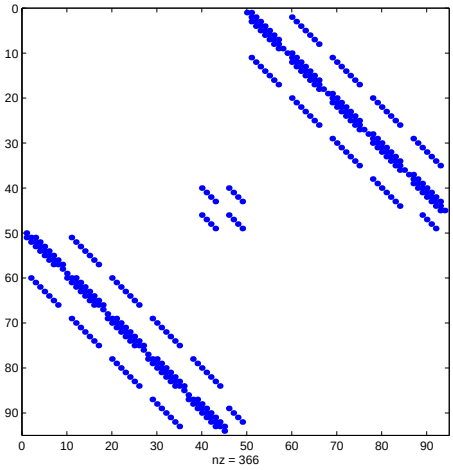


Table 7.3.1: Dimension and sparsity of K for $k = n$.

n_z	n_x	n	m	N	nnz	nnz in %
4	8	4	45	94	366	4.100
8	16	8	153	314	1374	1.394
16	32	16	561	1138	5310	0.410
32	64	32	2145	4322	20862	0.112
64	128	64	8385	16834	82686	0.029
128	256	128	33153	66434	329214	0.007

The conditioning of the system can be seen in Table 7.3.2. Obviously, the condition number $\kappa(K)$ increases considerably as the mesh size h approaches zero. This depends mainly on the conditioning of the submatrix A. It deteriorates as the mesh size h is decreased. The conditioning of the remaining submatrices B, L_{uu}, $L_{\phi u}$, and $L_{\phi\phi}$ remains unchanged as the grid refines. It is obvious from the structure of the matrices, however, that the eigenvalues and singular values of B, L_{uu}, $L_{\phi u}$, and $L_{\phi\phi}$ move to zero as the mesh size decreases. The combination of these effects leads to an unfavorable eigenvalue distribution in the assembled system. This is the reason for the rapidly increasing spectral condition number $\kappa(K)$ that is exhibited in Table 7.3.2.

The small eigenvalues of K decrease to zero, and the large eigenvalues increase as the mesh is refined. This effect is clearly visible in Figure 7.3.2, where for four different discretizations the positive eigenvalues of K are plotted on a logarithmic scale along with the negative eigenvalues in absolute value.

As the size of K and its condition number increase, the performance of iterative solvers on the linear system $K\,x = r$ deteriorates considerably. In our numerical ex-

Table 7.3.2: Condition numbers of K and A ($k = n$).

n_z	N	$\kappa(K)$	$\kappa(A)$
4	94	1356.03	47.974
8	314	11238.04	192.475
16	1138	94713.87	770.496
32	4322	832836.00	3082.586

periments, the KKT system is solved with the Krylov subspace methods MINRES [66] and GMRES [70]. On the original system K, MINRES diverges, and GMRES takes an unacceptably high number of steps. The number of required steps is almost as high as the dimension N of the linear system. See Table 7.3.3 for the iteration and floating point operation numbers. Note that due to the increasing work and storage requirements of GMRES iterations the machine was able only to perform the computations up to the discretization $n_z = 32$. The high computational effort of the unpreconditioned iterative solution, due to the unfavorable eigenvalue distribution of K depicted in Figure 7.3.2, motivates the use of preconditioners. Discretizations up to $n_z = 128$ can be considered in the preconditioned cases.

Table 7.3.3: Iteration and operation count for GMRES on the original K ($k = n$).

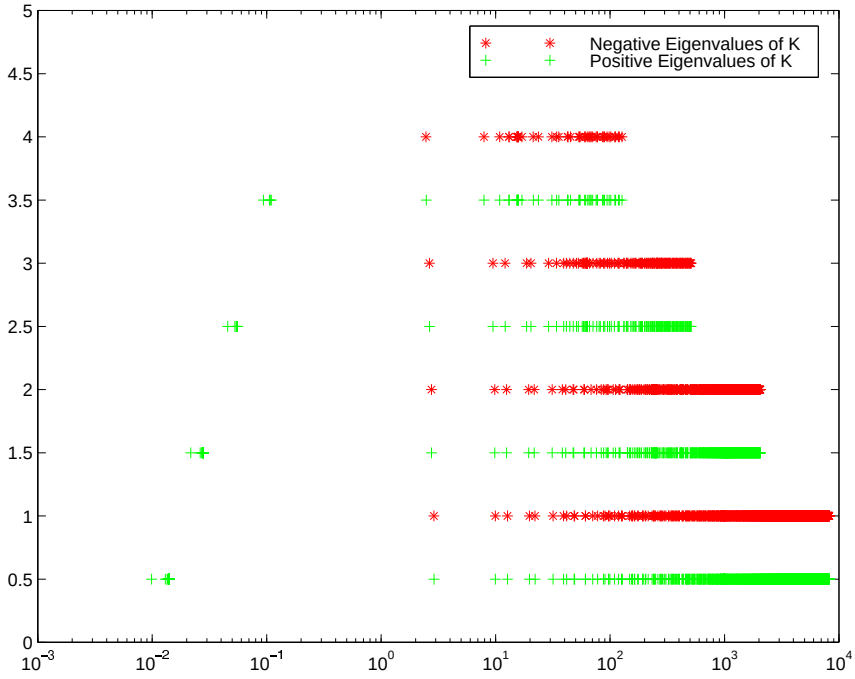
n_z	4	8	16	32
N	94	314	1138	4322
iterations	88	294	1028	3666
Megaflops	2.73	93.44	4,015.24	191,291.60

The implementation of GMRES used in the numerical experiments requires only to furnish a routine for matrix–vector multiplication. Thus, explicitly storing the system matrix is not necessary. Only the smaller subblocks are used, e.g. A and B. To perform the solve with the preconditioner, we proceed like it is described in Section 5.2.3. We deal analogously with MINRES and the preconditioners P_1, P_2 and P_3 from [5]. Compare Section 6.2. The stopping criterion for all iterative procedures is a threshold value of 10^{-6} for the l_2–residual. All computations are done with *Matlab, Version 5.3* on a *Sun Ultra 60*.

Iteration counts for GMRES and MINRES, employing the considered preconditioners $\tilde{K}$, P_1, P_2, and P_3, are given in Tables 7.3.4 and 7.3.8. The corresponding operation counts can be found in Tables 7.3.5 and 7.3.9.

The first set of computations, Tables 7.3.4 and 7.3.5, is dedicated to a comparison between the different preconditioners. The preconditioner $\tilde{K}$ analyzed in this work is compared to the preconditioners P_1, P_2, and P_3 from [5]. In this set of compu-

Figure 7.3.2: Eigenvalues of original system K for $n_z = 4$, 8, 16, and 32 ($k = n$).



tations, exact inverses for the submatrices are used, usually obtained via (complete) LU–decompositions. We use $P_A = A$ for $\tilde{K}$ and P_1, P_2, and P_3 in all these computations. For $\tilde{K}$, we consider both the ideal case $P_H = H$, and, due to cost issues, also the cheap version $P_H = I$. As a first observation we note that, as predicted by theory in Section 5.3.3, only three GMRES iterations are necessary for the ideal version of $\tilde{K}$ with $P_A = A$ and $P_H = H$. Since the ideal version of $\tilde{K}$ requires the reduced Hessian H, it is desirable to use more accessible versions of P_H also. Upon use of $P_H = I$, the iteration count is only marginally higher, and still constant over all considered discretizations. Similarly, a constant number of iterations seems to be sufficient upon use of P_3. The number of iterations increases with mesh refinement when P_1 and P_2 are used. Theoretical results are cited in Section 6.2 that provide an upper limit to the number of MINRES steps. These limits are $2k+3$ and $k+2$ steps for systems preconditioned with P_1 and P_2, respectively, in the ideal cases. These results hold for the case with zero off–diagonal entries $L_{\phi u}$ and $L_{u\phi}$, i.e., they are not strictly applicable to this example. Here, the number of steps in case $k = 4$ surpasses the upper limit. This can be explained with effects of finite–precision arithmetic and of nonzero $L_{\phi u}$, $L_{u\phi}$. In all other cases ($k \geq 8$), the performance of MINRES is at least as favorable as predicted by the upper iteration limit. The application of P_3 allows faster solutions than P_1 and P_2, but it cannot beat $\tilde{K}$. The preconditioner $\tilde{K}$ allows to solve the system with a smaller number of iterations than the other preconditioners.

Table 7.3.4: Iterations of GMRES on original system K and on preconditioned system $\tilde{K}^{-1}K$ with constellations $P_A = A, P_H = H$ and $P_A = A, P_H = I$. Iterations of MINRES on the systems preconditioned with P_1, P_2 and P_3 in their ideal versions, $P_A = A$. In all cases, $k = n$.

n_z	4	8	16	32	64	128
N	94	314	1138	4322	16834	66434
K	88	294	1028	3666	$\star$	$\star$
$\tilde{K}$ ($P_A = A, P_H = H$)	3	3	3	3	3	3
$\tilde{K}$ ($P_A = A, P_H = I$)	4	4	4	4	4	4
P_1 ($P_A = A$)	18	20	25	30	42	63
P_2 ($P_A = A$)	16	19	22	25	34	47
P_3 ($P_A = A$)	5	6	8	8	8	8

Table 7.3.5: Computational effort of GMRES iterations on original K and on preconditioned system $\tilde{K}^{-1}K$ in Megaflops. Computational effort of MINRES iterations with preconditioners P_1, P_2 and P_3 in their ideal versions, $P_A = A$. In all cases $k = n$.

n_z	4	8	16	32	64	128
N	94	314	1138	4322	16834	66434
K	2.73	93.44	$4K$	$191K$	$\star$	$\star$
$\tilde{K}$ ($P_A = A, P_H = H$)	0.03	0.25	2.90	38.52	548.30	8,091.43
$\tilde{K}$ ($P_A = A, P_H = I$)	0.04	0.27	2.61	30.77	418.49	5,976.46
P_1 ($P_A = A$)	0.12	0.73	6.41	64.48	779.12	10,226.02
P_2 ($P_A = A$)	0.11	0.71	5.94	58.66	708.18	9,118.92
P_3 ($P_A = A$)	0.05	0.42	4.33	44.86	528.83	6,850.00

Besides iterations, consider also the computational effort. The computational effort is measured with flops, see Table 7.3.5. Although only three steps are taken with $\tilde{K}$ employing $P_H = H$, this is not the recommended option. The choice $P_H = I$, requiring four steps, is considerably cheaper. Also cheaper than $\tilde{K}$ with $P_H = H$ is P_3 with eight steps. Nevertheless, $\tilde{K}$, especially in its realistic version with $P_H = I$, compares favorably to the symmetric positive definite preconditioners.

We have noted before that, as predicted by theory, only three GMRES iterations are necessary for the ideal version of $\tilde{K}$ ($P_A = A$, $P_H = H$). The preconditioned system has the single eigenvalue 1 with the geometric multiplicity 3. The appropriate Krylov subspace is very well captured after three iterations on the preconditioned system. This is obvious from the steep drop that occurs both for the norms of the residual $||r||$ and the error $||e||$ in the third iteration. Note also that the norm of the reduced gradient $||g||$ of the underlying optimization problem (7.2.1) is practically zero. See Table 7.3.6.

Typically, no such steep drops in residual, error, and gradient norm occur upon closure with a threshold value of 10^{-6} for the residual. Compare Table 7.3.7.

Table 7.3.6: Performance of GMRES for ideal $\tilde{K}$ with $P_A = A$, $P_H = H$ ($k = n$).

n_z	#it	CPU in s	$\|r\|$	$\|e\|$	$\|g\|$
4	3	$4.0 \cdot 10^{-2}$	$1.0 \cdot 10^{-14}$	$3.9 \cdot 10^{-14}$	$1.0 \cdot 10^{-15}$
8	3	$9.0 \cdot 10^{-2}$	$2.4 \cdot 10^{-14}$	$1.2 \cdot 10^{-12}$	$4.3 \cdot 10^{-15}$
16	3	$4.9 \cdot 10^{-1}$	$5.0 \cdot 10^{-14}$	$1.1 \cdot 10^{-11}$	$2.4 \cdot 10^{-15}$
32	3	$4.0 \cdot 10^{0}$	$1.8 \cdot 10^{-14}$	$7.8 \cdot 10^{-11}$	$3.3 \cdot 10^{-15}$
64	3	$4.2 \cdot 10^{1}$	$1.8 \cdot 10^{-13}$	$6.4 \cdot 10^{-10}$	$4.5 \cdot 10^{-15}$
128	3	$5.3 \cdot 10^{2}$	$4.8 \cdot 10^{-14}$	$4.4 \cdot 10^{-9}$	$4.9 \cdot 10^{-14}$

Table 7.3.7: Performance of GMRES for $\tilde{K}$ with $P_A = A$, $P_H = I$ ($k = n$).

n_z	# it	CPU in s	$\|r\|$	$\|e\|$	$\|g\|$
4	4	$4.0 \cdot 10^{-2}$	$1.5 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	$7.5 \cdot 10^{-6}$
8	5	$1.1 \cdot 10^{-1}$	$2.7 \cdot 10^{-7}$	$5.1 \cdot 10^{-6}$	$1.7 \cdot 10^{-6}$
16	5	$5.1 \cdot 10^{-1}$	$1.7 \cdot 10^{-7}$	$6.3 \cdot 10^{-6}$	$1.4 \cdot 10^{-6}$
32	4	$3.4 \cdot 10^{0}$	$4.8 \cdot 10^{-6}$	$3.5 \cdot 10^{-4}$	$5.6 \cdot 10^{-5}$
64	4	$3.2 \cdot 10^{1}$	$1.8 \cdot 10^{-6}$	$2.7 \cdot 10^{-4}$	$2.9 \cdot 10^{-5}$
128	4	$3.5 \cdot 10^{2}$	$6.6 \cdot 10^{-7}$	$1.9 \cdot 10^{-4}$	$1.5 \cdot 10^{-5}$

General versions of the preconditioner $\tilde{K}$ are tested in the second set of computations that is described in Tables 7.3.8 and 7.3.9. The ideal version, with $P_A = A$ and $P_H = H$ we have seen before, and also the case $P_H = I$. We also consider the choice $P_H = L_{uu}$ here which furnishes results only slightly different to $P_H = I$. This is no surprise because in this application L_{uu} is a scaled identity. Note that due to this, the case $P_H = \mathrm{diag}(L_{uu})$ that is of interest in Section 6.3 need not be specifically considered. The computations furnish the expected results: Use of the identity instead of the reduced Hessian causes GMRES to take more steps, but the higher number of steps is easily offset by the lower cost of individual iterations.

In these three constellations, P_A is, as before, the (complete) LU–decomposition of A. In the remaining computations, an approximation P_A to A is employed. This approximation is an incomplete LU–decomposition of A. It is used in conjunction with $P_H = I$ and with $P_H = \tilde{H}$. The matrix $\tilde{H}$, see Equation (5.3.8), is an approximation to the reduced Hessian in that it is built with P_A instead of A. For the approximation of A via its incomplete decomposition we use different drop tolerances. The incomplete LU–decomposition is a built–in *Matlab* routine. It is computed in the same (column-oriented) manner as the LU factorization except that after each column of L and U has been calculated, all entries in that column smaller in magnitude than the local drop tolerance are set to zero in the factors L or U. The local drop tolerance is the

user–defined drop tolerance multiplied by the norm of the column of A.

Table 7.3.8: Iterations of GMRES on original system K and on preconditioned system $\tilde{K}^{-1}K$ for different variants and combinations of the subpreconditioners P_H and P_A. In all cases, $k = n$.

n_z	4	8	16	32	64	128
N	94	314	1138	4322	16834	66434
K	88	294	1028	3666	$\star$	$\star$
($P_A = A$, $P_H = H$)	3	3	3	3	3	3
($P_A = A$, $P_H = L_{uu}$)	5	4	4	4	4	4
($P_A = A$, $P_H = I$)	4	5	5	4	4	4
($P_A = ILU(A, 10^{-5})$, $P_H = \tilde{H}$)	3	3	3	4	6	8
($P_A = ILU(A, 10^{-4})$, $P_H = \tilde{H}$)	3	4	4	6	10	16
($P_A = ILU(A, 10^{-3})$, $P_H = \tilde{H}$)	4	5	7	11	20	37
($P_A = ILU(A, 10^{-2})$, $P_H = \tilde{H}$)	6	8	13	24	44	82
($P_A = ILU(A, 10^{-5})$, $P_H = I$)	4	6	7	7	10	14
($P_A = ILU(A, 10^{-4})$, $P_H = I$)	5	7	9	10	15	24
($P_A = ILU(A, 10^{-3})$, $P_H = I$)	5	9	13	17	28	50
($P_A = ILU(A, 10^{-2})$, $P_H = I$)	8	14	21	32	50	92

With P_A instead of A, additional iterations are necessary in comparison to the ideal case to reach the stopping criterion. This must be expected and is obvious already for the smallest drop tolerance 10^{-5}. The effect is more pronounced for the larger drop tolerances of 10^{-4} to 10^{-2}. The influence of the approximation is even stronger for $P_H = I$. Consistently, more iterations are needed than in the case $P_H = \tilde{H}$. But this does not consistently require more work. See Table 7.3.9 for the corresponding operation count. For an incomplete LU–decomposition of A with drop tolerance 10^{-4}, the choice $P_H = I$ furnishes lower computational cost than $P_H = \tilde{H}$. Overall lowest computational cost is achieved in this set of experiments for the constellation $P_A = ILU(A, 10^{-3})$, $P_H = \tilde{H}$. For the choices $P_H = I$ and $P_H = \tilde{H}$ there are different optimal drop tolerances for the incomplete decomposition. In all these computations, the number of iterations increases as the grid refines. The incomplete decompositions of A are not good enough to achieve mesh–independent convergence. They do, however, furnish very usable alternatives to the ideal case. The most expensive of the considered incomplete decompositions with a drop tolerance of 10^{-5} allows already cheaper solves than any version with $P_A = A$.

After assembling and analyzing the results for the preconditioned iterative procedures, we now turn to the results of the optimization problem. The computed flow profile is given in Figure 7.3.3. Driven by the fixed potentials on Γ_4 and Γ_6, flow takes place from the stream on the right to the left hand side of the considered domain, where a perfect well extracts water. Controlled inflow occurs on Γ_2. This inflow

Table 7.3.9: Computational effort corresponding to Table 7.3.8 in Megaflops.

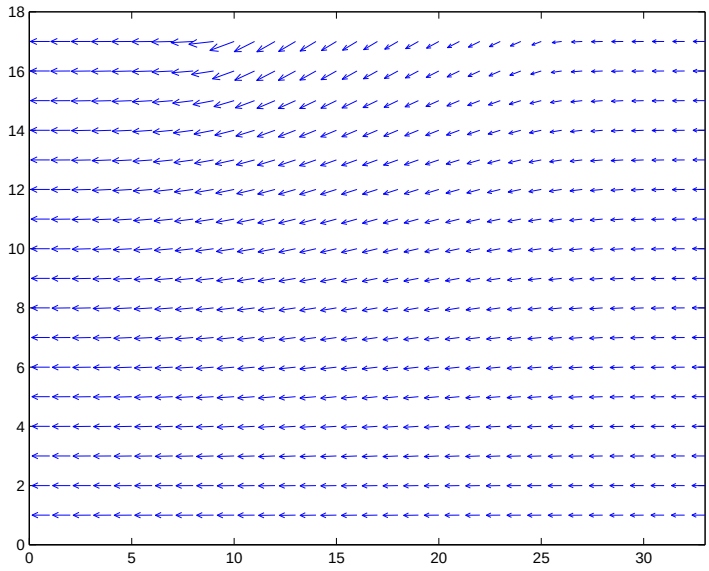
n_z	4	8	16	32	64	128
N	94	314	1138	4322	16834	66434
K	2.73	93.44	$4K$	$191K$	$\star$	$\star$
$P_A = A$, $P_H = H$	0.03	0.25	2.90	38.52	548.30	8,091.43
$P_A = A$, $P_H = L_{uu}$	0.04	0.24	2.48	30.75	418.43	5,976.24
$P_A = A$, $P_H = I$	0.04	0.27	2.61	30.77	418.49	5,976.46
$P_A = ILU(10^{-5})$, $P_H = \tilde{H}$	0.04	0.37	4.35	54.77	581.04	5,274.11
$P_A = ILU(10^{-4})$, $P_H = \tilde{H}$	0.04	0.40	4.06	41.62	395.67	3,782.59
$P_A = ILU(10^{-3})$, $P_H = \tilde{H}$	0.05	0.39	3.44	32.86	336.50	3,547.27
$P_A = ILU(10^{-2})$, $P_H = \tilde{H}$	0.06	0.39	3.47	37.23	409.94	4,690.98
$P_A = ILU(10^{-5})$, $P_H = I$	0.05	0.45	4.74	52.58	541.59	4,820.23
$P_A = ILU(10^{-4})$, $P_H = I$	0.06	0.48	4.80	42.36	387.13	3,674.33
$P_A = ILU(10^{-3})$, $P_H = I$	0.06	0.53	4.45	36.58	357.82	3,746.87
$P_A = ILU(10^{-2})$, $P_H = I$	0.08	0.63	5.00	44.89	434.35	4,961.86

is strongest on the left hand side of the considered boundary part Γ_2. See also Figures 7.3.4 and 7.3.5. Figure 7.3.4 shows the flow in x–direction, and Figure 7.3.5 shows the flow in z–direction. The maximum velocity in x–direction is about four times stronger than the maximum velocity in z–direction.

The solutions to the considered problem are given in Table 7.3.10 for different levels of grid refinement. We use $n_z = 4, 8, \ldots, 128$ discretization points in the vertical. This corresponds, due to the special choice of discretization, to just as many control variables u_i ($i = 1, \ldots, n; n = 4, 8, \ldots, 128$). The size of the full system K ranges from 94×94 to 66434×66434 variables. The computed controls u_1 on the left hand side and u_n on the right hand side of the boundary Γ_2 are given in Table 7.3.10. Also, the arithmetic average of the controls is computed. In the right columns of the table, the inflow Q_2 over the boundary Γ_2 and the extraction Q_6 on Γ_6 are given. The value of the cost function is called F in the table. A decrease in the computed values for the control can be seen with grid refinement. Accordingly, less inflow occurs on Γ_2. Smaller amounts of water are extracted on Γ_6. In the sum, the cost function increases with grid refinement for this example problem. In Table 7.3.11, relative changes in the considered quantities are shown. The rate of change in the different quantities is diminished significantly with decreasing mesh size h. This can be viewed as an indication for convergence to a mesh–independent solution.

After this detailed discussion of the numerical results in case $k = n$ we go briefly through the cases of Sections 7.2.2 and 7.2.3. The nonzero structure of K in case of a single control is given in Figure 7.3.6. The conditioning of the system matrix is better in case of a single control than in the case $k = n$. Nevertheless, the condition number

Figure 7.3.3: Flow profile.



$\kappa(K)$ still increases considerably as the mesh size h is diminished. The condition numbers of K and its submatrix A are given in Table 7.3.12.

As can be expected, the iterative solvers' performance on this system K is not significantly different from the case $k = n$. On the original system, GMRES takes an unacceptably high number of steps that is almost as large as the dimension of the system. On that same system, MINRES diverges. See Table 7.3.13 for the iteration count. The two largest systems ($n_z = 64, 128$) cannot be solved due to space restrictions. On the preconditioned system $\tilde{K}^{-1}K$ with ideal subpreconditioners $P_A = A$, $P_H = H$, GMRES takes three steps like in the previously considered situation, $k = n$. As before, MINRES is tested with the preconditioners P_1, P_2, and P_3. They all do better than in the case $k = n$, P_1 and P_2 with only nine to twelve iterations for all considered grids, and P_3 with only three iterations. See Table 7.3.13 for these results. Analogous statements hold for the cases $k = 2$ and $k = 4$. Similar to the solver's performance, analogies can be drawn for the computed solutions. The parameters that have been computed for the case $k = 1$, differ from the corresponding values for $k = n$ in Table 7.3.10 typically by less than 0.1%. The single control u in case $k = 1$ is compared to the arithmetic average of the control vector in case $k = n$. Similar statements can be made for the cases $k = 2$ and $k = 4$.

In summary, the results are very encouraging in two respects. For one, a problem from ground water modeling was formulated as a mathematical optimization problem. Specifically, we consider it as a boundary control problem governed by a partial differential equation. The results obtained for the practical application exhibit decreasing relative change with refined grids. One can hope to theoretically investigate the control

Figure 7.3.4: Flow in x–direction.

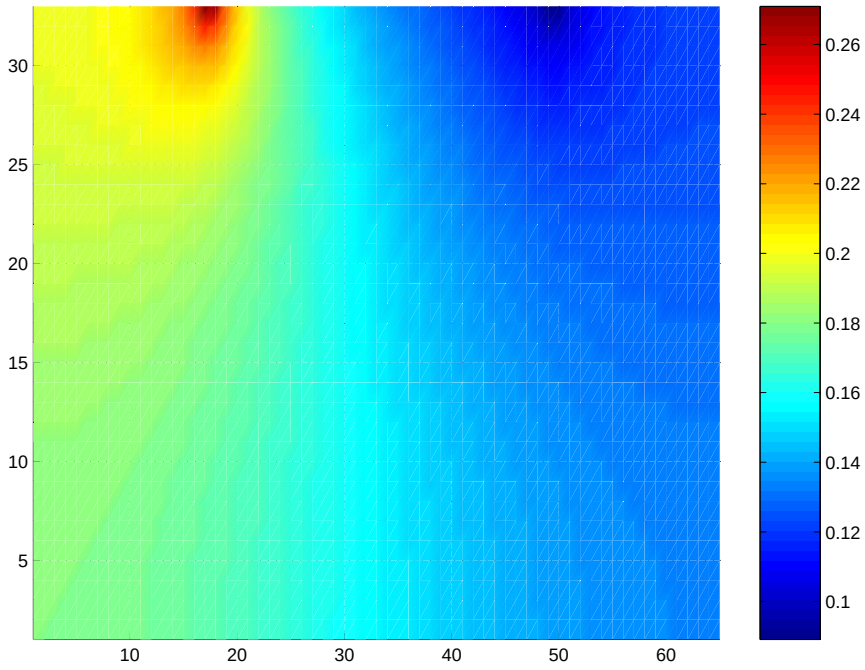


Figure 7.3.5: Flow in z–direction.

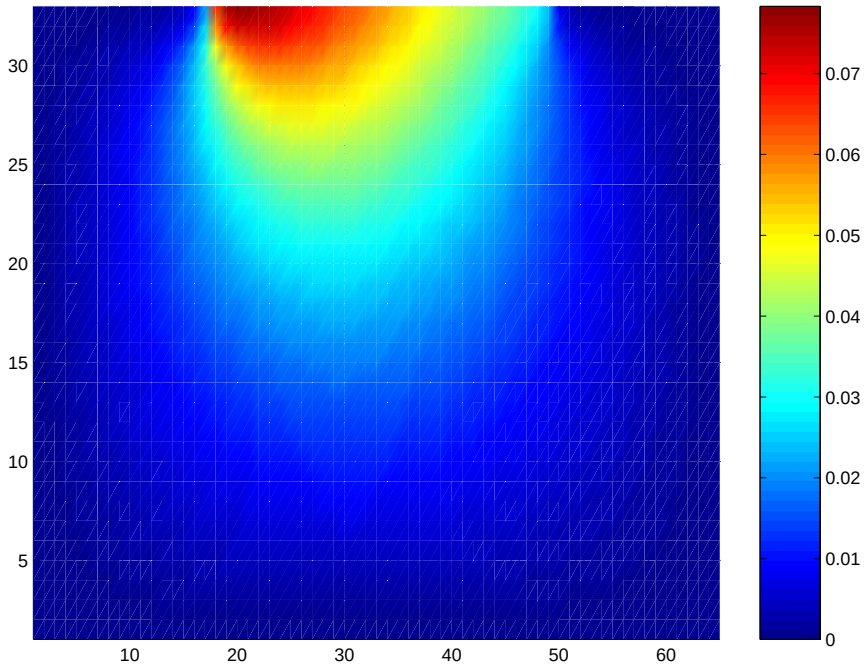


Table 7.3.10: Parameters of the problem ($k = n$).

n_z	u_1	u_n	$\frac{1}{n}\sum_{i=1}^{n} u_i$	F	Q_2	Q_6
4	26.219	22.822	24.535	−138.874	1.941	−37.149
8	25.715	22.464	24.140	−125.566	1.923	−33.689
16	25.383	22.282	23.914	−118.798	1.908	−31.916
32	25.190	22.189	23.792	−115.384	1.899	−31.019
64	25.084	22.142	23.729	−113.668	1.894	−30.567
128	25.028	22.117	23.697	−112.808	1.891	−30.340

Table 7.3.11: Relative change in the parameters of the problem ($k = n$).

n_z	u_1	u_n	$\frac{1}{n}\sum_{i=1}^{n} u_i$	F	Q_2	Q_6
$4 \to 8$	1.92%	1.57%	1.61%	9.58%	0.93%	9.31%
$8 \to 16$	1.29%	0.81%	0.94%	5.39%	0.78%	5.26%
$16 \to 32$	0.76%	0.42%	0.51%	2.87%	0.47%	2.81%
$32 \to 64$	0.42%	0.21%	0.26%	1.49%	0.26%	1.46%
$64 \to 128$	0.22%	0.11%	0.13%	0.76%	0.16%	0.74%

Table 7.3.12: Condition numbers of K and A ($k = 1$).

n_z	N	$\kappa(K)$	$\kappa(A)$
4	91	377.42	47.974
8	307	2016.56	192.475
16	1123	15488.79	770.496
32	4291	176902.40	3082.586

problem for existence and uniqueness of an optimal control.

Second, the problem was solved via its KKT system which sent us to explore efficient preconditioning techniques. Our focus are block preconditioners to exploit the structure of the system. The analysis of the indefinite preconditioner in Chapter 5.2 shows that it has interesting properties that are backed by the numerical results. In its ideal version ($P_A = A, P_H = H$), three steps of preconditioned GMRES are sufficient to stop with a very small residual for all considered grid sizes. With $P_A = A, P_H = I$ four steps furnish the iterative solution. The indefinite preconditioner compares favorably to the investigated symmetric positive definite preconditioners. The ideal version of $\tilde{K}$, with exact solutions for the constraints, is not desirable from a computational standpoint, though. Relaxation to approximate solutions requires more steps, but usually admits considerable computational savings. It is subpreconditioners P_A that furnish a usable alternative to the ideal case.

Figure 7.3.6: Structure of K for $k = 1$.

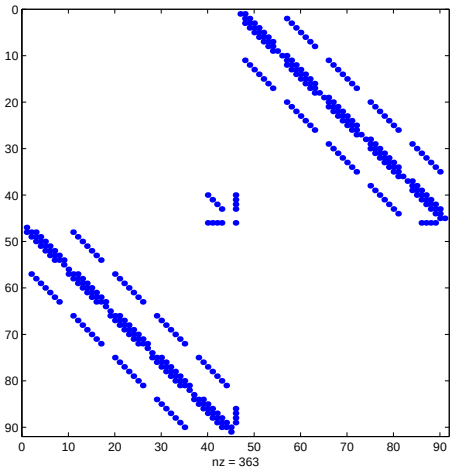


Table 7.3.13: Iterations of preconditioned GMRES and MINRES ($k = 1$). GMRES is used on system preconditioned with $\tilde{K}$ and MINRES on systems with P_1, P_2, P_3.

n_z	4	8	16	32	64	128
N	91	307	1123	4291	16771	66307
K	83	286	991	3589	$\star$	$\star$
$\tilde{K}$ ($P_A = A, P_H = H$)	3	3	3	3	3	3
$\tilde{K}$ ($P_A = A, P_H = I$)	3	3	3	3	3	3
P_1 ($P_A = A$)	9	9	9	10	11	11
P_2 ($P_A = A$)	10	10	11	11	11	13
P_3 ($P_A = A$)	3	3	3	3	3	3

Table 7.3.14: Computational effort corresponding to Table 7.3.13 in Megaflops.

n_z	4	8	16	32	64	128
N	91	307	1123	4291	16771	66307
K	2.36	86.52	$4K$	$181K$	$\star$	$\star$
$\tilde{K}$ ($P_A = A, P_H = H$)	0.03	0.23	2.29	29.12	394.69	5,680.48
$\tilde{K}$ ($P_A = A, P_H = I$)	0.03	0.22	2.25	28.86	392.60	5,663.73
P_1 ($P_A = A$)	0.07	0.41	3.50	38.91	477.36	6,321.30
P_2 ($P_A = A$)	0.07	0.44	3.86	40.19	477.60	6,461.49
P_3 ($P_A = A$)	0.03	0.30	2.85	33.59	430.37	5,965.80

Chapter 8

Conclusions and Outlook

This work is concerned with the numerical solution of optimization problems which arise in the context of ground water hydraulic and quality management. The optimization problems considered in this work are discretized optimal control problems, governed by (discretized) partial differential equations. We treat two applications in this work. These were brought to our attention by our industrial partner in this project, a consulting company for ground water and water resources.

Our first application is a ground water quality management problem. Heated water is, after use for cooling purposes, reinjected into the ground. This leads to an increase in temperature at a set of drinking water wells which are located downstream relative to the injection well. German law, the Wasserhaushaltsgesetz, requires that anthropogenic changes of ground water properties be minimized. In this regulation is the requirement that drinking water be provided at the lowest temperature that is possible under undisturbed conditions. Our goal is to control the temperature at the drinking water wells. This is done via the optimization objective to minimize a quadratic function. The quadratic involves the pumping rates at a set of barrier wells which is an approximate measure of cost. As a penalty for increased temperature, a linear combination of the pumping rates and the temperature at the drinking water wells is additionally taken into account.

In this case study, practicality enforces the usage of external, off–the–shelf software. Our industrial partner relies on software that furnishes the evaluation of the state equation in dependence of the current control. The adjoint equation is not solved by this software package. Our goal in this application is not to newly code the equation solve and, additionally, the adjoint solve. Our goal is to provide optimization tools within the given setting.

The black–box type use of external software introduces severe inaccuracies into the function. Nonsmoothness of the problem easily traps conventional optimization

algorithms. Such algorithms generally require smoothness of the involved functions and are usually also based on the assumption of accuracy in the order of machine epsilon. In addition to the before mentioned inaccuracy, large error margins are in the case of our application induced through simplifications in the modeling and the sparse data that is typical for such applications. In dealing with our problem, one is well advised to refrain from taking higher derivatives and to pay close attention when obtaining first–order information through finite–differencing. Methods for noisy functions are appropriate for our application. To our knowledge, such methods have not yet been used in the context of ground water modeling.

Specialized optimization routines are useful tools in the given setting. The optimization landscape with its nondifferentiable rim shows that optimization algorithms using first– and second–order information under the assumption of smoothness encounter difficulties. It is highly advisable in this setting to use methods that either take special care in the finite–differencing or entirely forego derivative information.

The Nelder–Mead algorithm does the latter. It is a classical direct search method which typically only requires a low number of function evaluations. At low computational expense, the Nelder–Mead algorithm achieves good progress. But the algorithm is sensitive to the starting point and can easily be trapped in one of the local minima of the problem which have large regions of attraction. Moreover, in view of possible extensions and future work, it is disadvantageous that the conception of the Nelder–Mead algorithm does not invite parallelization. Parallelization, in contrast, does hold promise for Implicit Filtering.

The Implicit Filtering algorithm does the former. It is in its simplest unconstrained form the steepest–descent algorithm with difference gradients. The difference increment is reduced in size as the iteration progresses so that in fact the algorithm is a repeated call of steepest descent. `IFFCO`, an implementation of Implicit Filtering, furnishes reliable results for the application while investing only a small budget of function evaluations.

It was mentioned above that parallelization holds promise for Implicit Filtering. Implicit Filtering is readily implemented in parallel by simply performing those function evaluations in parallel that are needed for the difference gradient. A parallel implementation of `IFFCO` exists [17]. It was used in [4] for the case study of Chapter 4 and the simulation codes described in that chapter. Future work on the application is anticipated in [4]. For one, an extension to three–dimensional geometry is of interest. The single–layered depth–integrated model set up in this work is justified because the considered region is relatively homogeneous in hydrogeologic terms. Nevertheless, a multi–layer model is likely to enhance the simulation. Moreover, the treatment laid out in this work is not limited to quadratic objective functions — a multitude of other,

more complicated objective functions describe practical problems driven by similar dynamics — and not to the simulation codes that are currently used. Other codes can be treated similarly. More general codes might be used, e.g. codes that can solve the coupled flow and transport equations. In reality, density and viscosity of the fluid do depend on the temperature. Certainly, an extension can also be thought of in terms of the optimization routine. The study of optimization methods that do not require gradients is an active research area. For pointers to the literature see e.g. [52].

Our second application is a ground water hydraulic management problem. Like the first problem, it is formulated as an optimal control problem governed by a partial differential equation. This boundary control problem, defined on a rectangular domain, occurs as a mixing problem when extracting ground water that originates from two different water bodies. One is a stream, the other a surface water body in reclaimed land. The motivation behind the problem is the controlled flooding of land for use as a relief basin when the water level is high in the stream. The height of water in the relief basin is considered as the control. The question is raised which height is advantageous. The answer must take into account that leakage from the overlying surface water occurs into the ground water body and that this incoming water is not of good quality. It requires treatment. A quadratic objective function models pumping and treatment costs.

The discretization of this problem is a quadratic programming problem with linear constraints. Quadratic problems with linear constraints are frequent in optimization. In the Introduction it was laid out that for instance such problems have to be solved as subproblems within a sequential quadratic programming approach. The solution of such linear quadratic problems can be found via the solution of the associated KKT system. In this instance, all variables, state, control, and adjoint, are considered as independent variables and simultaneously solved for. We can perform all necessary computations for this example with a self–coded discretization.

Whenever discretizations of partial differential equations are involved, the KKT systems generally tend to large dimensions and exhibit a specific sparse structure. The use of iterative solvers is usually advocated under these circumstances. Direct methods typically involve decompositions of the system matrix. Storage issues are much worked on. Iterative methods are advantageous in that they do not require to assemble the entire system K. Instead, only a matrix–vector multiplication Kv per iteration is necessary. Common choices are so called Krylov subspace methods. Two especially well–known Krylov subspace methods are the conjugate gradient and the generalized minimum residual method. We employ the latter, GMRES, and also MINRES, a method tailored to the symmetric indefinite case.

The solution of the original system K with these methods is impractical. Almost

N steps are needed by the employed methods on the original system. Here, N is the dimension of the system and also the maximum number of steps in exact arithmetic. This is a too high computational expense. Consequentially, we concern ourselves with preconditioners to accelerate the iterative solution of the linear systems. Preconditioners attempt to improve on the spectral properties of the iteration matrix such that the total number of steps required to solve the linear system within some tolerance is decreased substantially. Specifically, our concern are block preconditioners, e.g. the preconditioners $\tilde{K}$, P_1, P_2, and P_3 of Chapters 5 and 6. The preconditioners are composed of blocks and maintain the block structure of the original system. In such manner they allow to exploit that structure to computational advantage and to specifically address the problematic parts of the original system. Ideally, the blocks in the preconditioner are the inverses of the submatrices of the original system or of products of the submatrices. But it can usually not be assumed that an exact solution is affordable, not even with the submatrices. In the contrary, we must assume that the submatrix A arises from the discretization of a partial differential equation and that it is the computationally most expensive part of the system. Thus, we also account for the general case where the preconditioner for the full system is composed of preconditioners for the submatrices. The preconditioners can be perceived to approximate the respective inverses.

Several preconditioners for the symmetric indefinite KKT systems are reviewed, symmetric positive definite preconditioners and indefinite preconditioners. The analysis of the indefinite preconditioner $\tilde{K}$ in Chapter 5 shows that it has interesting properties backed by the numerical results. When this preconditioner is employed, the preconditioned system is not normal. Although convergence behavior of iterative methods is in the nonnormal case not necessarily well described by the eigenvalues, eigenvalue information does prove useful for the system preconditioned with $\tilde{K}$.

A clear–cut analysis of the minimum polynomial is performed for the case where the exact constraints are maintained. This corresponds to the choice $P_A = A$ as a block in the preconditioner $\tilde{K}$, Equation (5.2.9). In this case, the solve with A is done exactly. Realistically, the admission of approximate constraints in the preconditioner is highly desirable. This corresponds to the relaxation $P_A \approx A$. Then, the solve with A is done only approximately. This relaxation is a generalization of previous work and represents an important step towards computational efficiency. The eigenvalue analysis is disturbed in this case, but can be qualitatively maintained. It is acknowledged that more effort must be invested to fully grasp the effects of approximate constraints.

The case where the exact constraints are maintained is the starting point for analyzing the eigenvalue distribution. Then the eigenvalues of the preconditioned system are the desired eigenvalue 1 and at most k eigenvalues distinct from 1. The request eigenvalue 1 has high algebraic multiplicity $2m$ and low geometric multiplicity 2. This

delays convergence of GMRES by only one step. The remaining k eigenvalues are those of a low–dimensional subsystem involving the reduced Hessian of the underlying quadratic programming problem. The ideal version of the indefinite preconditioner lets GMRES terminate with the exact solution after 3 steps. A considerably cheaper and computationally more sensible choice can be shown to permit capturing of an invariant subspace of the system after at most $k + 2$ steps. Almost N steps, $N = 2m + k$, are needed by the employed methods on the original system. In comparison, $k + 2$ steps, with $k \ll m$, represent major improvement. The improvement is even more pronounced since the upper bound is typically considerably underbid by the actual number of steps. The preconditioner is especially successful if the number k of controls is small. The numerical experience acquired so far is encouraging, and expectations can be set high with boundary control problems.

Two additional aspects deserve mindfulness. One, we have formulated a problem from ground water modeling as a mathematical optimization problem. Specifically, a boundary control problem governed by a partial differential equation was set up. The control problem can be theoretically investigated for existence and uniqueness of an optimal control. However, if existence and uniqueness of an optimal control are shown for this specific example, an extension will be possible only to very special cases. Most commonly, the complicated geometries of real applications are prohibitive to such analyses. In addition it is desirable to incorporate the before mentioned uncertainty. The uncertainty inherent to ground water modeling problems, induced through the modeling and the typically sparse data, is not yet regarded.

Two, the problem was solved via its KKT system which sent us to explore efficient preconditioning techniques. Our focus is the construction of block preconditioners, i.e., preconditioners composed of blocks maintaining the structure of the original system. In the ideal case, the blocks in the preconditioner are the inverses of the submatrices of the original system. However, it can usually not be assumed that an exact equation solve is affordable. Approximate solutions must be accounted for. This corresponds to the general case where the preconditioner for the full system is composed of preconditioners for the submatrices. Here, further contributions and research are promising. Such block preconditioners, composed of preconditioners for submatrices, can generally be adapted to specific applications by incorporating known effective preconditioners for subsystems of the system matrix. This is certainly a viable approach to that area of active research identified in [43], “generalizing application–specific preconditioners to a broader setting”.

Bibliography

[1] O. AXELSSON, *A survey of preconditioned iterative methods for linear systems of algebraic equations*, BIT, 25 (1985), pp. 166–187.

[2] ——, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.

[3] A. BATTERMANN, *Preconditioning of Karush–Kuhn–Tucker Systems Arising in Optimal Control Problems*, Master's thesis, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, June 1996. Available electronically at `http://scholar.lib.vt.edu/theses`.

[4] A. BATTERMANN, J. M. GABLONSKY, A. PATRICK, C. T. KELLEY, T. COFFEY, K. R. KAVANAGH, AND C. T. MILLER, *Solution of A Groundwater Control Problem with Implicit Filtering*, Tech. Rep. CRSC-TR00-30, North Carolina State University, Center for Research in Scientific Computation, 2000. Submitted.

[5] A. BATTERMANN AND M. HEINKENSCHLOSS, *Preconditioners for Karush–Kuhn–Tucker Systems Arising in the Optimal Control of Distributed Systems*, in Optimal Control of Partial Differential Equations, Vorau 1996, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser Verlag, Basel, Boston, Berlin, 1996, pp. 15–32.

[6] A. BATTERMANN AND E. W. SACHS, *Block Preconditioners for KKT Systems in PDE–Governed Optimal Control Problems*, in Proceedings of the *Workshop on Fast Solutions of Discretized Optimization Problems, Berlin, 2000*, Birkhäuser Verlag, Basel, Boston, Berlin, 2000. To appear.

[7] ——, *An Indefinite Preconditioner for KKT Systems Arising in Optimal Control Problems*, Tech. Rep. Trierer Forschungsbericht 00-10, Universität Trier, Germany, 2000. Submitted.

[8] J. BEAR, *Dynamics of Fluids in Porous Media*, American Elsevier, New York, 1972.

[9] ——, *Hydraulics of Groundwater*, McGraw–Hill, New York, 1979.

[10] J. BEAR AND Y. SUN, *Optimization of pump–treat–inject (PTI) design for the remediation of a contaminated aquifer: multi–stage design with chance constraints*, Journal of Contaminant Hydrology, 29 (1998), pp. 225–244.

[11] G. BIROS AND O. GHATTAS, *Parallel preconditioners for KKT systems arising in optimal control of viscous incompressible flows*, in Proceedings of Parallel CFD '99, Williamsburg, VA, May 23–26, North Holland, 1999. To appear.

[12] P. BOGGS AND J. TOLLE, *Sequential quadratic programming*, Acta Numerica 1995, Cambrigde (1995), pp. 1–51.

[13] D. M. BORTZ AND C. T. KELLEY, *The Simplex Gradient and Noisy Optimization Problems*, in Computational Methods in Optimal Design and Control, J. T. Borggaard, J. A. Burns, E. Cliff, and S. Schreck, eds., vol. 24 of Progress in Systems and Control Theory, Birkhäuser, Boston, 1998, pp. 77–90.

[14] S. L. CAMPBELL, I. C. F. IPSEN, C. T. KELLEY, AND C. D. MEYER, *GMRES and the Minimal Polynomial*, BIT, 36 (1996), pp. 664–675.

[15] F. CHAITIN-CHATELIN, *Is nonnormality a serious difficulty?*, Tech. Rep. TR/PA/94/18, CERFACS, Toulouse, France, 1994.

[16] T. S. CHAU, *Analysis of Sustained Groundwater Withdrawals by the Combined Simulation–Optimization Approach*, Ground Water, 26 (1988), pp. 454–463.

[17] T. D. CHOI, P. GILMORE, O. E. ESLINGER, C. T. KELLEY, A. PATRICK, AND J. GABLONSKY, *IFFCO: Implicit Filtering for Constrained Optimization, Version 2*, Tech. Rep. CRSC-TR99-23, Department of Mathematics, Center for Research in Scientific Computation, North Carolina State University, July 1999.

[18] T. D. CHOI AND C. T. KELLEY, *Estimates for the Nash–Sofer preconditioner for the reduced Hessian for some elliptic variational inequalities*, SIAM J. Optim., 9 (1999), pp. 327–341.

[19] ——, *Superlinear Convergence and Implicit Filtering*, SIAM J. Optim., 10 (2000), pp. 1149–1162.

[20] T. F. COLEMAN, *Linearly Constrained Optimization and Projected Preconditioned Conjugate Gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, SIAM, Philadelphia, 1994, pp. 118–122.

[21] T. F. COLEMAN AND A. VERMA, *A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization*, tech. rep., Computer Science Department and Cornell Theory Center, Cornell University, Ithaca, NY 14850, 1998.

[22] L. COLLATZ AND W. WETTERLING, *Optimierungsaufgaben*, Springer Verlag, Berlin, Heidelberg, New York, 2nd ed., 1971.

[23] J. CONWAY, *A Course in Functional Analysis*, Springer Verlag, New York, 1990.

[24] T. B. CULVER AND C. A. SHOEMAKER, *Dynamic Optimal Control for Groundwater Remediation with Flexible Management Periods*, Water Resources Research, 28 (1992), pp. 629–641.

[25] G. DE MARSILY, *Quantitative Hydrogeology*, Academic Press, Inc., Orlando, 1986.

[26] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1983.

[27] P. A. DOMENICO AND F. W. SCHWARTZ, *Physical and Chemical Hydrogeology*, John Wiley & Sons, Inc., New York, 2nd ed., 1997.

[28] DVWK, *Stofftransport im Grundwasser (Solute transport in groundwater)*, Schriftenreihe des Deutschen Verbands für Wasserwirtschaft und Kulturbau e.V., Heft 83, DVWK, ed., Parey Verlag, 1989.

[29] L. ELSNER, *An optimal bound for the spectral variation of two matrices*, Lin. Alg. Appl., 33 (1985), pp. 77–80.

[30] M. EMBREE, *How descriptive are GMRES convergence bounds?*, Tech. Rep. Numerical Analysis Group Research Report NA–99/08, Oxford University Computing Laboratory, Oxford, England, 1999.

[31] R. W. FREUND, *Preconditioning of Symmetric, but Highly Indefinite Linear Systems*, Tech. Rep. 97–3–03, Lucent Technologies, Bell Laboratories, Murray Hill, New Jersey 07974–0636, 1997.

[32] R. W. FREUND AND F. JARRE, *A QMR–based interior–point algorithm for solving linear programs*, Math. Programming, 76 (1997), pp. 183–210.

[33] R. W. FREUND AND N. NACHTIGAL, *QMR: a quasi–minimal residual method for non–Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[34] P. E. GILL, W. MURRAY, D. B. PONCELEÓN, AND M. A. SAUNDERS, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.

[35] P. E. GILL, W. MURRAY, M. A. SAUNDERS, AND M. WRIGHT, *Inertia–controlling methods for general quadratic programming*, SIAM Review, 33 (1991), pp. 1–36.

[36] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization*, Academic Press, London, San Diego, New York, 1981.

[37] P. GILMORE, *IFFCO: Implicit Filtering for Constrained Optimization*, Tech. Rep. CRSC-TR93-7, Department of Mathematics, Center for Research in Scientific Computation, North Carolina State University, May 1993.

[38] P. GILMORE AND C. T. KELLEY, *An Implicit Filtering Algorithm for Optimization of Functions with many Local Minima*, SIAM J. Optim., 5 (1995), pp. 269–285.

[39] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore and London, 1989.

[40] S. GORELICK, *A Model for Managing Sources of Groundwater Pollution*, Water Resources Research, 18 (1982), pp. 773–781.

[41] S. M. GORELICK, C. I. VOSS, P. E. GILL, W. MURRAY, M. A. SAUNDERS, AND M. H. WRIGHT, *Aquifer Reclamation Design: The Use of Contaminant Transport Simulation Combined with Nonlinear Programming*, Water Resources Research, 20 (1984), pp. 415–427.

[42] N. GOULD, M. HRIBAR, AND J. NOCEDAL, *On the solution of equality constrained quadratic programming problems arising in optimization*, Tech. Rep. RAL–TR–1998–069, Rutherford Appleton Laboratory, Oxfordshire, England, 1998.

[43] A. GREENBAUM, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[44] R. GUENTHER AND J. LEE, *Partial Differential Equations of Mathematical Physics and Integral Equations*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[45] W. HACKBUSCH, *Theorie und Numerik elliptischer Differentialgleichungen*, B. G. Teubner Verlag, Stuttgart, 1986.

[46] M. HEINKENSCHLOSS, *Optimization methods for optimal control problems*, 1997. Lecture Notes, Summer School on Continuous Optimization, Technical University Hamburg–Harburg, Germany.

[47] I. C. F. IPSEN, *A Note on Preconditioning Non–Symmetric Matrices*, SIAM J. Sci. Comp., (2000). To appear.

[48] P. JUSTEN, *Optimal Control of Thermally Coupled Navier–Stokes Equations in Food Industry*, PhD thesis, University of Trier, Germany, 1999.

[49] C. KELLER, N. GOULD, AND A. WATHEN, *Constraint Preconditioning for Indefinite Linear Systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[50] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[51] C. T. KELLEY, *Detection and Remediation of Stagnation in the Nelder–Mead Algorithm Using a Sufficient Decrease Condition*, SIAM J. Optim., 10 (1999), pp. 43–55.

[52] C. T. KELLEY, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[53] A. KLAWONN, *An Optimal Preconditioner for a Class of Saddle Point Problems with a Penalty Term*, SIAM J. Sci. Comput., 19 (1998), pp. 540–552.

[54] ——, *Block–Triangular Preconditioners for Saddle Point Problems with a Penalty Term*, SIAM J. Sci. Comput., 19 (1998), pp. 5172–184.

[55] J. C. LAGARIAS, J. A. REEDS, M. H. WRIGHT, AND P. E. WRIGHT, *Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions*, SIAM J. Optim., 9 (1998), pp. 112–147.

[56] A. LARABI AND F. DESMEDT, *Solving three–dimensional hexahedral finite element groundwater models by preconditioned conjugate gradient methods*, Water Resources Research, 30 (1994), pp. 509–521.

[57] T. LEGE, O. KOLDITZ, AND W. ZIELKE, *Strömungs– und Transportmodellierung*, BGR Bundesanstalt für Geowissenschaften und Rohstoffe, ed., Springer Verlag, Berlin, Heidelberg, New York, 1983.

[58] J. LIESEN, *Computable Convergence Bounds for GMRES*, SIAM J. Mat. Anal. Appl., 21 (2000), pp. 882–903.

[59] M. G. MCDONALD AND A. W. HARBAUGH, *A modular three-dimensional finite-difference groundwater flow model*, U.S. Geological Survey Techniques of Water Resources Investigations, Book 6, Ch. A1, Reston, VA, 1988.

[60] K. I. M. MCKINNON, *Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point*, SIAM J. Optim., 9 (1998), pp. 148–158.

[61] M. F. MURPHY, G. H. GOLUB, AND A. J. WATHEN, *A Note on Preconditioning for Indefinite Linear Systems*, SIAM J. Sci. Comp., 21 (2000), pp. 1969–1972.

[62] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[63] S. NASH AND A. SOFER, *Preconditioning reduced matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 47–68.

[64] J. A. NELDER AND R. MEAD, *A Simplex Method for Function Minimization*, Computer Journal, (1965), pp. 308–313.

[65] A. OLIVEIRA AND D. SORENSEN, *A new class of preconditioners for large–scale linear systems from interior point methods for linear programming*, Tech. Rep. Rice CAAM TR97–27, Rice University, Houston, Texas, 1997. In Review for SIAM J. Opt.

[66] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[67] S. S. PAPADOPULOS, *MT3D: A modular three-dimensional transport model, Version 1.5, Documentation and User's Guide*, S. S. Papadopulos & Associates, Inc., Bethesda, Maryland, 1992.

[68] W. PELKA AND W. BOGACKI, *Mathematisch–numerische Verfahren zur Lösung von Optimierungsproblemen im Grundwasser (Numerical methods for the solution of optimization problems in ground water)*, Mitteilungen Institut für Wasserbau und Wasserwirtschaft, 41 (1982), pp. 205–244.

[69] T. RUSTEN AND R. WINTHER, *A preconditioned iterative method for saddle-point problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[70] Y. SAAD AND M. H. SCHULTZ, *GMRES: A Generalized Minimal Residual Algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[71] H.-C. SCHÖPFER, *Erfassung und Beurteilung der Transportvorgänge im Grundwasser am Beispiel der Pflanzenschutzmittel (Investigation and evaluation of transport processes in groundwater with special emphasis on pesticides)*, PhD thesis, Fachgebiet Wasserbau und Wasserwirtschaft, Universität Kaiserslautern, 1994.

[72] V. SCHULZ, *Incomplete indefinite decompositions as multigrid smoothers for KKT systems*, Internat. Series in Numerical Math. Vol. 133 (1999), pp. 257–266.

[73] V. SCHULZ AND G. WITTUM, *Transforming smoothers for optimization saddle-point problems*, Tech. Rep. 99 - 07, Interdisciplinary Center for Scientific Computing, Univ. of Heidelberg, Germany, 1998.

[74] D. SILVESTER AND A. WATHEN, *Fast iterative solution of stabilized Stokes systems part I: using simple diagonal preconditioners*, SIAM J. Numer. Anal., 30 (1992), pp. 630–649.

[75] ——, *Fast iterative solution of stabilized Stokes systems part II: using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1414.

[76] T. SÖLL, *Berechnungsverfahren zur Abschätzung anthropogener Temperatur–anomalien im Grundwasser (Computational methods for the estimation of anthropogenic temperature anomalies in groundwater)*, Institut für Wasserbau, Universität Stuttgart, Heft 67, Stuttgart, 1988.

[77] T. SÖLL AND H. KOBUS, *Modellierung des grossräumigen Wärmetransports im Grundwasser (Modeling of large-scale heat transport in groundwater)*, in Schadstoffe im Grundwasser, Bd. 1, Wärme– und Schadstofftransport im Grundwasser, H. Kobus, ed., DFG, Deutsche Forschungsgemeinschaft, Weinheim, Basel, Cambridge, New York, 1992, pp. 81–133.

[78] G. STEWART AND J. SUN, *Matrix Perturbation Theory*, Academic Press, San Diego, 1990.

[79] J. STOER, *Solution of large linear systems of equations by conjugate gradient type methods*, in Mathematical Programming, The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer Verlag, Berlin, Heidelberg, New York, 1983.

[80] J. W. THOMAS, *Numerical Partial Differential Equations*, Finite–Difference Methods, Springer–Verlag, Texts in Applied Mathematics 22, New York, 1995.

[81] B. J. WAGNER, *Recent advances in simulation–optimization groundwater management modeling*, in U.S. National Report to IUGG, 1991–1994, R. Vogel, ed., vol. 33, Reviews of Geophysics, 1995, pp. 1021–1028.

[82] M. H. WRIGHT, *Interior point methods for constrained optimization*, in Acta Numerica 1992, A. Iserles, ed., Cambridge University Press, Cambridge, London, New York, 1992, pp. 341–407.

[83] Y. XIANG, J. F. SYKES, AND N. R. THOMSON, *Optimization of Remedial Pumping Schemes for a Ground–Water Site with Multiple Contaminants*, Ground Water, 34 (1996), pp. 2–11.

[84] E. ZEIDLER, *Nonlinear Functional Analysis and its Applications I*, Springer Verlag, New York, Berlin, Heidelberg, Tokyo, 1986.

Appendix

This appendix is designed to provide the reader who is not familiar with ground water flow with some of the basic definitions related to aquifers and with some basic assumptions used in the derivation of the common model. The presentation is based on the book by Bear [9].

Ground Water and Aquifers

The purpose of this section is to introduce some of the basic definitions related to aquifers and to focus the attention on the flow of water in the saturated zone. The concept of essentially horizontal flow in aquifers, the so called hydraulic approach, is introduced as a powerful simplification justified in most situations of flow in aquifers.

Classifications

Ground Water, or subsurface water, is a term used to denote all the waters found beneath the surface of the ground. An aquifer is a geologic formation that contains water and permits significant amounts of water to move through it under ordinary field conditions. Other common terms are ground water reservoir or water bearing zone.

Subsurface formations containing water may be divided vertically into several horizontal zones according to the relative proportion of the pore space that is occupied by water. Essentially one distinguishes between the zone of saturation in which all pores are completely filled with water, and an overlying unsaturated zone in which the pores contain both gases (mainly air and water vapor) and water. The term ground water is in general used to denote the water in the zone of saturation.

The saturated zone can be bounded from above either by an impervious formation or by a water table. Aquifers can be classed as confined or unconfined, depending upon the presence or absence of a water table. A confined aquifer is one bounded from above and from below by impervious formations. In a well just penetrating such an aquifer, the water level will rise above the base of the upper confining formation. This water level is called the piezometric head. The piezometric head is usually denoted by ϕ. It is a function of space and time, $\phi = \phi(x, y, z, t)$. A phreatic (or unconfined) aquifer is one in which a water table, a phreatic surface, serves as its upper boundary. The phreatic surface is an imaginary surface, at all points of which the pressure is

atmospheric. Aquifers, whether confined or unconfined, that can lose or gain water through either or both of the formations bounding them from above or below are called leaky aquifers. Although these bounding semipervious formations have a relatively high resistance to the flow of water through them, over the large horizontal areas of contact involved, significant quantities of water may leak through them into or out of an aquifer. The amount and direction of leakage is governed in each case by the difference in piezometric head that exists across the semipervious formation.

A porous medium domain is said to be homogeneous if its permeability is the same at all its points. Otherwise, the domain is said to be heterogeneous or inhomogeneous. If the permeability at a considered point is independent of direction, the medium is said to be isotropic at that point. If the permeability does depend on the direction, the medium is called anisotropic. Similar considerations apply to the hydraulic conductivity defined below.

Continuum Approach to Flow through Porous Media

In an aquifer, flow takes place through a complex network of interconnected pores. However, when dealing with flow in an aquifer, the microscopic flow pattern inside individual pores is usually not considered. Instead, some fictitious average flow is considered which takes place in the porous medium comprising the aquifer. This is called the continuum approach. The obvious reason for employing the continuum approach in flow through a porous medium is that it is practically impossible to describe in any exact manner the complicated geometry of the solid surfaces that bound the flowing fluid. Consequently, a solution at the microscopic level is precluded.

A porous medium is defined as a portion of space occupied by heterogeneous material, at least one of the phases being a persistent, possibly deformable, solid phase. The solid phase is called the solid matrix. The domain which is not occupied by a solid matrix is referred to as the pore space. From the point of view of fluid flow through the porous medium, only the effective pore space is of interest. The effective pore space is that part of the interconnected pore space that does not contain any dead–end pores. In dead–end pores no flow occurs.

For the continuum approach it is necessary that the solid matrix, and hence the pore space, be distributed throughout the domain occupied by a porous medium. Solid must be present within each representative elementary volume. The representative elementary volume is an essential step in passing from the microscopic level to the macroscopic level at which only averaged phenomena are considered. The representative elementary volume can be derived as follows. Let $U(X)$ denote a volume of a spatial domain centered at a point whose position vector is X. Let E denote the amount of some extensive property of the material system contained in U. The (average) density ρ of E over U at time t is defined by

$$\rho(X, t; U(X)) = \frac{E|_{U(X)}}{U(X)}.$$

In general, ρ is a function of size, shape and orientation of $U(X)$ at time t. In order to make ρ depend on X only, a volume $U = U_0$ must be selected that is bounded between two spheres $U_{min} = (\pi/6)l_{min}^3$ and $U_{max} = (\pi/6)l_{max}^3$ such that for U_0 it holds

$$\frac{\partial \rho(X, t; U_0(X))}{\partial U_0(X)} = 0, \qquad (\pi/6)l_{min}^3 < U_0 < (\pi/6)l_{max}^3.$$

If a range for U_0 can be found that is common to all points within a given spatial domain R, one can define a field $\rho(X, t)$ throughout R and treat R as a continuum for the property E. The length $l(E)$ in the range $l_{min} < l(E) < l_{max}$ is called the scale of continuity of E in R. If R is a multiphase system, it can be treated as a continuum in describing a process involving a set of properties E in it, provided a common scale of continuity exists for all the properties E. The volume U_0 is then the representative elementary volume of the material system within R. Certainly l_{max} must be much smaller than the dimensions of R, and l_{min} must be much larger than the size of an individual pore.

Thus, by employing the definition of a representative elementary volume the actual medium is replaced by a fictitious continuum in which values of any property may be assigned to any single point. The values assigned to a point x in the continuum, the macroscopic level of description, are averaged ones. The averaged values are taken over the representative elementary volume centered at x.

Hydraulic Approach to Flow in Aquifers

In general, flow through a porous medium is three–dimensional. However, since the geometry of most aquifers is such that they are thin relative to their horizontal dimensions, e.g. tens or hundreds of meters as compared to thousands of meters, a simpler approach can be justified, cf. [9, p. 26ff.]. According to this approach it can be assumed that the flow in the aquifer is everywhere essentially horizontal (or that it may be approximated as such), neglecting vertical flow components. Flow that is essentially horizontal is called aquifer type flow. The assumption of horizontal flow is strictly true and not only an assumption for flow in a horizontal, homogeneous, isotropic, confined aquifer of constant thickness and with fully penetrating wells. The approximation is still good when the thickness of the aquifer varies in such a way that the variations are much smaller than the average thickness.

Whenever justified on the basis of the geometry, i.e., thickness versus horizontal length, and the flow pattern, the assumption of horizontal flow greatly simplifies the mathematical analysis of the flow in the aquifer. The assumption of horizontal flow is equivalent to assuming vertical equipotentials $\phi = \phi(x, y, t)$. The error introduced in this assumption is small in most cases of practical interest, cf. [9, p. 28ff.]. We see in the following discussion how the aquifer flow equations are derived by averaging the basic three–dimensional flow equations along the thickness of the aquifer, using the assumption of vertical equipotential surfaces. This procedure is called the hydraulic approach.

The assumption of essentially horizontal flow fails in regions where the flow has a large vertical flow component as, for example, in the vicinity of partially penetrating wells or outlets in the form of springs, rivers, etc. However, even in these cases, at some distance from the source or the sink, the assumption of essentially horizontal flow is valid again. As a simple rule, one may assume aquifer–type flow at distances larger than 1.5 to 2 times the thickness of the aquifer at that vicinity. See e.g. [9, pp. 28ff., 80ff.]. At smaller distances, equipotentials are no more vertical. The flow is three–dimensional and must be treated as such. Because of its simplicity and relatively small error involved, the assumption of essentially horizontal flow is usually applied also to those relatively small parts of an investigated region where it is not strictly applicable. One must, however, be careful in making use of results derived for these parts of an investigated region.

The assumption of essentially horizontal flow can be applied also to leaky aquifers. When the hydraulic conductivity of the aquifer is much larger than that of the semipermeable layer, and the thickness of the first is much larger than that of the latter, it follows that the flow in the aquifer is essentially horizontal, while it is essentially vertical in the semipermeable layer. Compare [8, p. 26].

The assumption that the flow is essentially horizontal in unconfined aquifers is the basis for the Dupuit assumption presented below.

Ground Water Flow

This section deals with the basic law governing the motion of ground water in aquifers and with the porous matrix and aquifer properties appearing in this law. Only saturated flow is considered here. In saturated flow, water completely fills the pore space of the considered porous medium domain. The continuum approach introduced above is employed, and all variables and parameters have already their average meaning in a porous medium regarded as a continuum. The presentation follows [9, Ch. IV].

Darcy's Law

Darcy's law governs the motion of ground water in aquifers. It was derived in experiments in its one–dimensional form and can be extended to three–dimensional flow. In combination with the continuity equation presented below and further conditions, Darcy's law completely describes flow in an aquifer.

Empirical One–Dimensional Form

In 1856, Henry Darcy investigated the flow of water in vertical homogeneous sand filters in connection with the fountains of the city of Dijon, France. From his experiments he concluded that the rate of flow Q, i.e., volume of water per unit time, is

- $\diamond$ proportional to the cross–sectional area A normal to the flow,

- ◇ proportional to the difference in height $h_1 - h_2$ (measured with respect to some geometric fixed point),
- ◇ inversely proportional to the length l of the filter.

The combination of these conclusions is the Darcy formula

$$Q = K \frac{A(h_1 - h_2)}{l}. \tag{A.1.1}$$

The coefficient K of proportionality is discussed below. In Equation (A.1.1), h is the piezometric head, and $h_1 - h_2$ is the difference in piezometric head across the filter of length l. The piezometric head describes in terms of head of water the sum of pressure and potential energy of the fluid per unit weight. Thus, the quantity $(h_1 - h_2)/l$ can be interpreted as hydraulic gradient. Taking l to be unit length, we denote the hydraulic gradient by ∇h . Defining the specific discharge, q, as the volume of water flowing per unit time through a unit cross–sectional area normal to the direction of flow,

$$q = Q/A\,,$$

we obtain as another form of Darcy's law

$$q = -K\,\nabla h. \tag{A.1.2}$$

Darcy's law (A.1.1) can be extended to flow through an inclined porous medium column. The porous medium is assumed to be homogeneous. For this extension, define the piezometric head ϕ as the sum of pressure head and elevation head,

$$\phi = z + \frac{p}{\gamma}. \tag{A.1.3}$$

The length z in (A.1.3) represents the elevation head. The elevation head is potential energy per unit weight of water. The quotient p/γ appearing in (A.1.3) is called the pressure head. Pressure is denoted by p and the specific weight of water by γ. The pressure head represents the pressure energy per unit weight of water of specific weight γ at that point. For a compressible fluid under isothermal conditions, the specific weight depends on pressure, $\gamma = \gamma(p)$. The pressure head is defined by

$$\frac{p}{\gamma} = \int_{p_0}^{p} \frac{ds}{\gamma(s)}.$$

Under most circumstances, however, the dependency of γ on p can be neglected. For incompressible fluids, the specific weight does not depend on pressure.

With the definition of ϕ in (A.1.3), the equations

$$Q = K \frac{A(\phi_1 - \phi_2)}{l}, \quad q = -K\,\nabla\phi \tag{A.1.4}$$

hold in analogy to (A.1.1) and (A.1.2). Note that flow takes place from a higher to a lower piezometric head and not from a higher to a lower pressure.

The energy loss $\delta\phi = \phi_1 - \phi_2$ is due to friction in the flow through the porous medium. Actually, flow takes place only through part of the cross–sectional area A of the column of porous medium that is considered for the derivation of Darcy's law. The remaining part is occupied by the solid matrix. The portion of the area A available to flow is $n_v\, A$, where n_v denotes the volumetric (or average) areal porosity. Accordingly, the field velocity v of flow through the column is defined as, see [8, p. 23],

$$v = \frac{Q}{n_v\, A} = \frac{q}{n_v}. \tag{A.1.5}$$

In some instances, the volumetric porosity n_v needs to be complemented by the concept of effective porosity, n_e. The effective porosity n_e is determined with respect to the flow through the medium, $n_e < n_v$. This is of importance when part of the fluid in the pore space is immobile, e.g. when the flow takes place in a medium where adhesion plays a role or when the porous matrix includes a large portion of dead–end pores. Then we write

$$v_e = \frac{Q}{n_e\, A} = \frac{q}{n_e}.$$

Note that both the specific discharge q and the field velocity v (or simply velocity) are averaged values, their definition justified by the continuum approach. They will in general not coincide with the actual local velocity of the fluid at the microscopic level.

Extension of Darcy's Law

The experimentally derived equation of motion in the form of Darcy's law (A.1.4) is limited to one–dimensional flow of a homogeneous incompressible fluid. When the flow is three–dimensional, the generalization of (A.1.4) and (A.1.5) is

$$q = -K\,\nabla\phi, \quad v = \frac{q}{n_v}. \tag{A.1.6}$$

Here, v is the velocity vector, q the vector of specific discharge, and $\nabla\phi$ the hydraulic gradient, all with components in the directions x, y, z of Cartesian coordinates. The coefficient K is discussed below. The extension of the one–dimensional equation (A.1.4) of motion to three–dimensional flow is justified in [8, p. 104].

Range of Validity

As the specific discharge, q, increases, the relationship between this discharge and the hydraulic gradient $\nabla\phi$ gradually deviates from the linear relationship that is expressed by Darcy's law (A.1.6). Therefore it is necessary to define a range of validity for Darcy's linear law.

In flow through conduits, the Reynolds number Re is used as a criterion to distinguish between laminar flow occurring at low velocities and turbulent flow occurring at higher velocities. The Reynolds number is a dimensionless number expressing the

ratio of inertial to viscous forces acting on the fluid. By analogy, a Reynolds number is defined also for flow through porous media by

$$Re = \frac{q\,l}{\nu}.$$

Here, l is some representative length of the porous matrix and ν is the kinematic viscosity of the fluid.

Bear [9, p. 66], states that most ground water flow occurs at Reynolds numbers Re well within the laminar flow range. In this range the linear Darcy law (A.1.6) is applicable. Although flow at large Re may sometimes occur, and although there may also exist a lower limit to the validity of Darcy's law, this phenomenon is accordingly considered to be of no significance in aquifer flows of practical interest. In both applications that are worked through in this thesis, the Reynolds numbers are well below 10, i.e., in the laminar range.

Hydraulic Conductivity

The coefficient of proportionality appearing in the various forms of Darcy's law discussed above is called hydraulic conductivity. Depending on whether the underlying porous medium is isotropic or anisotropic, the hydraulic conductivity K is either a scalar or a (3×3)–tensor. Depending on whether the underlying porous medium is homogeneous or heterogeneous, the hydraulic conductivity K is either a constant or varies in space.

An isotropic medium is a medium in which the permeability is independent of direction. Then the hydraulic conductivity can be defined using (A.1.6) as the specific discharge per unit hydraulic gradient. It is a scalar with dimension l/t, where l denotes unit length and t unit time. (In German literature, the hydraulic conductivity in the isotropic case is in general denoted by k_f, while in English literature the upper case letter K is used, cf. [9], [57].)

The hydraulic conductivity expresses the ease with which a fluid is transported through a porous matrix. It is, therefore, a coefficient which depends on both matrix and fluid properties. The relevant fluid properties are density ρ and kinematic viscosity ν. The relevant solid matrix properties are mainly grain or pore size distribution, shape of pores or grains, tortuosity, specific surface and porosity. These determine the permeability k_0 of the porous matrix which has dimension l^2. (Typical values of hydraulic conductivity and permeability for various types of soil are given in e.g. [9, p. 68], [57, p. 42].) In a homogeneous medium, $k_0 = const.$ In a heterogeneous medium, the permeability varies in space and must be conceived as $k_0 = k_0(x, y, z)$. The hydraulic conductivity can be expressed as

$$k_f(x, y, z) = \frac{\rho\, g}{\nu}\, k_0(x, y, z),$$

where g denotes the gravitational constant. We write $k_f(x, y, z) = k_f$ in the homogeneous case because then the hydraulic conductivity is independent of the spatial variables.

Under anisotropic conditions, the hydraulic conductivity is a (3×3)–matrix. In a homogeneous medium, the entries are constant; in an inhomogeneous medium, the coefficients may vary in space.

Dupuit Assumption for an Unconfined Aquifer

An unconfined aquifer is defined above as an aquifer in which a water table, the phreatic surface, serves as its upper boundary. The phreatic surface is an imaginary surface at all points of which the pressure is atmospheric. The piezometric head ϕ varies from point to point within an unconfined aquifer. Except for special cases like water at rest, the phreatic surface is not strictly horizontal and equipotential surfaces are not strictly vertical. Then $\phi = \phi(x, y, z, t)$ must be derived by solving a partial differential equation in the three–dimensional xyz space. The Dupuit assumption discussed below is equivalent to assuming horizontal flow. This allows the reduction of a three–dimensional partial differential equation to a two–dimensional one and is thus a powerful tool.

In steady flow without accretion in the vertical xz plane, the phreatic surface is a streamline. At every point P along this streamline, the specific discharge is in a direction s tangent to the streamline and is given by Darcy's law as

$$q_s = -K \frac{d\phi}{ds} = -K \frac{dz}{ds} = -K \sin(\theta) \tag{A.1.7}$$

because along the phreatic surface $p = 0$ and $\phi = z$. See Equation (A.1.3). Dupuit based his assumption in 1863 on the observation that in most ground water flows the slope of the phreatic surface is very small. Slopes of 0.1% or 1% are often encountered. The Dupuit assumption can be phrased in the following form: *The slope θ in* (A.1.7) *is negligibly small, i.e.,*

$$\theta \approx 0\,.$$

Thus, $\sin(\theta)$ *can be replaced by* $\tan(\theta) = dh/dx$. The assumption of small θ is equivalent to assuming that equipotential surfaces are vertical and that the flow is essentially horizontal. This means that $\phi = \phi(x)$ rather than $\phi = \phi(x, z)$. It is common in this case to denote the piezometric head by h rather than by ϕ. Compare Equations (A.1.3) and (A.1.1). Thus, the Dupuit assumption leads to the specific discharge in x–direction being expressed by

$$q_x = -K \frac{dh}{dx}, \quad h = h(x).$$

In general, the head h depends on both x and y so that

$$q_x = -K \frac{\partial h}{\partial x}, \quad q_y = -K \frac{\partial h}{\partial y}, \quad h = h(x, y).$$

The Dupuit assumption may be considered as a good approximation in regions where θ is indeed small and the flow is essentially horizontal. The error that is introduced is

in general small. For a discussion see e.g. [9, p. 77f.], [8, p. 363]. The important advantage gained by employing the Dupuit assumption is that $\phi = \phi(x, y, z)$ is replaced by $h = h(x, y)$, so that z does not appear as an independent variable. Also, since at a point on the free surface $p = 0$ and $\phi = h$, we assume that the vertical line through the point is also an equipotential line on which $\phi = h = \text{const}$. The Dupuit assumption cannot be applied in regions where the vertical flow component is not negligible.

Note that in general ϕ and h vary not only in space but also with time so that $\phi = \phi(x, y, z, t)$ and $h = h(x, y, t)$.

Continuity Equation

Darcy's law, governing the flow of water in confined and unconfined aquifers, has been presented in the previous section. An additional law we have to invoke is that of conservation of matter. This takes here the form of a continuity equation. The distribution of $\phi = \phi(x, y, z, t)$ in an aquifer is obtained by solving the combination of Darcy's law and this equation subject to appropriate boundary and initial conditions. In order to state the continuity equation, one has to distinguish between different types of aquifers. For the derivation of the continuity equation we refer to [9, Ch. V].

Aquifer Storativity

The continuity equation derived in this section comprises a storage term and an expression describing the flow. It is the storage term we turn to now.

The specific storativity S_0 of the porous medium of an aquifer is the volume of water δU_w released from storage or added to it in a unit volume δU_b of aquifer per unit change in piezometric head $\delta\phi$,

$$S_0 = \frac{\delta U_w}{\delta U_b \, \delta\phi} \, .$$

Accordingly, one can define a storativity S for a confined aquifer as the volume δU_w of water released from storage or added to it per unit horizontal area A of aquifer and per unit decline or rise of piezometric head $\delta\phi$,

$$S = \frac{\delta U_w}{A \, \delta\phi} \, .$$

The storativity for a confined aquifer mainly depends on water and matrix compressibility.

A storage coefficient can also be defined for an unconfined aquifer. Consider a unit (horizontal) area A of an unconfined aquifer. The volume of water stored in an unconfined aquifer is indicated by the water table. If, as a result of the flow in the aquifer, a volume of water leaves this area in excess of the volume of water entering it, the water table drops. The storativity S for an unconfined aquifer can be defined in the

same way as for a confined aquifer, except that here the drop δh is of the water table,

$$S = \frac{\delta U_w}{A\,\delta h}.$$

Basic Continuity Equation

The basic continuity equation for three–dimensional flow in a porous medium is usually developed using a control volume U of dimensions δx, δy, δz centered at some point $P(x, y, z)$. This is called Eulerian Approach, see [9, p. 89ff.]. A balance for the mass of water entering, leaving and being stored in the box can then be written. The excess of inflow over outflow per unit time and per unit volume around P is expressed by $-\nabla(\rho q)$. Here, $\rho q^T = \rho(q_x, q_y, q_z)$ denotes the mass flux of water of density ρ around point P per unit area per unit time. By the principle of mass conservation, in the absence of sources and sinks of mass, this excess of mass must equal the mass of water accumulated per unit time and per unit volume of porous medium around P. The mass balance can be written as

$$\frac{\partial \rho n_v}{\partial t} = -(\frac{\partial \rho q_x}{\partial x} + \frac{\partial \rho q_y}{\partial y} + \frac{\partial \rho q_z}{\partial z})$$

or, equivalently, in the form

$$\frac{\partial \rho n_v}{\partial t} + \nabla \cdot (\rho q) = 0. \tag{A.1.8}$$

Fluid compressibility is low. Compare [57, p. 38]. We assume ground water to be incompressible. Neglecting fluid compressibility, the constant density ρ can be omitted in (A.1.8). Taking into account the relationship between the porosity n_v and the specific storativity S_0, Equation (A.1.8) reduces to

$$S_0 \frac{\partial \phi}{\partial t} + \nabla \cdot (\rho q) = 0. \tag{A.1.9}$$

When the flow is steady, $\partial\phi/\partial t = 0$, or when both fluid and solid are incompressible, $S_0 = 0$ and ρ = const., the mass balance (A.1.8) reduces to

$$\nabla \cdot q = 0\,.$$

This states volume continuity.

The next step is to introduce the equation of motion (A.1.6) into the continuity equation (A.1.9). See [9, p. 92f.], [8, p. 205]. The continuity equation (A.1.8) can be written in terms of the single variable ϕ as

$$\nabla \cdot (K\,\nabla\phi) = S_0 \frac{\partial \phi}{\partial t}. \tag{A.1.10}$$

For a homogeneous isotropic medium, (A.1.10) reduces to

$$k_f\,\Delta\phi = S_0 \frac{\partial \phi}{\partial t} \tag{A.1.11}$$

with the Laplace operator $\Delta\phi = \nabla \cdot (\nabla\phi)$. Finally, if the flow is steady or when both fluid and solid are assumed to be incompressible, the storage term in (A.1.11) vanishes. The time–dependent equation (A.1.11) then reduces to the Laplace equation

$$k_f \, \Delta\phi = 0. \tag{A.1.12}$$

Equations (A.1.8) to (A.1.12) have been developed for a domain without sources or sinks. If sources or sinks are present, they must be represented by an additional term on the left–hand side of (A.1.8). The sink or source term expresses the rate at which mass of water is added per unit time and unit volume of porous medium around P.

Further Conditions

Each of the equations presented in the previous section is a partial differential equation that describes a class of phenomena. To obtain a particular solution corresponding to a specific problem, it is necessary to provide further specifications that are not contained in the equations. These must include:

- ◇ The geometry of the domain in which the considered flow takes place.
- ◇ Values of all relevant physical coefficients.
- ◇ *Initial conditions* which describe the initial state of the fluid in the considered flow domain.
- ◇ *Boundary conditions* describing how the fluid in the considered domain interacts with its surroundings.

Initial conditions are the specification of the piezometric head ϕ at all points within the domain at some initial time t, usually taken as $t = 0$. We denote the domain by Ω_ϕ, $\Omega_\phi \subset I\!R^3$. Initial conditions are usually written, with $f_0(.)$ on Ω_ϕ a known function, as

$$\phi(x, y, z, 0) = h_0(x, y, z) \quad \text{in } \Omega_\phi \, .$$

We now examine the boundary conditions in more detail. These describe the flow on the boundary of the considered domain. Three main types of boundary conditions are generally distinguished, the Dirichlet boundary conditions, the Neumann boundary conditions, and the mixed (or Cauchy) boundary conditions. The main types of boundary conditions encountered in flow through porous media can be grouped according to these distinctions. We consider boundary parts Γ_D, Γ_N, and Γ_C, with $\Gamma_D \cap \Gamma_N \cap \Gamma_C = \emptyset$, $\Gamma_D \cup \Gamma_N \cup \Gamma_C = \partial\Omega$. We consider as known real–valued functions $f_D(x, y, z, t)$, $f_N(x, y, z, t)$, and $f_C(x, y, z, t)$ defined on $\Gamma_D \times [0, t_1]$, $\Gamma_N \times [0, t_1]$, and $\Gamma_C \times [0, t_1]$, respectively.

- ◇ The boundary of prescribed potential corresponds to a Dirichlet boundary value problem. A boundary of this kind occurs whenever the flow domain is adjacent to a body of open water. The boundary is then an equipotential surface that can vary with time or be constant. Since the piezometric head is the same at all points

of an open water body, this piezometric head is also the boundary condition at all points on the interface between a porous medium and the water body. In this type of boundary value problem, the piezometric head ϕ is prescribed for all points of the boundary Γ_D, i.e.,

$$\phi(x,y,z,t) = f_D(x,y,z,t) \quad (x,y,z,t) \in \Gamma_D \times [0,t_1]\,.$$

◇ The boundary of prescribed flux leads to a Neumann boundary value problem. For such a boundary, the flux normal to the boundary surface is prescribed,

$$\frac{\partial}{\partial n}\phi(x,y,z,t) = f_N(x,y,z,t) \quad (x,y,z,t) \in \Gamma_N \times [0,t_1]\,.$$

A special case of this type of boundary is the impervious boundary, where the flux normal to the boundary vanishes everywhere. This can be expressed as a homogeneous Neumann boundary condition. In addition to the obvious case of an impervious boundary, such a condition is also encountered along a streamline that is used as a boundary of a flow domain, and along a water divide in an aquifer.

◇ The semipervious boundary accounts for mixed boundary conditions. This type of boundary condition occurs when the porous medium domain is in contact with a water body or another porous medium domain, but a relatively thin semipervious layer separates the two domains. An example is when a clogged river bed serves as a boundary for the flow domain. Let ϕ, as before, denote the piezometric head in the considered domain, and f_C that in the external domain. Assuming that no storage occurs in the semipervious layer, then the flux through that layer is, normal to the boundary of the considered domain,

$$\frac{\partial}{\partial n}\phi(x,y,z,t) = \frac{1}{\sigma}\left(f_C(x,y,z,t) - \phi(x,y,z,t)\right) \quad (x,y,z,t) \in \Gamma_C \times [0,t_1]\,.$$

The factor σ is the resistance of the semipermeable layer, equal to the ratio of its thickness to its hydraulic conductivity.

◇ The unsteady phreatic surface with accretion and the seepage face both are boundaries of prescribed potential.

Curriculum Vitae

A. BATTERMANN

BORN ON JULY 23RD, 1973

IN HANNOVER, GERMANY

03/1998 – 02/2001	**Dr. rer. nat.**, Universität Trier Advisor: Prof. Dr. E. W. Sachs Project "Mathematical Optimization Methods for the Remediation of Ground Water Contaminations" financed by the foundation Stiftung Rheinland–Pfalz für Innovation, realized jointly with TGU GmbH
11/1994 – 02/1998	**Diplom**, Universität Trier (*Graduate Degree in Applied Mathematics, emphasis on Numerical Analysis, Minor in Business Management*) Advisor: Prof. Dr. E. W. Sachs
08/1995 – 06/1996	**M. S.** in Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, U.S.A. Advisor: Prof. Dr. M. Heinkenschloss
10/1992 – 10/1994	**Vordiplom**, Universität Trier (*Undergraduate Degree in Applied Mathematics, Minor in Business Management*)
04/1984 – 06/1992	**Abitur**, Wilhelm–Remy–Gymnasium (Bendorf, Rhineland–Palatinat)
08/1983 – 03/1984	Schule Am Schloßpark (Stadthagen, Lower Saxony)
08/1979 – 07/1983	Grundschule Am Stadtturm (Stadthagen, Lower Saxony)