MULTIGRID METHODS IN CONVEX OPTIMIZATION WITH APPLICATION TO STRUCTURAL DESIGN

by

SUDABA AREF MOHAMMED

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Mathematics
College of Engineering and Physical Sciences
The University of Birmingham
October 2015

ABSTRACT

This dissertation has investigated the use of multigrid methods in certain classes of optimization problems, with emphasis on structural, namely topology optimization. In the first part, we have investigated the solution bound constrained optimization problems arising in discretization by the finite element method, such as elliptic variational inequalities. For these problems we have proposed a "direct" multigrid approach which is a generalization of existing multigrid methods for variational inequalities. We have proposed a nonlinear first order method as a smoother that reduces memory requirements and improves the efficiency of the resulting algorithm compared to the second order method (Newton's methods), as documented on several numerical examples.

The project further investigates the use of multigrid techniques in topology optimization. Topology optimization is a very practical and efficient tool for the design of lightweight structures and has many applications, among others in automotive and aircraft industry. The project studies the employment of multigrid methods in the solution of very large linear systems with sparse symmetric positive definite matrices arising in interior point methods where, traditionally, direct techniques are used. The proposed multigrid approach proves to be more efficient than that with the direct solvers. In particular, it exhibits linear dependency of the computational effort on the problem size.

ACKNOWLEDGEMENTS

First and foremost I would like to thank the God. You have given me the power to believe in myself. I could never have done this without the faith I have in you, the Almighty.

I take immense pleasure to express my sincere and deep sense of gratitude to my supervisor Professor Michal Kočvara. He patiently provided the vision and encouragement to proceed through the thesis program step by step, without his guidance, experience and persistent help, this work would not have been possible. I also acknowledge my co-supervisor Sándor Zoltán Németh.

Special thanks to my colleague James Turner who has been supporting me in many issues without hesitate. I would like to thank Azizah Alrashidi along with everyone I have collaborated with in the department over the past four years in particular Mrs Janette lowe. Many thanks to my husband whose patience and encouragement allowed me to continue this long journey. I wish to thank my parents, I owe them a much and wish I could show them just how much I love and appreciate them.

I would like to thank the Iraqi's ministry of higher education and scientific research for the financial support and resources to be able to complete this degree along with the staff of Iraqi Cultural Attach in the UK for their valuable support. Finally, I do really appreciate the efforts made by school of mathematics at Birmingham University for their funding in relation to conferences and workshops.

CONTENTS

1 Introduction **1**
- 1.1 Research hypothesis and objectives 6
- 1.2 Main results in the thesis 8
- 1.3 Practical impact 9

2 PRELIMINARIES **10**
- 2.1 Matrix analysis 10
- 2.2 Solution methods of linear systems $Ax = b$ 11
  - 2.2.1 Krylov subspace methods for solving linear systems 13
  - 2.2.2 The conjugate gradient method 14
  - 2.2.3 Preconditioning 20
- 2.3 Functional analysis 23

3 Multigrid Methods **31**
- 3.1 Introduction 31
- 3.2 Ingredients of multigrid method 33
  - 3.2.1 Hierarchy of Linear Systems 33
- 3.3 Finite Element Introductory Model Problem 35
  - 3.3.1 Correction Scheme 39
  - 3.3.2 Interpolation and restriction operators 41
- 3.4 Multigrid method algorithms 45
  - 3.4.1 Two-grid method 45
- 3.5 Nonlinear multigrid methods 47
  - 3.5.1 Full approximation scheme 48
  - 3.5.2 Full multigrid method 50

4 Optimization **52**
- 4.1 Basic definitions and theorems 52
- 4.2 Line search methods 57
  - 4.2.1 Step Length 58
  - 4.2.2 Convergence of generic line search method 62
  - 4.2.3 Steepest descent method 63
  - 4.2.4 Newton's methods 64
  - 4.2.5 Nonlinear conjugate gradient method 67
- 4.3 Minimization of constrained problems 69
  - 4.3.1 Projection gradient methods 69

4.3.2 Optimality conditions and step lengths 70
4.3.3 Projection on a convex set 72
4.4 Interior point methods 74
4.4.1 Primal barrier methods for constrained optimization 75
4.4.2 Interior-point for nonlinear programming 77
4.4.3 Basic interior-point algorithm 80
4.4.4 Algorithm development 82
4.4.5 Solving the primal-dual system 83
4.4.6 Updating the barrier parameter 84

5 A First-Order Multigrid Method for Bound-Constrained Convex Optimization 86
5.1 Introduction 86
5.2 Multigrid for bound-constrained optimization 89
5.2.1 The problem 89
5.2.2 Truncation 91
5.2.3 Correction scheme truncated multigrid for quadratic problems 93
5.2.4 Full approximation scheme truncated multigrid for general problems . . 95
5.2.5 Full approximation scheme multigrid without truncation 97
5.3 Equality constraints 99
5.4 Smoothing by the steepest descent method 100
5.4.1 Steepest descent smoother for unconstrained quadratic problems 101
5.4.2 Line search 104
5.5 Numerical experiments 109
5.5.1 Example: quadratic obstacle problem 111
5.5.2 Example: non-quadratic obstacle problem 113
5.5.3 Example: minimal surface problem 116
5.5.4 Example: obstacle problem with an equality constraint 118
5.5.5 To truncate or not to truncate 119
5.6 Conclusions 120

6 Structural Optimization Problem 122
6.1 Linear elasticity 122
6.1.1 Stress tensor 124
6.1.2 Strain tensor 124
6.1.3 Equations of equilibrium 127
6.1.4 Lamé equations 128
6.1.5 The classic formulation of basic boundary value problems of elasticity . 129
6.1.6 Korn‘s Inequality 131
6.1.7 Variational formulation of the elasticity problem 131
6.1.8 Existence and uniqueness of solutions to variational formulation of the elasticity problem 133
6.2 Structural Optimization Problem 134
6.2.1 Existence and Uniqueness of Variable Thickness Sheet 136
6.2.2 Discretization of Variable Thickness Sheet Problem 137
6.2.3 Weak Convergence Result 139
6.2.4 Minmax formulation of the variable thickness sheet problem 140

7 Primal-Dual Interior-Point Multigrid Method for Topology Optimization 143
7.1 Introduction . . . 143
7.2 Can direct multigrid from Chapter 5 be used for topology optimization problem? 146
7.3 Newton systems for KKT conditions . . . 148
7.4 Interior point method . . . 151
7.4.1 The algorithm . . . 152
7.4.2 Barrier parameter update . . . 152
7.4.3 Step length . . . 153
7.4.4 Stopping rules . . . 154
7.5 Optimality Conditions method . . . 154
7.5.1 OC algorithm . . . 155
7.5.2 Damped OC . . . 156
7.5.3 Averaged OC . . . 156
7.6 Numerical experiments . . . 157
7.6.1 Example shape1 (Figure 7.6) . . . 157
7.6.2 Example shape2 (Figure 7.6) . . . 160
7.7 Multigrid conjugate gradient method . . . 164
7.7.1 Multigrid method for linear systems . . . 164
7.7.2 Multigrid preconditioned conjugate gradient method . . . 166
7.8 Multigrid conjugate gradients for IP and OC methods . . . 167
7.8.1 Multigrid conjugate gradients for IP . . . 167
7.8.2 Multigrid conjugate gradients for OC . . . 169
7.9 Numerical experiments . . . 171
7.9.1 Example 1 . . . 172
7.9.2 Example 2 . . . 175
7.9.3 Example 3 . . . 177
7.10 How exact is ‘exact’? . . . 180
7.10.1 Interior point method . . . 180
7.10.2 OC method . . . 183
7.10.3 Interior point versus OC method . . . 185
7.11 Conclusions . . . 186

8 CONCLUSION AND FUTURE WORK 189

References 191

SOME BASIC NOTATION

Table 1: Notation and Symbols

Description	Notation
Domain with Lipschitz boundary	$\Omega, \mathit{\Gamma} := \partial\Omega$
$\overline{\Omega}$	$\Omega \cup \mathit{\Gamma}$
The body dimension	$d = 2, 3$
Real numbers	$\mathbb{R}$
Vector u	$u = (u_1, u_2, ..., u_n)^T$
Vector norm	$\|\|.\|\|$
absolute value	$\|.\|$
Inner product	$(.,.)$
Unit outer normal vector $\mathbf{v}$	$\mathbf{v} = (v_1, ..., v_{n-1}, v_n)$
Matrices	A, B
Hessian matrix	H
Space of square integrable functions	$L^2(\Omega)$
Sobolev spaces	$W^{k,p}(\Omega), H^k(\Omega)$
Gradient of a vector f	∇f
Set of real symmetric matrices of size $n \times n$	$S^n = \mathbb{R}^{n \times n}$
Transpose of a matrix A	A^T
Inverse of a matrix A	A^{-1}
The residuum	r
The search direction	d
The step length	α
Identity matrix $(n \times n)$	I_n or only I
The finite strain tensor	$\varepsilon(u)$
Strain tensor (small strain tensor)	$e(u)$
Stress tensor	τ
Material tensor	E
Strain energy for material E	$a_E(u, u)$
Surface force	t
Body Force	f
Work resulted by force f	$l(u)$
Thickness of a sheet	ρ
A restriction operator from the fine level to the coarse level	I_h^{2h}
An interpolation operator from the coarse level to the fine level	I_{2h}^h
Linear complementarity problems	LCP

CHAPTER 1

INTRODUCTION

Multigrid methods were originally developed for the solution of large systems of linear algebraic equations arising from discretization of partial differential equations. The aim was to accelerate the convergence of relaxation iterative methods such as Gauss-Seidel and Jacobi methods. These methods typically eliminate quickly high frequency components of the error while leaving the low frequency components. The remedy of this limitation is the pathway to multigrid procedure by involving communication between a hierarchy of levels such that the low-frequency error components at the finer level h can be restricted to the coarser level $2h$ in order to reduce the error effectively by a coarse grid correction technique. Once this coarser problem is solved, the solution interpolates back to the fine grid to correct the approximation for its low-frequency errors.

A short look at the history of multigrid methods, Fedorenko [34] introduced the first two-grid method for the Poisson equation, while Fedorenko [35] contains the first multi-grid method. Bakhvalov [7] followed that by the first more general convergence analysis. The method, however, only gained huge popularity following the seminal paper by Brandt [16] who demonstrated the tremendous computational potential of these methods. Since then a vast literature about multigrid methods has been published and introduced, and we do not try to present this. Instead we refer to the monographs [57], [115] which are particularly devoted to problems of fluid dynamics. For the interested reader, the classical texts in multigrid methods include Hackbusch and Trottenberg's Multigrid methods [60], Brandt's guide to multigrid methods [18], the introductory tutorial by Briggs et al., [20] and the comprehensive textbook by Trottenberg et al. [108], An extensive overview of multigrid methods may also be found in [114].

With regard to applying the multigrid method to optimization problems, multigrid method has been first and foremost used to solve unconstrained optimization problems; see the articles [43, 53, 82]. The concept of MG/Opt is also applied to constrained optimization models, for example see [74, 111]. In [30] multigrid method has been efficiently applied to optimization problems with differential equations. Additionally, for an interesting overview about Multigrid Methods for optimization problems with partial differential equations (PDE) constraints, we refer the reader to [14].

In reference to application of multigrid to constrained optimization problems, it has been well-known since the dark ages of multigrid that the methods may loose their superior efficiency when used for slightly different type of problems, namely the linear complementarily problems (LCP) [19, 59, 78]. This is caused by the presence of unilateral obstacles (or box constraints in the optimization formulation of the problem). The fact that the sets of active constraints may vary for different discretization levels, and that the constraints may not even be recognized on very coarse meshes, may lead to poor quality of the coarse level corrections and, in effect, to significant deterioration or even loss of convergence of the method. Various remedies have been proposed by different authors [19, 59, 62, 63, 78]; these usually resulted in "conservative" methods that were often significantly slower than standard methods for linear systems. Finally Kornhuber [69] proposed a truncated monotone multigrid method for LCP problems. This method has the property that as soon as the set of active constraints is correctly identified, the method converges with the same speed as without the presence of the constraints.

Not many attempts have been done to generalize the multigrid technique to the solution of optimization problems. From the successful ones, most focused on unconstrained problems [39, 51, 53, 73, 82]. In this case, the problem can be often identified with a discretized nonlinear PDE and thus techniques of nonlinear multigrid can be used. Treating general (equality or inequality) constraints by multigrid may be difficult, if not impossible, as we may not be able to find the corresponding restriction operators. If the number of constraints is directly proportional to the number of variables (such as for the bound constraints), the restriction operator for these constraints may be based on that for the variables. On the other hand, if the number of constraints is independent of the discretization (e.g., a single equality constraint) then the prolongation/restriction is simply the identity. All other situations are difficult, in our opinion. For this reason, all articles on multigrid for constrained problems either treat the bound-constrained problems or problems with a single equality constraint (e.g., [51, 52, 110]) or assume that a restriction operator for the constraints exists [83].

The first goal of this thesis is to extend Kornhuber's technique [69] to nonlinear convex optimization problems with bound constraints. This consists of proposing a smoothing operator that would only use first-order information, and study the efficiency of the resulting method. Finally, we extend the developed algorithm to problems with an additional equality constraints. We study the behaviour of the proposed algorithms on a number of numerical examples.

The second goal is to apply multigrid on topology optimization problem as the discipline of topology optimization offers challenging problems to researchers working in large scale numerical optimization. The results are essentially colors of pixels in a 2d or 3d "pictures". Hence, in order to obtain high-quality results, i.e., fine pictures capturing all details, a very large number of variables is essential. We will consider the basic problem of topology optimization: minimization of compliance under equilibrium equation constraints and the most basic linear constraints on the design variable:

$$\min_{\rho\in\mathbb{R}^m,\, u\in\mathbb{R}^n} f^T u \tag{1.1}$$

subject to

$$K(\rho)u = f$$

$$\sum_{i=1}^{m} \rho_i = V$$

$$\rho_i \geqslant 0, \quad i = 1,\ldots,m$$

$$\rho_i \leqslant \overline{\rho}, \quad i = 1,\ldots,m$$

where $K(\rho) = \sum_{i=1}^{m} \rho_i K_i$, $K_i \in \mathbb{R}^{n\times n}$ and $f \in \mathbb{R}^n$. We assume that K_i are symmetric and positive semidefinite and that $\sum_{i=1}^{m} K_i$ is sparse and positive definite. For further reference, we will call the design variable ρ the *density*. The most established and commonly used optimization methods to solve this problem are the Optimality Conditions (OC) method [10] and the Method of Moving Asymptotes by Svanberg [106]. In both methods, the computational bottleneck consists of the solution of a large scale linear system with a sparse symmetric positive definite matrix (the equilibrium equation). This is traditionally solved by a direct solver, such as the Cholesky decomposition. Recently, several authors proposed the use of iterative solvers, mostly a preconditioned Krylov subspace solver, such as Conjugate Gradients (CG), MINRES or GMRES. These have one big advantage which is specific for their use within optimization algorithms: in the early (or even not-so-early) stages of the optimization method, only a very low accuracy of the linear solver is needed. They also have one big disadvantage: in the late stages of the optimization method, the linear solvers become very ill-conditioned and thus a vanilla iterative solver can come into extreme difficulties.

It is therefore essential to use a good preconditioner for the Krylov subspace method. The difficulty lies in the fact that as we approach the optimal solution of the topology optimization problem, the condition number of the stiffness matrices increases significantly. In fact, it is only controlled by an artificial lower bound on the variable—if this bound was zero, the stiffness matrix would be singular. Wang et al. [113] studied the dependence of the condition number on the variables and concluded that it is a combination of the ratio of maximum and minimum density and the conditioning of a corresponding problem with constant density. Consequently, they proposed a rescaling of the stiffness matrix combined with incomplete Cholesky preconditioner. The rescaling results in constant order of condition number during the optimization iterations. For large scale example still hundreds of MINRES iterations are needed and hence the authors use recycling of certain Krylov subspaces from previous iterations of the optimization method. Recently, Amir et al. [3] proposed a multigrid preconditioner for the systems resulting from OC or MMA methods and demonstrated that the resulting linear system solver keeps its efficiency also for rapidly varying coefficient of the underlying PDE, i.e., rapidly varying ρ in (1.1).

In Chapter 2 the basic concepts related to the background of the current work are provided. Fundamental concepts from matrix analysis are explained because they are theoretically and computationally important especially in the optimization field. In the functional analysis section, all important definitions and theorems for vector spaces which are needed to define the mathematical formulation of elasticity are presented. Afterwards, we specify a section for the solution methods of a linear system of equations in order to introduce fundamental aspects of such methods which will be used in the current work, among these methods Krylov subspace methods.

Chapter 3 is concerned with elements of multigrid methods for solving linear and nonlinear system of equations which results in the solution of partial differential equations. It is an essential chapter to preface understanding the generalized versions in relation to applying multigrid methods to optimization problems.

Optimization methods for solving nonlinear constrained problems are presented in Chapter 4. This includes, line search methods, projected gradient methods and interior-points methods.

Application of multigrid to constrained optimization problems is given in Chapter 5. This is achieved by considering first order methods such as projected gradient methods, steepest methods for constrained optimization problems. Numerical examples demonstrate the efficiency of the proposed methods.

In Chapter 6 the concepts of the stress tensor, strain tensor and their relations are familiarized with introducing the formulation of basic boundary value problems of elasticity classically and formulations of the elasticity problem with converting it to topology optimization problems. Finite element discretization for the variable thickness sheet which represents a special case of topology optimization problems is discussed with the weak convergence theorem.

Finally, in Chapter 7 we propose a new method for the solution of the topology optimization problem introduced in Chapter 6. The framework of the method is given by the interior-point algorithm. The systems of linear equations, arising in this algorithm, are then solved by the conjugate gradient method preconditioned by one step of the multigrid algorithm. Extensive numerical test will show that the resulting algorithm is extremely efficient and stable, and its complexity grows only linearly with the problem size. The thesis is concluded by an outlook to future work.

1.1 Research hypothesis and objectives

The initial goal of the research was to investigate the multigrid efficiency in certain classes of optimization problems, with emphasis on structural design, namely topology optimization. With regard to applying the multigrid method to optimization problems, multigrid method has been first and foremost used to solve unconstrained optimization problems; see the articles [43,53,82]. The concept of MG/Opt is also applied to certain constrained optimization models, for example see [74,111]. In [30] multigrid method has been efficiently applied to optimization problems in differential equations. As to our knowledge, the article [76] is the first one that considered the use of multigrid techniques to topology optimization. Similarly, as in the later article [104], the authors apply the "traditional" multigrid method to systems of linear equations arising from Newton-type methods. Contrary to that, the aim of this thesis was to apply the

MG/Opt technique directly to the full topology optimization problem. Originally MG/Opt [83] has been developed for unconstrained optimization problem

$$\min_{x_h} f_h(x_h), \tag{1.2}$$

where h indicates the level in the hierarchy of models. The attempt was to apply a generalized version introduced in [83] to solve the so-called minimum compliance problem of topology optimization that can be written as follows (see [10])

$$\begin{aligned} \min_{\rho} \quad & f^T K(\rho)^{-1}(\rho) f \\ \text{subject to:} \quad & \sum_{i=1}^{m} \rho_i \leqslant V, \\ & 0 < \underline{\rho} \leqslant \rho_i \leqslant \overline{\rho}. \end{aligned} \tag{1.3}$$

where $K(\rho) = \sum_{i=1}^{m} \rho_i K_i$, $K_i \in \mathbb{R}^{n \times n}$ and $f \in \mathbb{R}^n$. We assume that K_i are symmetric and positive semidefinite and that $\sum_{i=1}^{m} K_i$ is sparse and positive definite. For further reference, we will call the design variable ρ the *density*. We will assume that there exists a hierarchy of models, organized from fine h to coarse grids $2h, 4h, 8h, ...$ with an interest in finding the solutions on the finest level. In other words, the computations on a coarse level for the problem can be used to upgrade an approximate result of a finer resolution problem. MG/Opt iteratively utilizes coarse resolution problem in order to obtain search directions for finer-resolution problem. As a result, the solution of each finer-resolution problem can be repeated by using a line search method. Then, applying a line search technique can leads to the possibility to demonstrate the convergence results for the MG/Opt algorithm [74].

In conclusion, the main target was to introduce a multigrid algorithm for the topology optimization problem. That was based on recalling the definition and convergence analysis of the MG/Opt method for unconstrained optimization problems, by Nash [83]. After that, defining and analyzing the method for the topology optimization problem. However, this could not be done easily because of reasons have been mentioned in Chapter 7 (see 7.2). Alternatively, we proposed direct multigrid with first order smoother for nonlinear constrained convex optimization problems. And also proposing conjugate gradient method preconditioned by multigrid method within interior point method for solving topology optimization problems as it given in the following section.

1.2 Main results in the thesis

The main results of the thesis can be classified into two main contributions. In the first case, multigrid methods is applied as a direct iterative method to bound constrained optimization problems. Here a first order methods (steepest descent) is proposed as a smoother within multigrid technique for the developed algorithms. Numerical experiments show that the proposed smoother eliminates the high frequency components of the error quickly in the first few iterations. The developed algorithms consider an extension of Kornhuber's [69] and Hackbusch-Mittelmann [59] to nonlinear convex optimization problems with bound constraints and an additional equality constraints. The behavior of the proposed algorithms is studied on a number of numerical examples.

The second main result represents the application of multigrid within the interior point methods for the solution of topology optimization problem. This follows the path outlined by Jarre et al. [64] and Maar-Schulz [76]. Unlike these, the linear systems which results from the Newton method within interior point method is reduced to obtain positive definite matrices. This allows to use standard conjugate gradient preconditioned by standard V-cycle multigrid. Furthermore, we use the same linear solver in the so-called OC method (in the same way suggested Amir et al. [3]) to get a comparison with interior point method. In both cases the inexact multigrid preconditioned CG method leads to a very efficient optimization solver. Most notably, in case of the interior point method we obtain an approximately constant number of CG iterations needed to solve the full problem which is independent of the size of the problem. In case of the OC method, the total number of OC iterations is increasing with the problem size; however, for a given problem size, the number of CG steps per one linear systems remains almost constant, and very low, in all OC iterations, notwithstanding the condition number of the stiffness matrix.

1.3 Practical impact

The proposed methodology allows to solve very large-scale optimization problems, in particular PDE constrained problems and topology optimization problems, that could not be solved by existing software. All these problems are of big practical importance. The first approach is the multigrid method for bound constrained convex optimimztion problems. Example of these are elliptic variational inequalities discretized by the finite element method, which have been proved extremely useful for mathematical description of a wide rang of material science, electrodynamics, continuum mechanics and many others (we refer to [48] for the literature). In addition to that, a large number of applications are covered even for the special case of obstacles problems, such as contact problems in continuum mechanics and option pricing in computational finance [92].

The project further investigates the use of multigrid techniques in topology optimization. Topology optimization is a very practical and efficient tool for the design of lightweight structures and has many applications, among others in automotive and aircraft industry. The discipline of topology optimization offers challenging problems to researchers working in large scale numerical optimization. The results are essentially colors of pixels in a 2d or 3d “pictures”. Hence, in order to obtain high-quality results, i.e., fine pictures capturing all details, a very large number of variables is essential. Therefore, the project studies the employment of multigrid methods in the solution of very large linear systems with sparse symmetric positive definite matrices arising in interior point methods where, traditionally, direct techniques are used.

CHAPTER 2

PRELIMINARIES

2.1 Matrix analysis

In this section, we present a number of results from the field of matrix analysis, primarily related to the classification of positive (semi-) definite matrices. As will be shown such matrices are required in order to provide asserts of convergence. We first present the notion of the gradient and Hessian of a real valued function in the following manner.

Definition 2.1. (Gradient and Hessian)

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable. The gradient of f at a point $x \in \mathbb{R}^n$ is defined as:

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)^T, \tag{2.1}$$

and the Hessian matrix is a symmetric $n \times n$ matrix of second derivatives, defined as:

$$H(x) = \nabla^2 f(x) := \left(\frac{\partial^2 f(x)}{\partial x_i x_j} \right), \qquad i, j = 1, 2, \dots, n. \tag{2.2}$$

The notion of positive (semi-) definiteness can be described as follows.

Definition 2.2. (Positive and Negative (Semi-) Definite Matrices)

Let $A \in \mathbb{R}^n$ be a symmetric matrix and let $x \in \mathbb{R}^n$. If $x^T Ax > 0$ ($x^T Ax \geqslant 0$) is satisfied for all $x \in \mathbb{R}^n$, $x \neq 0$, then A is called positive definite (positive semi-definite). We will use the notation $A > 0$ and $A \geqslant 0$ to denote positive definiteness and positive semi definiteness of A, respectively.

A negative (semi-) definite matrix amounts to a matrix whereby the respective inequalities presented above are reversed. In other words, a matrix A is negative (semi-) definite if $-A$ is positive definite or positive(semi-) definite. Finally, a matrix $A \in S^n$ is called indefinite if $x^T Ax$ gives both positive and negative values for some $x \in \mathbb{R}^n$.

Lemma 2.1 (Cholesky decomposition)**.** A matrix A is positive definite if and only if there is a nonsingular lower triangular matrix $L \in S^n$ with positive diagonal elements such that $A = LL^T$.

Theorem 2.1. (Schur Complement) Let $A \in S^n$ be partitioned as

$$\begin{pmatrix} A_1 & A_2 \\ A_2^T & A_3 \end{pmatrix}$$

where both A_1 and A_3 are square matrices. Then A is positive definite if and only if A_1 is positive definite and $A_3 > A_2^T A_1^{-1} A_2$.

2.2 Solution methods of linear systems $Ax = b$

The study of solution methods of the linear system of equations is a wide topic in itself which arise particularly in solving partial differential equations. However, we have not chosen to go in details here and just give an overview of methods that have been used in this work. Consider a linear system of the form

$$Ax = b, \tag{2.3}$$

where $A \in \mathbb{R}^{n\times n}$, $x, b \in \mathbb{R}^n$. Numerical methods for solving the system (2.3) can be classed as either direct methods or iterative methods, with combinations of the two (termed hybrid methods) also possible.

Direct solution methods work by considering a factorization of A into a product of matrices BC, where both of B and C are of a certain structure that can be exploited. In the case of Gaussian elimination, B and C amount to lower and upper triangular matrices L and U respectively. A finite number of operations is required to obtain a solution to (2.3). However, if A is dense, this can still be prohibitive for large n. For instance, such problems are encountered in systems arising from the discretization of partial differential equations. Whilst efficient direct methods have been developed that are able to exploit sparsity potions within A [31], recent approaches of research have focused on iterative methods. These approaches seek to determine a suitable approximate solution to (2.3) by constructing a sequence of iterations that leads to the solution to (2.3). In particular, preconditioned conjugate gradient like methods are particularly popular within the field of numerical analysis, and we will provide further details in due course.

An iterative method can generally be classed as either relaxation iterative method (also known as stationary iterative method) or Krylov subspace method. Examples of relaxation iterative methods include the Gauss-Seidel method, Jacobi method and the Successive over-relaxation method (SOR). For clarity, we describe the relaxation iterative methods in a general context. They consist of constructing the following fixed point iteration to solve linear systems of the form (2.3):

$$x^{k+1} = Bx^k + c,$$

based on a specific definition of B and c, where neither the matrix B nor the vector c depend on the iteration count k. Consider a more useful illustration of this general construction, the Jacobi method. The matrix A is divided into two parts, diagonal matrix D whose diagonal elements identical to those of A and whose off-diagonal elements are zero; and the matrix E, whose diagonal elements are zero, and whose off-diagonal elements are identical to those of A.

Thus, $A = D + E$. The method is derived as

$$\begin{aligned} Ax &= b \\ Dx &= -Ex + b \qquad (2.4) \\ x &= -D^{-1}Ex + D^{-1}b \qquad (2.5) \\ x &= Bx + c, \text{ where } B = -D^{-1}E, \; c = D^{-1}b. \qquad (2.6) \end{aligned}$$

It is easy to invert the diagonal matrix D. However, for discretized boundary value problems, one may obtain a slow convergence rate. For a more rigorous explanation regarding these methods including convergence analysis, we refer to [119].

However, the Krylov subspace methods examples include conjugate gradients method, these methods are based on generating a basis of the so-called Krylov subspace

$$\text{span}\{v, Av, A^2v, ..., A^{m-1}v\}, m \in \mathbb{N}, m \leqslant n, \qquad (2.7)$$

as before, these methods seek an approximate solution to (2.3) from this subspace in which iterates involve projecting the residual onto lower dimensional Krylov subspaces. Krylov subspace methods have been applied successfully for solving eiginvalue problems and for solving linear systems. For further study about Krylov subspace solvers we suggest recent books [54, 97] as well as the overview paper [96].

2.2.1 Krylov subspace methods for solving linear systems

We recall again the linear system $Ax = b$ with an initial guess x^0. In general, projection methods seek an approximate solution x^m from an affine subspace $x^0 + K_m$ of dimension m satisfying the orthogonality

$$b - Ax^m \perp L_m \qquad (2.8)$$

where L_m is a certain subspace of dimension m which is specified in this section. In reference to a Krylov subspace method, the subspace K_m denotes the Krylov subspace such that

$$K_m(A, r^0) = \text{span}\{r^0, Ar^0, A^2r^0, ..., A^{m-1}r^0\}, \tag{2.9}$$

in which $r^0 = b - Ax^0$. We give a short description about two particular choices of L_m which will be of interest in this work.

The first choice is corresponding to $L_m = K_m$. In the case where the matrix A is symmetric positive definite, an appropriate inner product can be defined allowing optimality properties to be considered in relation to the norm of the error, namely $\|e^m\|_A := \|x^* - x^m\|_A$. Such methods are termed error projection methods, with the Conjugate Gradient (CG) method [61] the most widely used. Other examples include, the Orthogonal Residual (ORTHORES) method [120] and also the Full Orthogonalization Method (FOM) [95].

The second case corresponds to the choice $L_m = AK_m$. Here, the approximate solution x^m will minimize the residual norm $\|b - Ax\|_2$ over the affine space $x^0 + K_m$ (as it shown, for example, in [98]). Unlike the previous choice, a number of methods have been developed in this case for non-symmetric matrices, as described, for instance, in [4, 33]. Two particular methods which are widely used are the Minimum Residual Method MINRES, pioneered by Paige and Saunders [87] for symmetric indefinite matrices, and the generalized Minimum Residual Method GMRES presented by Saad and Schultz [99] for non-symmetric indefinite matrices.

2.2.2 The conjugate gradient method

Conjugate gradient CG is theoretically a finite method. It gets exact solution in n steps. Round-off error cause loss of orthogonality, making it iterative. Here, we provide a brief introduction to the conjugate gradient method. For a more detailed presentations the interested reader should consult [118, Chapter 1]. Hestenes and Stiefel [61] proposed the linear conjugate gradient method as an iterative method for solving linear systems of the form (2.3) in the case when the matrix A is symmetric positive definite. It is well suited for solving large problems as an iterative alternative to Gaussian elimination. Fletcher and Reeves [38] were the first to introduce a nonlinear conjugate gradient method for solving large scale nonlinear optimization problems, for a historical overview we suggest [5, p.451] and references therein.

Conjugate gradient methods, in general, were introduced to accelerate the convergence rate of steepest descent. They are based on the aim to determine the solution to the following unconstrained quadratic problem

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T A x - b^T x, \tag{2.10}$$

where A is a symmetric positive definite matrix and $b \in \mathbb{R}^n$ is a given vector. A being symmetric positive definite, the problem (2.10) is convex so we can grantee existence of a unique global minimizer. As we consider an iterative approach, we denote by $r(x^k)$ the residuum at a point x^k we get

$$r(x^k) := r^k = b - Ax^k. \tag{2.11}$$

For the quadratic definition (2.10), residuum is equal to the negative gradient of f, $-\nabla f(x)$. Which is in turn represents a direction of the steepest descent procedure in which f decreases quickly. The key feature of the conjugate gradient method is its ability to construct A-conjugate directions of nonzero vectors $d_0, ..., d_k$ with respect to the symmetric positive definite matrix A

$$d_i^T A d_j = 0, \quad \text{for all } i \neq j. \tag{2.12}$$

The line search method

$$x^{k+1} = x^k + \alpha_k d_k, \tag{2.13}$$

with an initial point $x^0 \in \mathbb{R}^n$ and a set of conjugate directions $d_0, d_1, ..., d_{n-1}$ represents the search direction algorithm. In case of linear systems, the step length α_k for the quadratic problem $f(x)$ along $x^k + \alpha_k d_k$ can be written explicitly as

$$\alpha_k = \frac{r_k^T d_k}{d_k A d_k}. \tag{2.14}$$

See [101, p. 23] for the derivation of the step length α. Convergence of the algorithm is assured via the following theorem.

Theorem 2.2. For $x^0 \in \mathbb{R}^n$, the conjugate direction algorithm (2.13), (2.14) with d_k satisfying (2.12) generates the sequence $\{x^k\}$ which converges to the solution x^* of the linear system $Ax = b$ in at most n iterations.

Proof. The proof is given in [118, pp.103-104]. □

There are a number of ways to choose the set of conjugate directions, with one example corresponding to the eigenvectors of the matrix in question. However, for large scale problems, a substantial amount of computation maybe needed in order to compute the complete set of eigenvectors. Alternatively, modified Gram-Schmidt orthogonalization process [13, Chapter 1] can present an approach to generate a set of conjugate (A-orthogonal) directions instead of a set of orthogonal directions. The latter approach, however, is also expensive as it is necessary to store all previously computed directions.

In the conjugate direction method each direction d_k is selected as a linear combination of the steepest descent directions calculated at the previous points $x^0, ..., x^k$ for the function f. Based on r^k defined in by (2.11), we use definition of previous direction d_{k-1} to write

$$d_k = r^k + \beta_k d_{k-1}, \tag{2.15}$$

where the scalar β_k is determined depending on the necessity of orthogonality of d_k and d_{k-1} with respect to A. Then, multiplying (2.15) by $d_{k-1}^T A$ and applying the conjugacy condition

$d_{k-1}^T A d_k = 0$, yields

$$\beta_k = -\frac{(r^k)^T A d_{k-1}}{(d_{k-1})^T A d_{k-1}}.$$

The first search direction d_0 is the steepest descent direction at the initial point x^0. Thus the preliminary version of the conjugate gradient is expressed formally as follows.

Algorithm 2.1. Select x^0 as an initial point;

Set $r^0 = b - Ax^0$, $d_0 = r^0$ and $k = 0$;

Do while $r^k \neq 0$

$$\alpha_k = \frac{(r^k)^T d_k}{d_k A d_k}; \tag{2.16}$$

$$x^{k+1} = x^k + \alpha_k d_k; \tag{2.17}$$

$$r^{k+1} = b - Ax^{k+1}; \tag{2.18}$$

$$\beta_{k+1} = -\frac{(r^{k+1})^T A d_k}{d_k^T A d_k}; \tag{2.19}$$

$$d_{k+1} = r^{k+1} + \beta_{k+1} d_k; \tag{2.20}$$

$$k = k + 1; \tag{2.21}$$

end(while)

This algorithm is a useful version for studying basic properties of the conjugate gradient, however a more efficient version is presented later. Based on the description provided above, we present the following theorem

Theorem 2.3. Assume that the k-th iterate produced by the conjugate gradient Algorithm 2.1 is such that $\|x^k - x^*\|_2 > \varepsilon$ for suitably small $\varepsilon > 0$. Then following four properties are satisfied

$$r_k^T r_i = 0, \text{ for } i = 0, 1, ..., k-1, \tag{2.22}$$

$$\text{span}\{r^0, r^1, ..., r^k\} = \text{span}\{r^0, Ar^0, ..., A^k r^0\}, \tag{2.23}$$

$$\text{span}\{d_0, d_1, ..., d_k\} = \text{span}\{r^0, Ar^0, ..., A^k r^0\}, \tag{2.24}$$

$$d_k^T A d_i = 0 \text{ for } i = 0, 1, ..., k-1, \tag{2.25}$$

Consequently, the sequence converges to the solution x^* in at most n iterations.

The proof is given by induction and relies on the fact that the first direction d_0 is the steepest descent direction r^0 which is distinct from other choices of d_0, as shown in [118, pp.109-111]. The specifics of the theorem indicate some useful properties. In particular, since the residuals r^i are mutually orthogonal, each of residuals r^k and associated search directions d_k are included in the Krylov subspace $K_m(A, r^0)$. Furthermore, the fact that the Algorithm generates a set of conjugate search directions $d_0, d_1, ..., d_{k-1}$ guarantees convergence in at most in n steps.

A practical form of the conjugate gradient method is derived based on Theorem 2.3 using the fact that the current residual r^k is orthogonal to all previous search directions (as presented in [118, Theorem 5.2]), namely that

$$r_k^T d_i = 0, \text{ for } i = 0, 1, ..., k-1, \tag{2.26}$$

Therefore the step length presented in (2.16) maybe reformulated by using (2.20) to yield

$$\alpha_k = \frac{(r^k)^T r^k}{d_k A d_k}.$$

We now substitute our definition of the residual r^k into the search direction (2.17) so that

$$r^{k+1} = r^k - \alpha_k A d_k.$$

Under the expressions for both d_k and r^{k+1}, we are able to write

$$\beta_{k+1} = \frac{(r^{k+1})^T r^{k+1}}{(r^k)^T r^k},$$

which we now use in order to present the standard form of the conjugate gradient for linear systems as follows.

Algorithm 2.2. Select x^0 as an initial point;

```
Set $r^0 = b - Ax^0$, $d_0 = r^0$ and $k = 0$;

Do while $r^k \neq 0$

        $\alpha_k = \frac{(r^k)^T r^k}{d_k A d_k}$;                  (2.27)
        $x^{k+1} = x^k + \alpha_k d_k$;                               (2.28)
        $r^{k+1} = r^k - \alpha_k A d_k$;                             (2.29)
        $\beta_{k+1} = \frac{(r^{k+1})^T r^{k+1}}{(r^k)^T r^k}$;      (2.30)
        $d_{k+1} = r^{k+1} + \beta_{k+1} d_k$;                        (2.31)
        $k = k + 1$;                                                  (2.32)

end(while)
```

The purpose behind the reformulated algorithm results is that it is no longer necessary to retain the vectors x, r and d for any more than two previous iterations, meaning reduced storage costs when compared to Algorithm 2.1.

In terms of the rate of convergence, Theorem 2.2 guarantees that the algorithm will terminate in at most n steps. However, convergence is also dependent on the distribution of the eigenvalues of A, as stated in the following Theorem.

Theorem 2.4. If A has m distinct positive eigenvalues, then the conjugate gradient method will terminate in at most m iterations.

Proof. Omitted, please see [118]. □

Notice that these are theoretical results relying on exact arithmetics. In computer implementation, we calculate with finite precision arithmetics resulting in round-off errors and loss of orthogonality of d_k, r^k generated by Algorithm 2.1.

Therefore, a useful characterization of the behavior of the conjugate gradient method is given through the following estimate, initially derived by Luenberger [75].

Theorem 2.5. If A has eigenvalues $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_n$, then the following inequality is satisfied

$$\|x^{k+1} - x^*\|_A^2 \leqslant \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right) \|x^0 - x^*\|_A^2, \tag{2.33}$$

where $\|x\|_A^2 = x^T Ax$.

2.2.3 Preconditioning

Preconditioning of a linear system $Ax = b$ is a technique that is typically related to reducing the condition number of a matrix A, $\kappa(A)$ or to reducing the number of distinct clusters of eigenvalues. That is by solving alternatively the following scaled system

$$M^{-1}Ax = M^{-1}b, \tag{2.34}$$

assuming that M is a symmetric, positive-definite matrix that approximates A, but it is easier to be inverted. It is expected that, the iterative methods can solve the system (2.34) more quickly than the original problem if $\kappa(M^{-1}A) \ll \kappa(A)$ or if the eigenvalues of $(M^{-1}A)$ are better clustered than those of A, see [101]. However, $M^{-1}A$ is not generally symmetric nor definite, even if the matrices M and A are. The remedy of this is the use of the fact that, for every symmetric, positive-definite M there is a matrix E (not necessarily unique) that satisfies the property $EE^T = M$. For instance, the matrix E can be obtained by Cholesky factorization.

Furthermore the eigenvalues of the matrices $M^{-1}A$ and $E^{-1}AE^{-T}$ are equal according to the fact that if v is an eigenvector of $M^{-1}A$ with eigenvalue λ, then $E^T v$ is an eigenvector of $E^{-1}AE^{-T}$ with eigenvalue λ

$$(E^{-1}AE^{-T})(E^T v) = E^T(E^{-T}E^{-1})Av = E^T M^{-1}Av = \lambda E^T v.$$

Therefore, we can transform the problem $Ax = b$ as follows

$$M^{-1}Ax = M^{-1}b,$$

by substituting $M = EE^T$ and multiplying both sides by E^T, we get the following scaled system

$$Bz = c; \text{ such that } B := E^{-1}AE^{-T} \;\; z := E^T x, \text{ and } c := E^{-1}b.$$

We can solve this system first for z, then for x, that is by the use of steepest descent or conjugate gradient method as $E^{-1}AE^{-T}$ is symmetric and positive definite. The process of using CG to solve this scaled system is referred to as transformed preconditioned conjugate gradient method, see [101]. The procedure is as follows

$$\text{define } B := E^{-1}AE^{-T} \;\; z := E^T x, \;\; c := E^{-1}b. \tag{2.35}$$

$$\widehat{d}_0 = \widehat{r}_0 = c - Bz, \tag{2.36}$$

$$\alpha_k = \frac{\widehat{r}_k^T \widehat{r}_k}{\widehat{d}_k^T B \widehat{d}_k}, \tag{2.37}$$

$$z^{k+1} = z^k + \alpha_k \widehat{d}_k, \tag{2.38}$$

$$\widehat{r}_{k+1} = \widehat{r}_k - \alpha_k B \widehat{d}_k, \tag{2.39}$$

$$\beta_{k+1} = \frac{\widehat{r}_{k+1}^T \widehat{r}_{k+1}}{\widehat{r}_k^T \widehat{r}_k}, \tag{2.40}$$

$$\widehat{d}_{k+1} = \widehat{r}_{k+1} + \beta_{k+1} \widehat{d}_k. \tag{2.41}$$

The necessity of computing E in this procedure represents undesirable characteristic of the method. However, one can eliminate E by setting the following variable substitutions instead; $\widehat{r}_k := E^{-1}r_k$ and $\widehat{d}_k := E^{-T}d_k$, with using the identities $z_k := E^T x_k$ and $E^T E^{-1} = M^{-1}$. Then we derive the *untransformed preconditioned conjugate gradient method*

$$r_0 = b - Ax_0, \tag{2.42}$$

$$d_0 = M^{-1}r_0, \tag{2.43}$$

$$\alpha_k = \frac{r_k^T M^{-1} r_k}{d_k^T A d_k}, \tag{2.44}$$

$$x^{k+1} = x^k + \alpha_k d_k, \tag{2.45}$$

$$r_{k+1} = r_k - \alpha_k A d_k, \tag{2.46}$$

$$\beta_{k+1} = \frac{r_{k+1}^T M^{-1} r_{k+1}}{r_k^T M^{-1} r_k}, \tag{2.47}$$

$$d_{k+1} = M^{-1}r_{k+1} + \beta_{k+1}d_k. \tag{2.48}$$

It is clear that only the matrix M^{-1} is used in these equations rather than the matrix E. Moreover, we do not need to compute the matrix M^{-1} explicitly, as we only need its product with a vector. Similarly, one can derive a preconditioned steepest descent method that does not use E.

The effectiveness of a preconditioner M can be determined either by computing the condition number of $M^{-1}A$ or by the clustering of its eigenvalues. However, improving the convergence of the method depends mainly on finding a preconditioner that approximates A well enough. Accordingly, there is rich supply of possibilities. Examples of these, first, the preconditioner $M = A$ considers a perfect preconditioner in which the condition number of $M^{-1}A$ is one, whereas, we need to solve the system $Mx = b$ at the preconditioning step. So this preconditioner is not quite useful. The second simplest preconditioner is a diagonal matrix whose diagonal entries are identical to those of A, which is called diagonal preconditioning or Jacobi preconditioning. One more example is the incomplete Cholesky preconditioning, the matrix A is factorized into the form LL^T, where L is a lower triangular matrix. Many other preconditioners have been developed [5], but we will consider preconditioning conjugate gradient method, in Chapter 7, by one step of multigrid V-cycle.

2.3 Functional analysis

In this section, we present definitions and theory from the field of functional analysis that underpins the derivation of partial differential equations encountered in this thesis for real valued functions. For more details regarding this section, the interested reader can consult [91].

Definition 2.3. (Lipschitz continuous functions)

A function f on a domain Ω in $\mathbb{R}^n$ is called Lipschitz continuous if there is a constant $L > 0$ such that

$$|f(x) - f(y)| \leqslant L|x - y| \text{ for all } x, y \in \Omega.$$

Every Lipschitz continuous function is uniformly continuous, but the converse is not true.

Multi-index notation. In many of the following definitions it is required to introduce the multi-index notation for partial derivatives. Suppose that $\mathbb{Z}_+^n$ is the set of all ordered n-tuples of nonnegative integers. An element of $\mathbb{Z}_+^n$ is usually indicated by α or β, for example

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_n),$$

where each component α_i is a nonnegative integer. The sum is denoted by $|\alpha|$ and defined as

$$|\alpha| = \alpha_1 + \alpha_2 + ... + \alpha_n,$$

allowing for the partial derivative $D^\alpha u$ to be written as

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} ... \partial x_n^{\alpha_n}}.$$

Definition 2.4. (Space $C^k(\Omega)$)

The space $C^k(\Omega)$ of continuous functions f is defined as a set of functions $f : \Omega \to \mathbb{R}$, which posses at least k continuous derivatives, where $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$,

$$C^k(\Omega) := \{f : \quad D^\alpha f \text{ continuous }, 0 \leqslant |\alpha| \leqslant k\}.$$

For instance, C^0 and C^1 denote the spaces of continuous and continuously differentiable functions, respectively. Furthermore, we can denote the space of functions that are continuous with continuous derivatives of all orders by $C^\infty(\Omega)$. Using the definition provided above we have the following inclusions

$$C^\infty(\Omega) \subset ... \subset C^k(\Omega) \subset C^{k-1}(\Omega) \subset ... \subset C^0(\Omega) = C(\Omega)$$

Definition 2.5. (L^p Spaces)
The space $L^p(\Omega)$ defines the space of all measurable functions u defined on Ω such that the integral of the power p of the absolute value of a function $u(x)$ is finite

$$\int_\Omega |u(x)|^p dx.$$

Therefore we have

$$L^p(\Omega) := \left\{ u : \Omega \to \mathbb{R} \mid \int_\Omega |u(x)|^p dx < \infty \right\}, \qquad p \in [1, \infty). \tag{2.49}$$

In particular, when $p = 2$, we can obtain the space of *square integrable functions* $L^2(\Omega)$.
Definition 2.6. (The Space $L^\infty(\Omega)$)
The space $L^\infty(\Omega)$ is defined as the set of all measurable functions on Ω which are bounded almost everywhere in Ω:

$$L^\infty(\Omega) = \{u : |u(x)| \leqslant k \text{ a.e. on } \Omega \text{ for some } k \in \mathbb{R}^+\}. \tag{2.50}$$

Using [91, Theorem 2(b), p.74], for any function $u \in L^\infty(\Omega)$ defined on a bounded domain Ω, the following inequality holds

$$\int_{\Omega} |u(x)|^p dx \leqslant \int_{\Omega} k^p dx < \infty.$$

Consequently, $L^{\infty}(\Omega)$ represents a subset of $L^p(\Omega)$ for all $p \geqslant 1$.

Definition 2.7. (*Inner Product Spaces*)

A vector space X is called an *inner product space* if it is combined with the inner product operation $(\cdot\,,\cdot)$ for which the following axioms hold (for $u, v, w \in X$ and $\alpha, \beta \in \mathbb{R}$),

(a) $(u, v) \in \mathbb{R}$ (inclusion);

(b) $(v, u) = (u, v)$ (symmetry);

(c) $(\alpha u + \beta v, w) = \alpha(u, w) + \beta(v, w)$ (linearity);

(d) $(u, u) \geqslant 0$ and $(u, u) = 0$ if and only if $u = 0$ (positive definiteness).

Example 2.1. Assume that $X = \mathbb{R}^3$. Then, we define the Euclidean scalar product for $u, v \in X$ as follows

$$(u, v) = u_1 v_1 + u_2 v_2 + u_3 v_3.$$

Definition 2.8. (Normed Spaces)

A vector space X endowed with a norm operation $\|\cdot\|$ is called *norm space*, if for vectors $u, v \in X$ and $\alpha \in \mathbb{R}$ the norm satisfies the following axioms:

(a) $\|u\| \in \mathbb{R}$;

(b) $\|u\| \geqslant 0$ and $\|u\| = 0$ if and only if $u = 0$ (positive definiteness);

(c) $\|u\| \ = |\alpha| \|u\|$ (positive homogeneity);

(d) $\|u + v\| \leqslant \|u\| \ + \|v\|$ (triangle inequality).

Example 2.2. Firstly, consider $X = \mathbb{R}^n$, then the Euclidean norm for a vector $v \in \mathbb{R}^n$ is defined as

$$\|v\| = (v_1^2 + v_2^2 + ... + v_n^2)^{1/2}. \tag{2.51}$$

The operator $\| \cdot \|_p$ actually represents a whole family of norms for $1 \leqslant p < \infty$, defined for $v \in \mathbb{R}^n$ as

$$\|v\|_p = (|v_1|^p + |v_2|^p + ... + |v_n|^p)^{1/p}.$$

This family can include the norm corresponding to the case $p = \infty$, which is defined as

$$\|v\|_\infty = \max_{1 \leqslant i \leqslant n} |v_i|.$$

The usual norm for any $v \in L^p(\Omega)$ is defined as

$$\|v\|_{L^p} = \left[\int_\Omega |v(x)|^p d\Omega\right]^{1/p}.$$

Finally, the space $L^\infty(\Omega)$ (2.6) is a normed space, where the norm is defined by

$$\|u\|_{L^\infty} = \inf\{k : |u(x)| \leqslant k \text{ a.e. on } \Omega\}.$$

Definition 2.9. (Cauchy Sequence)

Let $n \in N$, a sequence $\{u_n\}$ is called a Cauchy sequence if

$$\lim_{m,n \to \infty} \|u_m - u_n\| = 0,$$

where $\{u_n\}$ belongs to a subset Y of a normed space X.

Definition 2.10. (Complete Space)

A subset Y of a normed space X is called a complete space if every Cauchy sequence of Y converges to an element of Y.

Definition 2.11. (Banach and Hilbert Spaces)

A Banach space is a complete normed space; a Hilbert space is a complete inner product space. Every Hilbert space represents a Banach space because every inner product induces a norm.

Definition 2.12. (Sobolev Spaces $W^{k,p}(\Omega)$)

For $0 \leqslant p \leqslant \infty$ and for $k = 0, 1, ...$, the Sobolev space $W^{k,p}(\Omega)$ is defined as

$$W^{k,p}(\Omega) := \{u \in L^p(\Omega) \mid D^\alpha u \in L^p(\Omega), |\alpha| \leqslant k\}, \qquad p \in [1, \infty]. \tag{2.52}$$

In other words, this represents the space of functions that, along with all their weak derivatives up to order k [91], belong to $L^p(\Omega)$. When equipped with the norm

$$\|u\|_{W^{k,p}} := \|u\|_{k,p} = \left(\sum_{|\alpha| \leqslant k} \int_\Omega |D^\alpha u|^p d\Omega \right)^{1/p}. \tag{2.53}$$

this space is a Banach space.

Definition 2.13. (Sobolev Spaces $H^k(\Omega)$)

For the special case of $p = 2$, we define

$$H^k(\Omega) := \{u \in L^2(\Omega) | D^\alpha u \in L^2(\Omega), |\alpha| \leqslant k\}, \qquad p \in [1, \infty]. \tag{2.54}$$

As such, $H^k(\Omega)$ is an inner product space when equipped with the inner product $(.,.)_{H^k}$ define as

$$(u, v)_{H^k} = \int_\Omega \sum_{|\alpha| \leqslant k} (D^\alpha u)(D^\alpha v) d\Omega \text{ for } u, v \in H^k(\Omega). \tag{2.55}$$

This in turn induces the Sobolev norm $\|.\|_{H^k}$ as follows

$$\|u\|^2_{H^k} = (u, u)_{H^k} = \int_\Omega \sum_{|\alpha| \leqslant k} (D^\alpha u)^2 dx.$$

From this definition, it is clear that $H^0(\Omega) = L^2(\Omega)$. Furthermore, we can write $(u, v)_{H^k}$ as the sum of the L^2-inner products of $D^\alpha u$ and $D^\alpha v$ over all α, $|\alpha| \leqslant k$, as follows

$$(u, v)_{H^k} = \sum_{|\alpha| \leqslant k} (D^\alpha u, D^\alpha v)_{L^2}.$$

The Sobolev norm can then be written as

$$\|u\|_{H^k}^2 = \sum_{|\alpha| \leqslant k} \|D^\alpha u\|_{L^2}^2.$$

The defined Sobolev space $H^k(\Omega)$ together with the inner product (2.55) represents a *Hilbert space* [91, p. 230].

We are interested in the problem of how to define the function u on the boundary $\varGamma$ when u belongs belongs to $L^2(\Omega)$ or, more generally, to one of the Sobolev space $H^k(\Omega)$. We can note that the functions in the space $H^k(\Omega)$ are defined on the domain Ω and not on $\overline{\Omega}$ because $\varGamma$ is a set of measure zero. Further, members of the space $H^k(\Omega)$ are in fact equivalence classes of functions, two functions being equivalent if they differ on a set of measure zero. Therefore, we need to introduce the following theorem.

Theorem 2.6. There exists a linear continuous operator $\gamma_0 : H^1(\Omega) \to L^2(\varGamma)$ such that

$$\gamma_0 v = v|_\varGamma \quad \text{for all } v \in C^1(\overline{\Omega})$$

The function $\gamma_0 v$ is called trace of v (often it is denoted by tr(v)). The meaning of the theorem is the following: if we have two close functions $u, v \in H^1(\Omega)$ then, due to continuity, also their traces are close. (Note that two functions from $H^1(\Omega)$ which are equal in and differ on $\varGamma$ have the same $H^1(\Omega)$ norm), see [86]. We can now introduce the definition of the space $H_0^1(\Omega)$ as follows.

Definition 2.14. (The space $H_0^1(\Omega)$)

The space $H_0^1(\Omega)$, is a subspace of $H^1(\Omega)$ (the Sobolev space $H^k(\Omega)$ with $k = 1$) to include homogeneous Dirichlet boundary conditions as follows

$$H_0^1(\Omega) = \left\{ u \in H^1(\Omega) | \;\; \gamma_0 u = 0 \;\; \text{on} \;\; \mathit{\Gamma} \right\}. \tag{2.56}$$

Definition 2.15. (*Bilinear Forms*)

Let X and Y denote two vector spaces. An operator $a(.,.) : X \times Y \to \mathbb{R}$ with the following properties is called a bilinear form:

$$\begin{aligned} &a(\alpha u + \beta w, v) = \alpha a(u, v) + \beta a(w, v), \; \text{for } u, w \in X, v \in Y, \; \text{(associativity/ commutativity)} \\ &a(u, \alpha v + \beta w) = \alpha a(u, v) + \beta a(u, w), \; \text{for } u \in X, v, w \in Y, \; \text{(reflexivity)} \end{aligned} \tag{2.57}$$

where α and $\beta \in \mathbb{R}$.

Furthermore, a is called a continuous bilinear form if X and Y represent normed linear spaces and a positive number c exists that satisfies the following inequality:

$$|a(u, v)| \leqslant c\|u\|_X \|v\|_Y \;\; \forall u \in X, v \in Y. \tag{2.58}$$

Definition 2.16. (H-Elliptic Bilinear Forms)

Let a bilinear form $a : H \times H \to \mathbb{R}$ be given with H be a Hilbert space. If for a constant $\alpha > 0$ the form a satisfies the following inequality

$$a(v, v) \geqslant \alpha \|v\|_H^2 \;\; \forall v \in H,$$

then a is called H-elliptic form.

Theorem 2.7. (Lax-Milgram theorem)

Assume that the operator $a(.,.) : H \times H \to \mathbb{R}$ is continuous and H-elliptic for a Hilbert space

H. Then, for any continuous linear functional l on H there exists a unique element $u \in H$ such that:

$$a(u, v) = (l, v)$$

for all $v \in H$.

The proof of the theorem is given in [pp. 167-169] [91].

CHAPTER 3

MULTIGRID METHODS

3.1 Introduction

The purpose of this chapter is to present basic concepts of multigrid methods. Such methods were originally developed for the solution of large systems of linear algebraic equations arising from the discretization of partial differential equations. The notable feature of such approaches is that the convergence speed does not deteriorate during the mesh refining process unlike a number of classical iterative methods such as Jacobi and Gauss-Seidel schemes which become slower with increasing dimension [20].

In 1961, Fedorenko [34] introduced the first two-grid method for the Poisson equation, while Fedorenko [35] contains the first multi-grid method. Bakhvalov [7] followed that by the first more general convergence analysis. The method, however, only gained huge popularity following the seminal paper by Brandt [16] (1973) who demonstrated the tremendous computational potential of these methods. Since then a vast literature about multi-grid methods has been published and introduced, we do not try to present this literature. Instead we refer to the monographs [57], [115] which are particularly devoted to problems of fluid dynamics. For the interested reader, many interesting classical texts in multigrid methods are introduced comprising Hackbusch and Trottenberg's Multigrid methods [60], Brandt's guide to multigrid methods [18], the introductory tutorial by Briggs et al., [20] and the comprehensive textbook by Trottenberg et al. [108]. The material presented in this chapter is in line with presentations provided in a number of classical references, including [16, 20, 57, 108]; for the interested reader, an extensive overview of multigrid methods may also be found in [114].

Generally, the multigrid procedure involves communication between a hierarchy of grids in finite difference or finite element discretization of an underlying partial differential equation PDE. For instance, the low-frequency components at the finer level h can be restricted to the coarser level $2h$, in order to reduce the error effectively by a coarse grid correction technique. Once this coarser problem is solved, the solution interpolates back to the fine grid to correct the approximation for its low-frequency errors. This approach is called geometric multigrid method and is considered within the scope of our work. Moreover, there is another approach of multigrid methods called algebraic multigrid which is a purely matrix-based approach. It can be directly applied to solve various types of elliptic partial differential equations without any geometric background and discretized on unstructured meshes in both two and three dimensions [105]. Alternatively it can be used whenever geometric multigrid can not be used or is difficult to apply, for instance if the discretization does not provide hierarchy of finite element meshes or the coarsest grid remains too large to be computed efficiently by a direct or classical iterative solver.

However, geometric multigrid methods are more general and can also be applied to nonlinear boundary value problems. The motivation for the chapter is to introduce fundamental algorithms of linear and nonlinear multigrid for system of equations and the mechanics behind multigrid including interpolation and restriction operators. Furthermore, the basic analysis of a one dimensional model problem will be presented to illustrate multigrid process in practice.

3.2 Ingredients of multigrid method

This section is devoted to presentation of the most prominent ingredients of multigrid methods which are commonly used in algorithms associated with multigrid methods. These will be demonstrated first by considering a linear system of equations

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

where $u, f \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ is a symmetric positive definite.

3.2.1 Hierarchy of Linear Systems

The solution of a linear system (3.1) requires a hierarchy $h_1 > ... > h_{\ell-1} > h_\ell$ of mesh sizes which correspond to the discretized domains $\Omega_1, ..., \Omega_{\ell-1}, \Omega_\ell$, respectively. We denote by h_ℓ the smallest mesh size of the finest level ℓ.

The linear system corresponding to the level k (mesh size h_k) can be written as

$$A^k u^k = f^k, \text{ for } k = \ell, \ell-1, ..., 1 \tag{3.2}$$

We first assume that there are two discretization levels to introduce two-grid method, fine h and coarse $2h$, resulting linear systems $A^h u^h = f^h$ and $A^{2h}u^{2h} = f^{2h}$ for the same discretized problem but on different levels.

3.2.1.1 Relaxation Iteration

In order to solve, (3.1), multigrid method should be combined with other iterative methods, called relaxation (smoothing) iteration, which have the smoothing property. Namely, these methods such as Jacobi or Gauss-Seidel iteration are slow but serve well to ''smoothen'' the error, i.e., to eliminate high-frequency components of the error, see pre-smoothing in Figure

3.1. However, these methods leave the low frequencies error which can be treated successfully by solving a ‘’correction’’ problem on the coarse mesh see post-smoothing in Figure 3.1, hence two-grid method.

Therefore, we denote by $u = RELAX(A, f; v)$, an iterative (relaxation) method to solve $Au = f$, where v is an initial approximation of the solution.

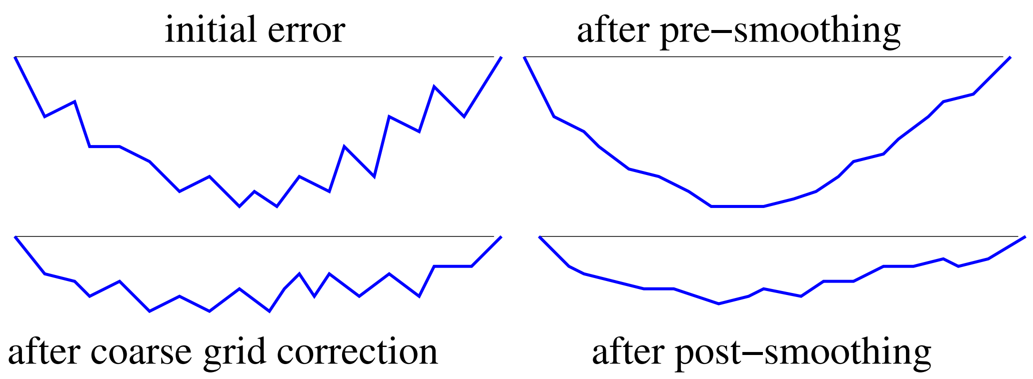


Figure 3.1: Pre and post-smoothing

3.2.1.2 Prolongation

By H_k, we denote the space of vectors u^k and f^k. The prolongation I_{k-1}^{k} is a linear transfer from the space of vectors H_{k-1} at the coarser level $k-1$ to the space at the finer H_k,

$$I_{k-1}^{k} : H_{k-1} \to H_k \text{ linear.} \tag{3.3}$$

3.2.1.3 Restriction

The restriction I_k^{k-1} is the opposite direction of a linear transfer of prolongation,

$$I_k^{k-1} : H_k \to H_{k-1} \text{ linear.} \tag{3.4}$$

3.3 Finite Element Introductory Model Problem

The analysis of multigrid methods was originally carried out on simple boundary value problems related to physical applications. Therefore, for simplicity we consider the following Poisson equation with homogeneous Dirichlet boundary conditions

$$\begin{aligned} Au \equiv -u_{xx} - u_{yy} &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \mathit{\Gamma}. \end{aligned} \tag{3.5}$$

Here $\Omega \subset \mathbb{R}^2$ denotes a bounded domain and $\mathit{\Gamma}$ its boundary. We assume that u belongs to the space $H := H_0^1(\Omega)$, namely the Sobolev space given in (2.14). We call A *formal differential operator*, as it only presents a certain formal writing; it is not clear if the derivatives in (3.5) exist.

Formally, solving the differential equation $Au = f$ and minimizing the functional

$$F(u) \equiv \frac{1}{2}(Au, u) - (f, u) \text{ over } H \subset L^2(\Omega) \tag{3.6}$$

$$\min_{v \in H} F(v), \tag{3.7}$$

are equivalent provided that the operator A is H-elliptic [20].
It is concluded that if $Au = f$, then u minimizes F over H, or $F(u) \leqslant F(u + v)$ for all $v \in H$. Conversely, if u minimizes F over H, then $(Au - f, v) = 0$ for all $v \in H$, which, using [20, p.178], is equivalent to

$$(Au, v) = (f, v) \text{ for all } v \in H \tag{3.8}$$

This condition describes another way of enforcing equality between Au and f, which we will now exploit in order to describe a discretization of the problem using finite elements. This approach is based on replacing the infinite dimensional space H by a finite dimensional space H_h, such that $H_h \subset H$ with H_h consisting of piecewise bilinear functions u_h. Each function u_h is continuous on Ω, zero on the boundary Γ and bilinear within each element. Therefore, on each element u^h has the form $u_h(x, y) = axy + bx + cy + d$ with a, b, c and $d \in \mathbb{R}$ constants.

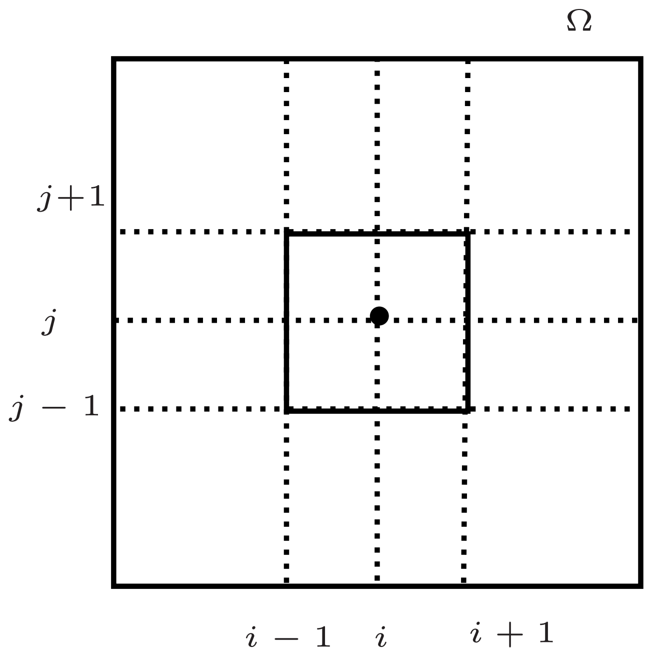


Figure 3.2: A domain Ω showing the four elements surrounding grid point (i, j) [20].

We suppose that the domain is a unit square, namely $\Omega = (0, 1) \times (0, 1)$, and is subject to a uniform discretization into $(n + 1) \times (n + 1)$ elements with mesh size $h = \frac{1}{n}$ to yield a discretization Ω^h. By denoting (x_i, y_j) the grid point with coordinates (ih, jh) (Figure 3.2); it can seen that under a mesh discretized into rectangles, the finite element method focuses on the four elements surrounding each grid point as opposed to the actual grid point itself. The sets of four elements associated to neighboring grid points interfere in one or two elements (Figure 3.2). The discrete minimization problem (3.7) at level h can be presented as follows

$$\min_{v^h \in H_h} F(v^h), \tag{3.9}$$

The equivalence described between (3.7) and (3.8) transfers over to the discrete formulation, meaning that (3.9) is equivalent to determining $u^h \in H_h$ such that

$$(Au^h, v^h) = (f, v^h) \text{ for all } v^h \in H_h \tag{3.10}$$

In practice, the solution of (3.10) needs further consideration. First, it is important to select suitable piecewise bilinear functions in H_h. For an interior point (x_i, y_j) denote $\phi_{i,j}(x, y)$, defined such that they take on the value 1 at (x_i, y_j) and zero at all other grid points. Then we can expand u^h in the form of the continuous piecewise linear function

$$u^h(x, y) = \sum_{i,j=1}^{n-1} u^h_{i,j}\phi_{i,j}(x, y), \text{ where } u^h_{i,j} = u^h(x_i, y_j). \tag{3.11}$$

This is referred to as a nodal basis expansion, as the nodal value $u^h_{i,j}$ gives the value of u^h at (x_i, y_j). The next step is to substitute (3.11) into (3.8), but direct implementation is not possible. The reason essentially lies in the fact that A is a second order operator, whereas u^h as defined is a linear function in both x and y and so is not sufficiently smooth. More precisely, the first partial derivatives of u^h are piecewise smooth and square integrable. However, their second derivatives are not square integrable due to their discontinuity at the element boundaries. To overcome these difficulties (3.8) is reformulated in order to include fewer derivatives. This is obtained by using the fact that u and v vanish on the boundary Γ. Applying the Gauss divergence theorem to (Au, v), we find that

$$\begin{aligned}(Au, v) &= \int_\Omega (-u_{xx} - u_{yy}) v \, d\Omega \\ &= \int_\Omega (u_x v_x + u_y v_y) \, d\Omega \\ &= (\nabla u, \nabla v).\end{aligned} \tag{3.12}$$

Substituting this into (3.8) leads to a weak form

$$(\nabla u^h, \nabla v^h) = (f, v^h) \text{ for all } v^h \in H^h, \tag{3.13}$$

which is more general or weaker than (3.8), as the formulation is free from being twice differentiable.

Now with this weak form we can substitute the expansion (3.11) into (3.13), choosing the so called trial functions v^h to be the basis functions $\phi_{k,l}$. The result of this is a linear system whose variables are the nodal values $u^h_{i,j}$. The inner product $(\nabla\phi^h_{i,j}, \nabla\phi^h_{k,l})$ represents the matrix coefficients in this system and $(f, \phi^h_{k,l})$ the right hand values. For the chosen bilinear basis functions, the overlapping only occurs at neighboring basis functions; thus the inner product $(\nabla\phi^h_{i,j}, \nabla\phi^h_{k,l})$ equals zero unless $k = i$ or $i \pm 1$ and $l = j$ or $j \pm 1$. Associating with the patch (i, j), the sixteen inner products result in a local stiffness matrix defined by the stencil

$$A^h_{i,j} = \frac{1}{6}\begin{pmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{pmatrix}. \tag{3.14}$$

Namely, this can be obtained by using standard quadrilateral bilinear finite elements discretization on a unit square in which the bases are defined as follows

$$\begin{aligned} \phi_{-1,-1} &= \frac{1}{4}(1-x)(1-y), \\ \phi_{1,-1} &= \frac{1}{4}(1+x)(1-y), \\ \phi_{1,1} &= \frac{1}{4}(1+x)(1+y), \\ \phi_{-1,1} &= \frac{1}{4}(1-x)(1+y). \end{aligned} \tag{3.15}$$

The right hand inner product includes the integrals

$$(f, \phi^h_{k,l}) = \int_{x_{i-1}}^{x_{i+1}} \int_{y_{i-1}}^{y_{i+1}} f\phi^h_{k,l}\,dxdy, \tag{3.16}$$

which are commonly approximated numerically. Replacing the function f by its value $f(x_i, y_j)$

is the simplest numerical integration scheme such that

$$\begin{aligned}(f, \phi_{k,l}^h) &\approx f(x_i, y_j) \int_{x_{i-1}}^{x_{i+1}} \int_{y_{i-1}}^{y_{i+1}} \phi_{k,l}^h dxdy, \\ &= \frac{h^2}{4} f(x_i, y_j). \end{aligned} \tag{3.17}$$

Therefore, assembling all local stiffness matrices (3.14) for all elements and associated right hand side can be written as the matrix vector system.

$$A^h u^h = f^h, \tag{3.18}$$

here, the solution vector u^h and the source vector are given by $(u^h)_{i,j} = (u_{i,j}^h) \in \Omega^h$ and $(f^h)_{i,j} = \left(\frac{h^2}{4} f(x_i, y_j)\right)$, respectively.

Under previous observations, we note that the solution u^h to (3.18) maybe equivalently obtained by solving the quadratic optimization problem

$$\min_{v^h \in \Omega^h} F^h(v^h), \tag{3.19}$$

where

$$F^h(v^h) \equiv \frac{1}{2}(A^h v^h, v^h) - (f^h, v^h).$$

3.3.1 Correction Scheme

The multigrid scheme can be developed by considering the functions u^h that solve (3.9) and (3.10) instead of their nodal values u^h defined in (3.18) (simplicity, we use the same notation for the function and the associated vector). The first step is to consider relaxation, for example Jacobi method, which can provide an inexpensive techniques for damping oscillatory errors in the approximation, v^h. This can be done by applying local changes of the form

$$v^h \leftarrow v^h - \alpha\phi^h_{i,j}, \tag{3.20}$$

for each $i, j = 1, 2, ..., n - 1$ where $\alpha \in \mathbb{R}$ is a suitably selected step size. The best way of choosing α in terms of the minimization principle is to minimize the functional over all possible choices. This leads to the following relaxation algorithm.

Algorithm 3.1. for each $i, j = 1, 2, ..., n-1$, compute $\alpha = \arg\min_{s\in\mathbb{R}} F(v^h - s\phi^h_{i.j})$, and then apply the replacement $v^h \leftarrow v^h - \alpha\phi^h_{i,j}$.

Now we can follow this by formulating the coarse grid correction. The coarse grid space $H^{2h} \subset H^h$ defines the set of piecewise bilinear functions connected with the standard coarse grid Ω^{2h} corresponding to the even numbered lines of Ω^h. The target is to correct the approximation v^h by a function $v^{2h} \in H^{2h}$, in order to approximate the presumably smooth error, the form of this correction is $v^h \leftarrow v^h + v^{2h}$; the best way of choosing v^{2h} in terms of the minimization principle is to minimize F over H^{2h}. The statement mathematically is

$$v^{2h} = \arg \min_{w^{2h}\in H^{2h}} F(v^h + w^{2h}). \tag{3.21}$$

The coarse grid correction scheme is simply stated as follows

- Compute $v^{2h} = \arg\min_{w^{2h}\in H^{2h}} F(v^h + w^{2h})$ and then set $v^h \leftarrow v^h + v^{2h}$.

The core of multigrid method constitutes of recursive combination of this correction scheme and the coordinate relaxation scheme defined above. The correction scheme can then be represented as follows in more general algorithm by using the residual equation.

Algorithm 3.2. - Use a few steps of a relaxation iterative method to solve $Au = f$ on Ω^h to generate an approximation v^h.

- Compute the residuum $r = f - Av^h$.

- Solve the residual equation $Ae = r$ (by a relaxation method) on Ω^{2h} to generate an approximation to the error e^{2h}.

- Correct the approximation obtained on the fine grid Ω^h by adding the error estimate generated on the coarse grid Ω^{2h}; $v^h \leftarrow v^h + e^{2h}$.

We now discuss the transfer mechanism which relies on previously presented spaces and bases as follows.

3.3.2 Interpolation and restriction operators

As described previously, we require the addition of a function from the coarse grid Ω^{2h} to a function from the fine grid Ω^h which then provides a complete to the relaxation scheme. We consider only the case where the coarse grid has double of the grid spacing of the next finest grid, this is the most common used in practice. As such, it is necessary to describe an appropriate mapping from the nodal representation of a function in Ω^{2h}, namely

$$v^{2h}(x, y) = \sum_{i,j=1}^{\frac{n}{2}-1} v_{i,j}^{2h} \phi_{i,j}^{2h}(x, y),$$

to Ω^h nodal representation,

$$v^{2h}(x, y) = \sum_{i,j=1}^{n} v_{i,j}^{h} \phi_{i,j}^{h}(x, y).$$

In other words, we look to obtain coefficients $v_{i,j}^h$ that enable v^{2h} to be written as a function in H^h, using the fact that $H^{2h} \subset H^h$. We denote such a mapping by $I_{2h}^h : \Omega^{2h} \rightarrow \Omega^h$ such that

$$v^h = I_{2h}^h v^{2h}. \tag{3.22}$$

This technique is referred to as a prolongation operator I_h^{2h}. The components of v^h can be obtained by defining a linear interpolation from the coarse grid Ω^{2h} to the next finer grid Ω^h, Fig. 3.3 which is illustrated as follows.

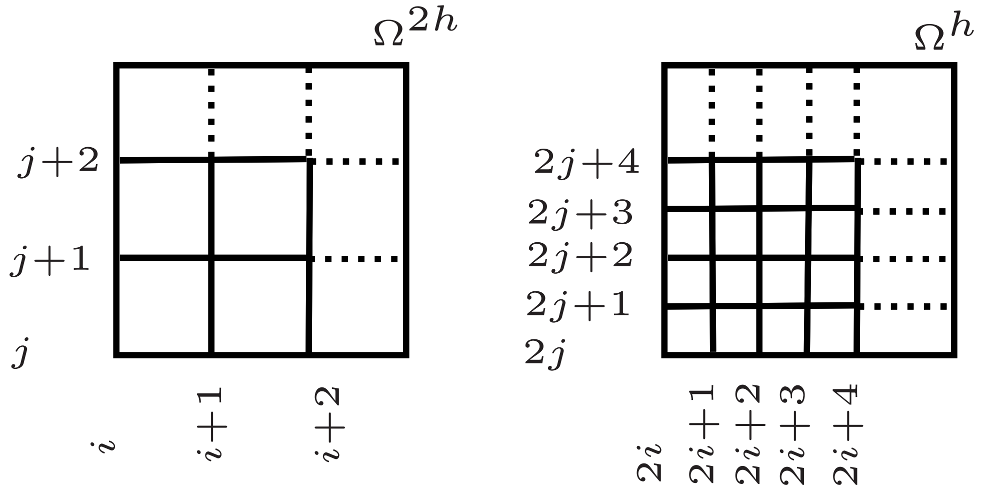


Figure 3.3: A domain Ω^{2h} with a coarse grid spacing $2h$ (left) and the associated fine grid Ω^h with grid space h (right).

Consider the grid Ω^{2h} node with indices (i, j) which is located at the point $(i(2h), j(2h))$. It is clear that this point is also the grid Ω^h node $((2i)h, (2j)h)$ with indices $(2i, 2j)$, Fig. 3.3. The coefficients in each case define the nodal function values at the nodes such that

$$v^h_{2i,2j} = v^{2h}((2i)h, (2j)h) = v^{2h}(i(2h), j(2h)) = v^{2h}_{i,j},$$

so that

$$v^h_{2i,2j} = v^{2h}_{i,j}.$$

Nodes on the fine grid with coordinates which have subsequent indices should be defined in an appropriate manner ensuring that v^{2h} is linear between subsequent points on the coarse grid. To enforce this at coordinates $((2i + 1)h, (2j)h)$ and $((2i)h, (2j + 1)h)$, we require

$$v^h_{2i+1,2j} = \frac{1}{2}(v^{2h}_{i,j} + v^{2h}_{i+1,j}),$$

and

$$v^h_{2i,2j+1} = \frac{1}{2}(v^{2h}_{i,j} + v^{2h}_{i,j+1}),$$

respectively. Both of theses relations imply that at a point on the fine grid with purely odd indices $((2i+1)h,(2j+1)h)$, we must have that

$$v^h_{2i+1,2j+1} = \frac{1}{4}(v^{2h}_{i,j} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}).$$

Thus the operator I^h_{2h} can be based on the nine-point interpolation scheme defined by the stencil

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}.$$

We now look to describe a counterpart to the discrete operator A^h on the coarse grid, which we will denote by A^{2h}. By using the minimization principle presented in (3.19), the coarse grid correction (3.21) is equivalent to

$$v^{2h} = \arg \min_{w^{2h} \in \Omega^{2h}} F^h(v^h + I^h_{2h} w^{2h}), \tag{3.23}$$

Using the properties of inner products, we then use this formulation in order to write down the following,

$$\begin{aligned} & F^h(v^h + I^h_{2h}w^{2h}) \\ =& \frac{1}{2}(A^h(v^h + I^h_{2h}w^{2h}), v^h + I^h_{2h}w^{2h}) - (f^h, v^h + I^h_{2h}w^{2h}) \\ =& F^h(v^h) + \frac{1}{2}((I^h_{2h})^T A^h I^h_{2h} w^{2h}, w^{2h}) - ((I^h_{2h})^T (f^h - A^h v^h), w^{2h}) \end{aligned}$$

We can notice from the first bracket that the matrix which operates on a coarse grid vector w^{2h} is represented by $(I^h_{2h})^T A^h I^h_{2h}$. Therefore, the coarse grid counterpart to A^h can be given by $(I^h_{2h})^T A^h I^h_{2h}$. The second bracket shows that a fine grid vector is transfered into a coarse grid vector by the operator $(I^h_{2h})^T$ which plays the role of a restriction operator. Therefore, it makes sense to represent them as

$$A^{2h} := (I_{2h}^{h})^T A^h I_{2h}^{h},$$

and

$$I_h^{2h} := (I_{2h}^{h})^T.$$

This restriction operation operation is called full weighting; see, e.g., [60]. Which can be written in more genral way as

$$I_h^{2h} := c(I_{2h}^{h})^T, \;\; c \in \mathbb{R}.$$

However, the most obvious restriction operator is injection that is defined by $I_h^{2h} v^h = v^{2h}$, where

$$v_j^{2h} = v_{2j}^{h}.$$

Namely, the coarse grid vector picks its values directly from the corresponding fine grid point. Now the relevant operators have been defined, we are in position to provide a precise definition of the multigrid algorithm. To illustrate this effectively, we first present the following two grid correction scheme.

So far we developed multigid scheme using the minimization principle relied on finite element discretization because it is our applied discretization method in this thesis. However, we have the option of developing multigrid solvers directly from the differential equation without adhere to optimality, as it is presented in many literatures [20, 57], but it is based on the finite differences method. To simplify things, we present in the next section the ingredient of multigrid method in a way that makes the practical concepts to be more obvious.

3.4 Multigrid method algorithms

3.4.1 Two-grid method

The two-grid iteration includes only two levels, a fine grid h and a coarse grid $2h$. Whilst not of practical interest for reasons to be described shortly, the resulting algorithm represents an important building block in the description of a generalized multigrid iteration. The two grid iteration may be described as follows.

Algorithm 3.3. Two-grid algorithm for solving $Au = f$

(a) Pre-smoothing step: Apply ν steps of a relaxation method, $u^h = RELAX_\nu(A^h, f^h; v^h)$ on H_h, with initial estimate v^h.

(b) Calculate the fine grid residuum $r^h = f^h - A^h v^h$.

(c) Restrict the fine grid residuum r^h to the coarse grid by $r^{2h} = I_h^{2h} r^h$.

(d) Compute $e^{2h} = RELAX_\nu(A^{2h}, r^{2h}; 0)$ on the coarse level H_{2h} to machine precision.

(e) Interpolate the coarse grid error to the fine grid by $e^h = I_{2h}^h e^{2h}$.

(f) Correct the fine grid approximation by $v^h \leftarrow v^h + e^h$.

(g) Post-smoothing step: Apply ν steps of a relaxation method, $u^h = RELAX_\nu(A^h, f^h; v^h)$ on H_h, with updated initial estimate v^h.

Here ν denotes the number of smoothing iterations. For the converge details regarding the two-grid iteration in the one-dimensional case, we refer the interest reader to [58]. As an iteration process, the two-grid method will be relatively fast in practice [58]. Nevertheless, it is impractical in the more interesting case of multi-dimensional boundary value problems due to the need to determine the exact solution to $A^{2h}e^{2h} = r^{2h}$ in (d) of Algorithm 3.3 in order to compute an update on the fine grid in the last step. However, since the error e^h on the fine grid is only an approximation, there is no explicit need to determine e^{2h} exactly. Instead, one can consider an iterative approximation to

$$A^{2h}e^{2h} = r^{2h},$$

it is clear that this formulation is the same as the original problem $A^h u^h = f^h$ with only the difference in the grids, $2h$ instead of h. We can therefore apply the coarse grid algorithm to the residual equation with grids$\{2h, 4h\}$ instead of $\{h, 2h\}$, so that the relaxation is considered on Ω^{2h} with associated coarse grid corresponding to Ω^{4h}. Assuming that a suitably fine mesh is considered in the first instance, this process can be repeated until it is computationally feasible to obtain a direct solution to the associated residual equation. One more point needs to be mentioned. Due to the fact that there is no information available regarding the solution on the coarse grid, u^{2h}, the initial guess to e^{2h} on Ω^{2h} is simply chosen to be zero, $e^{2h} = 0$, for simplicity the approximation to e^{2h} will be denoted by v^{2h}. We can now generalize the two-grid algorithm to multigrid algorithm for levels $k = \ell, \ell - 1, ..., 1$, as follows.

Algorithm MG (V-cycle correction scheme multigrid)

```
Set $\varepsilon, \varepsilon_0$. Initialize $v^{(\ell)}$.
for $i = 1 : niter$
    $u^{(\ell)} := mgm(\ell, v^{(\ell)}, f^\ell)$
    test convergence
end
function $u^k = mgm(k, v^k, f^k)$
    if $k = 0$
        $u^k := (A^1)^{-1} f^1$                                (coarsest grid solution)
```

```
else
    $v^k := RELAX_{\nu_1}(A^k, f^k; v^k)$                (pre-smoothing)
    $r_{k-1} = I_k^{k-1}(f^k - A^k v^k)$                 (restricted residuum)
    $v^{(k-1)} = mgm_\iota(k-1, 0_{n_{k-1}}, r_{k-1})$    (coarse grid correction)
    $v^{(k)} := v^k + I_{k-1}^k v^{(k-1)}$               (solution update)
    $u^k := RELAX_{\nu_2}(A^k, f^k; v^k)$                (post-smoothing)
end
```

where ν_1 and ν_2 denote the number of relaxation iterations at the pre- and post-smoothing steps, respectively. The resulting algorithm with $\iota = 1$ represents one call of multigrid algorithm and takes the letter V pattern (Fig. 3.4) which descends down to the coarsest grid then sets its way back to the finest grid. So it is called the $V-$cycle. One can apply two call ($\iota = 2$) of multigrid scheme called the w-cycle [58].

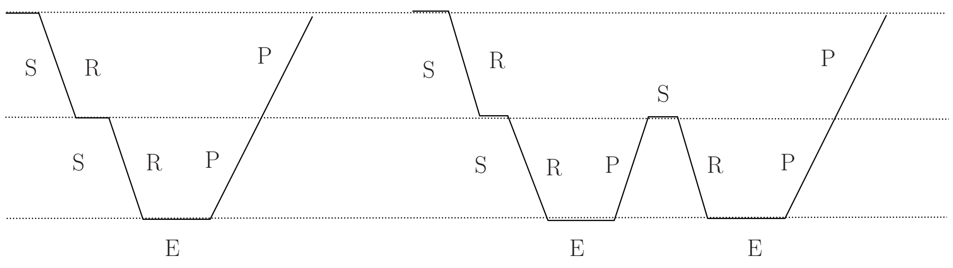


Figure 3.4: One $V - cycle$, and one $W - cycle$ for $l = 2$. R: restriction of the residual; P: correction $u^k \mapsto u^k - I_{k-1}^k u^{k-1}$, S: smoothing step; E: exact solution at level the coarsest level

3.5 Nonlinear multigrid methods

When using multigrid, there are two different approaches used in order to deal with nonlinear problems, the first approach is referred to as a full approximation scheme (FAS) by directly adapting the construction of the smoothing and coarse grid iterations to the nonlinear case.

The second of these approaches is a combination of Newton's method and multigrid which is called Newton-multigrid method. The second method does not treat the nonlinearity directly by multigrid ideas but it consists of Newton's method as the outer iteration and multigrid as an inner iteration applied to arising linear system from use of Newton method, see [20] for more details.

3.5.1 Full approximation scheme

To fully describe this approach, we consider

$$A(u) = f, \tag{3.24}$$

to be a system of nonlinear algebraic equations, whereby $u, f \in \mathbb{R}^n$. Let v be an approximate solution of u, so that the error $e = u - v$ with $r = f - A(v)$ denoting the residuum. We obtain the system

$$A(u) - A(v) = r, \tag{3.25}$$

after subtracting the equation $A(u) = f$ from the residual. Generally, in the nonlinear case one cannot deduce that $A(u) - A(v) = A(e)$, as illustrated in [20]. Thus, the system (3.25) must be solved rather than the error system. The full approximation scheme (FAS) is named so as it solves the coarse grid problem for the full approximation $u^h = I_{2h}^h v^{2h} + e^h$ rather than error e^h. The procedure can be viewed as a generalization of the two grid correction scheme presented in Algorithm 3.3 to nonlinear problems since the procedure reduces to the two grid correction scheme in the case where $A(.)$ is a linear operator. Thus, FAS similar to its linear counterpart, can also be implemented as a V-cycle or W-cycle scheme.

It is also useful to observe that the coarse grid equation of FAS can be written as

$$A^{2h}(u^{2h}) = f^{2h} + \delta_h^{2h},$$

where the delta correction is given by

$$\delta_h^{2h} := A^{2h}(I_h^{2h} v^h) - I_h^{2h} A^h(v^h).$$

The consequence of this relationship is that the resulting solution of the coarse grid FAS equation is not the same as the solution of the original coarse grid equation because in general $\delta_h^{2h} \neq 0$. We now describe the complete version of this approach for nonlinear system of equations $A(u) = f$. W assume that $nRELAX$ is a descent convergent algorithm for the solution of (3.24), ε denotes the required precision and ν is the maximum number of iterations allowed.

Algorithm FAS (Full Approximation Scheme Algorithm for Solving $A^k(u^k) = f^k$)

```
Set $\varepsilon$. Initialize $v^{(\ell)}$.
for $i = 1 : niter$
    $u^{(\ell)} := mgm(\ell, v^{(\ell)}, f^\ell)$
    test convergence
end
function $u^k = mgm(k, v^k, f^k)$
    if $k = 0$
        $u^k := nRELAX_{\nu_1}(A^k, f^k; v^k)$          (coarsest grid solution to high precision $\varepsilon$)
    else
        $v^k := nRELAX_{\nu_1}(A^k, f^k; v^k)$          (pre-smoothing)
        $v^{k-1} = I_k^{k-1} v^k$          (solution restriction)
        $f^{k-1} = I_k^{k-1} f^k + (A^{k-1}(I_k^{k-1} v^k) - I_k^{k-1} A^k(v^k))$          (correction r.h.s.)
        $cor^{(k-1)} = mgm_\iota(k-1, v^{k-1}, f^{k-1})$          (coarse grid correction)

        $z^{(k)} := v^k + I_{k-1}^k(cor^{k-1} - v^{(k-1)})$          (solution update)

        $u^k := RELAX_{\nu_2}(A^k, f^k; z^k)$          (post-smoothing)

    end
```

where ν_1 and ν_2 denote the number of relaxation iterations at the pre- and post-smoothing steps, respectively. The resulting algorithm with $\iota = 1$ represents one call of multigrid algorithm and leads to $V-$cycle, as it described in the previous section.

3.5.2 Full multigrid method

The concept of multigrid method can also be applied to a problem by considering a nested iteration. This strategy is based on using coarser grids in order to obtain improved initial guesses, namely that relaxing of the system $Au = f$ on the coarser level is considered in order to generate an initial guess for the next (finer) grid. This relaxation is repeated across the number of levels considered until the finest level is reached. It is depended on the definition of the original problem on the coarser grids in order to allow the improved results to be transferred to the finer grid.

We will now describe the so-called full multigrid method (FMG). In general, the solution process starts from the coarsest level, where the discretized problem can be solved in a straightforward manner. The obtained result is then interpolated to the next finer level and then used as an initial guess within FAS or MG procedure. The process repeats until the finest level is reached, with k used to denote the level and l the finest level.

Algorithm 3.4. Full Multigrid Algorithm
Full multigrid method for solving $A^l u^l = f^l$.

- Compute u^k on the working level k;

- If $k < l$ then FMG, interpolate the solution to the next finer working level

$$u^{k+1} = I_k^{k+1} u^k;$$

- Apply FAS (or MG) scheme to $A^{k+1}u^{k+1} = f^{k+1}$ with initial guess u^{k+1};

- If $k + 1 < l$ set $k = k + 1$ go to (3.4).

The idea principally can be combined with any iterative process to provide a good starting guess as we mentioned earlier. The method is not used in our practical methods because the solution is approachable when the initial point is set in all experiments to a zero vector (even if this was infeasible), however, an interested reader is recommended to consult [58].

CHAPTER 4

OPTIMIZATION

This section introduces the fundamental concepts of optimization required in this thesis. The main reference for this section is [85], moreover both [66] and [15] can be used for information regarding nonlinear optimization in general, as well as iterative solution methods used to solve resulting problems.

4.1 Basic definitions and theorems

Definition 4.1. (Convexity)
A set $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, is called *convex* if for every $x, y \in \Omega$ it holds that:

$$\alpha x + (1 - \alpha) y \in \Omega \qquad \forall \alpha \in [0, 1].$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ which for all $x, y \in \mathbb{R}^n$ satisfies the inequality

$$f(x + (1 - \alpha) y) \leqslant \alpha f(x) + (1 - \alpha) f(y) \qquad \forall \alpha \in [0, 1],$$

is called a convex function. Furthermore, f is called concave if $-f$ is convex.

Definition 4.2. (Optimization Problems)
Let $f, c_i, c_j : \mathbb{R}^n \to \mathbb{R}$ be continuous functions, $i = 1, \ldots, n_I$, $j = 1, \ldots, n_E$. The *constrained*

optimization problem (CP) is defined as

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{4.1}$$

$$\text{subject to} \quad c_i(x) \geqslant 0, \qquad i \in \mathcal{I} \tag{4.2}$$

$$c_i(x) = 0, \qquad i \in \mathcal{E}. \tag{4.3}$$

Here $x = (x_1, ..., x_n)$ defines a vector of unknowns, f in (4.1) is the objective function and (4.2) and (4.3) are the constraints with index sets, specified by $\mathcal{I}$ and $\mathcal{E}$. The defined problem turns into an *unconstrained optimization problem* (UP) when the union of the sets $\mathcal{E}$ and $\mathcal{I}$ is empty, in which case, the problem reduces to

$$\min_{x\in\mathbb{R}^n} f(x). \tag{4.4}$$

Depending on the character of the functions f, c_i, c_j, the optimization problem can be classified as either linear or nonlinear, convex or nonconvex, and smooth or nonsmooth.

Definition 4.3. (Local and Global Solution)
A point $x^* \in \Omega$ is called a local minimizer of an unconstrained problem (UP) if there exists a neighborhood of x^*, which is denoted by $U(x^*)$, such that $f(x) \geqslant f(x^*)$ for all $x \in \Omega \cap U(x^*)$. A point x^* is called a global minimizer if $f(x) \geqslant f(x^*)$ for all $x \in \Omega$.

We first state necessary and sufficient conditions for a point to be a local minimizer of (UP), where we see the importance of positive definiteness. For the proof of the next three theorems, see [85, pp. 15–16].

Theorem 4.1. (First-Order Necessary Condition for UP)
If x^* is a local minimizer and f is continuously differentiable in an open neighborhood of x^*, then $\nabla f(x^*) = 0$.

Theorem 4.2. (Second-Order Necessary Conditions for UP)
Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function in an open neighborhood of x^*. If

x^* is a local minimizer of f, then $\bigtriangledown f(x^*) = 0$ and $\bigtriangledown^2 f(x^*)$ is a positive semidefinite matrix.

Theorem 4.3. (Second-Order Sufficient Conditions for UP)

If $\bigtriangledown f(x^*) = 0$ and $\bigtriangledown^2 f(x^*)$ is continuous in an open neighborhood of x^* and is positive definite, then there exists an $\alpha > 0$, such that $f(x) \geqslant f(x^*) + \alpha\|x - x^*\|$ for all x near x^*.

We now turn our attention to constrained optimization problems.

Definition 4.4. (Lagrangian Function)

The function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{I}|} \rightarrow \mathbb{R}$ defined as

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{i \in \mathcal{I}} \lambda_i c_i(x) - \sum_{i \in \mathcal{E}} \mu_i c_i(x) \tag{4.5}$$

is called the Lagrangian function of CP (4.1-4.3). Here both of λ_i and μ_i are called Lagrange multipliers. For clarity, we call a vector of Lagrange multipliers by only λ.

Definition 4.5. (Active Set)

A set of active constraints can be defined as

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}. \tag{4.6}$$

Definition 4.6. (Linear Independence Constraint Qualification-$LICQ$)

We say that the *linear lndependence constraint qualification* ($LICQ$) holds at $x \in \Omega$ if the gradients of the active constraints $\{\nabla c_i(x) \mid, i \in \mathcal{A}(x)\}$ are linearly independent.

We are now in a position to introduce theorems regarding optimality conditions for constrained problems as described in (4.1)-(4.3).

Theorem 4.4. (First Order Necessary Conditions for CP)

Suppose that the functions f, c_i defined in (4.1)-(4.3) are continuously differentiable and ($LICQ$) holds at x^*, where x^* is a local minimizer of the constrained problem (4.1)-(4.3). Then there exists a Lagrange multiplier vector λ^*, with components $\lambda_i^*,\ i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*):

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0 \tag{4.7}$$

$$c_i(x^*) = 0, \quad \forall i \in \mathcal{E} \tag{4.8}$$

$$c_i(x^*) \geqslant 0, \quad \forall i \in \mathcal{I} \tag{4.9}$$

$$\lambda_i^* \geqslant 0, \quad \forall i \in \mathcal{I} \tag{4.10}$$

$$\lambda_i^* c_i(x^*) = 0, \quad \forall i \in \mathcal{I}. \tag{4.11}$$

These are the well-known Karush-Kuhn-Tucker (KKT) Conditions and the theorem is known as a KKT Theorem. The equations (4.11) are known as the complementary conditions. In particular, the Lagrange multipliers λ_i^* corresponding to inactive inequality constraints are zero. The proof of the KKT theorem can be found, for instance in [85, p.323].

The following definition is required to introduce the second order conditions.

Definition 4.7. Suppose that x^* is a solution of the problem (4.1-4.3) and the KKT conditions are satisfied. Then, an inequality constraint c_i is called strongly active or binding if $i \in A(x^*)$ and $\lambda_i^* > 0$ for some Lagrange multipliers λ^* satisfying KKT conditions. However, an inequality constraint c_i is called weakly active if $i \in A(x^*)$ and $\lambda_i^* = 0$ for all λ_i^* satisfying KKT conditions.

Definition 4.8. (Linearized Feasible Set of Directions)

The set of *linearized feasible directions* at a feasible point x is defined as

$$\mathcal{F}(x) = \left\{ d \mid d^T \nabla c_i(x) = 0, \;\; \forall i \in \mathcal{E}; \;\; d^T \nabla c_i(x) \geqslant 0, \;\; \forall i \in \mathcal{A}(x) \cap \mathcal{I}, \right\},$$

where $\mathcal{A}(x)$ represents the active set (4.6).

The KKT conditions describe the relation between the derivative of the objective function and the active constraints at a feasible point x^*. When these conditions hold, the approximation of the objective function will be dependent on the movement along $d \in \mathcal{F}(x^*)$, either by increasing it ($d^T\nabla f(x^*) > 0$), or by keeping it unchanged ($d^T\nabla f(x^*) = 0$). In other words, an increase or a decrease of the objective function cannot be indicated by the first order conditions alone. We therefore need to take into account higher order derivatives in order to be able to trade the curvature of the Lagrangian function in the direction $d \in \mathcal{F}(x^*)$. In particular, we look to present optimality conditions to encompass second order derivatives; for this, we require the notion of the critical cone.

Definition 4.9. (Critical Cone)

Consider the pair (x^*, λ^*) satisfying the KKT conditions. Then the critical cone is given as

$$C(x^*, \lambda^*) = \left\{d \in \mathcal{F}(x^*) \mid \bigtriangledown c_i(x^*)^T d = 0, \ \ \forall i \in \mathcal{A}(x^*) \cap \mathcal{I}, \ \lambda_i^* > 0\right\},$$

where $\mathcal{A}(x^*)$ is the active set and $\mathcal{F}(x^*)$ the linearized feasible directions set at a feasible point x^*.

For the proof of the following two theorems, see [85, pp. 332–333].

Theorem 4.5. (Second Order Necessary Conditions for CP)

Assume that x^* is a local minimizer of (CP) (4.1-4.3) and λ^* is the Lagrange multipliers vector such that they satisfy the KKT conditions and the $LICQ$ conditions hold at x^*. Then

$$d^T \bigtriangledown_{xx}^2 \mathcal{L}(x^*, \lambda^*) d \geqslant 0, \qquad \forall d \in C(x^*, \lambda^*).$$

Theorem 4.6. (Second Order Sufficient Conditions for CP)

Assume that the pair (x^*, λ^*) satisfies the KKT conditions. If

$$d^T \bigtriangledown_{xx}^2 \mathcal{L}(x^*, \lambda^*) d > 0, \qquad \forall d \in C(x^*, \lambda^*), \ d \neq 0,$$

then x^* is a strict local minimizer for the (CP).

4.2 Line search methods

Let first consider the unconstrained optimization problem

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

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$, we assume $f \in C^1$ with the Lipschitz continuous derivatives. In practice it is rather unusual to be able to compute or provide an explicit minimizer, despite knowing how to characterize local minimizers of the optimization problem. The consequence expect is to fall back on a suitable iterative process. An iteration is simply a procedure that generates sequence of points

$$\{x^k\}, \; k = 1, 2, ...$$

starts from some initial guess x^0, with the overall aim of ensuring that any limit generated satisfies first-order or, even better, second-order necessary optimality conditions.

In general, it is not guaranteed that an iteration converges to a global minimizer nor even to a local minimizer unless, respectively, f obeys very strong conditions. However, the general target is to ensure that, the iteration is globally convergent. From any initial point x^0, a subsequence of iterates $\{\nabla f(x^k)\}$ converges to zero. And with the hope that the convergence is reasonably fast. The heart of computational optimization is located on these two preoccupations. The concept of line search is based on a principal that it finds a direction of movement from an initial start point, according to a fixed rule; and then proceeds in that direction towards a minimum of the objective function on that line. At the new point it determines a new direction and repeats the process [75]. We come to introduce the line search method in a more formal way, which consists of computing a search direction d_k in each iteration that is required to be a descent direction, i.e.,

$$d_k \nabla f(x_k) < 0 \;\text{ if }\; \nabla f(x_k) \neq 0,$$

and calculating a step length along this direction $\alpha_k > 0$ such that

$$f(x_k + \alpha_k d_k) < f(x_k).$$

The iteration concludes, with given search direction and step length, by setting

$$x_{k+1} = x_k + \alpha_k d_k. \tag{4.12}$$

An effective choices of both the direction d_k and the step length α_k assure the success of a line search method. Moreover, the search direction often takes the form

$$d_k = -B_k^{-1} \nabla f_k, \tag{4.13}$$

where B_k is a symmetric non singular matrix. Variant choices of B_k lead to different line search methods. Namely, the steepest descent method, where B_k is the identity matrix, and Newton‘s method, where B_k is the exact Hessian $\nabla^2 f(x_k)$. Further, if B_k represents an approximation of the Hessian, then the line search method belongs to the class of so-called quasi-Newton methods. The primary differences between them depend on the selected directions d_k. In this chapter we focus mainly on study the steepest descent method as it is the main algorithm in Chapter 5. We preface with discussing how to choose α_k and d_k to raise convergence from distant starting point.

4.2.1 Step Length

Computing the step length α_k is not an easy task. Selecting the step size α_k should be based on producing a substantial reduction of f in a computationally efficient manner, as we discussed in the previous section. The ideal choice of α_k corresponds to the global minimizer of the following univariate function

$$\phi(\alpha) = f(x^k + \alpha_k d_k), \quad \alpha_k > 0. \tag{4.14}$$

However, identifying this value is generally too expensive. Typically too many evaluations of the objective function f and the associated gradient ∇f are required in order to determine a local minimizer of ϕ to moderate precision, and this approach known as an *exact linesearch* which is rarely used nowadays. Alternatively, many practical approaches prefer an inexact line search to determine a step length that attains sufficient reduction in f at minimal cost, which are assured to pick steps that are neither too short nor too long. The so-called Armijo-Wolf and backtracking-Armijo types are represented as the main contenders amongst the many possible inexact line searches, [47].

A first and a simple condition that can be imposed on α_k is that

$$f(x_k + \alpha_k d_k) < f(x_k).$$

This condition is not sufficient to achieve convergence to local solution x^* as it illustrated in [118, p.32]. This can be avoided by enforcing a sufficient decrease condition, such as the Wolf condition,

$$f(x_k + \alpha_k d_k) \leqslant f(x_k) + c_1 \alpha_k \nabla f_k^T d_k, \tag{4.15}$$

for some $c_1 \in (0, 1)$, assuring that α_k should give sufficient decrease in the objective function f. Further, the decrease in f will be proportional to both the directional derivative $\nabla f_k^T d_k$ and the step length α_k. The condition (4.15) is also referred to as Armijo condition.

The Figure 4.1 illustrates the sufficient decrease condition. Where $l(\alpha)$ denotes the right-hand side of the condition (4.15), which is a linear function and has a negative slope $c_1 \nabla f_k^T d_k$. But, as $c_1 \in (0,1)$, it prolongates above the graph of ϕ for small positive values of α. The sufficient decrease condition stipulates that if $\phi(\alpha) \leqslant l(\alpha)$ then α is acceptable. Therefore, as it is illustrated in the Figure 4.1, this condition satisfies at the acceptable intervals.

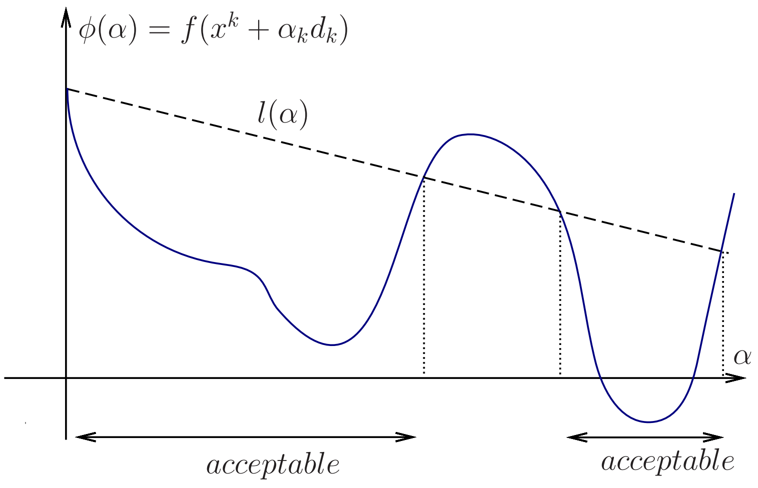


Figure 4.1: Sufficient decrease condition

However, the sufficient decrease condition by itself is not enough to guarantee the reasonable progress of the algorithm because a sufficient small values of α can satisfy that. For this, a second condition is introduced to exclude unacceptably short steps, called curvature condition, which needs α_k to satisfy

$$\nabla f(x_k + \alpha_k d_k)^T p_k \geqslant c_2 \nabla f_k^T d_k,$$

for some constant $c_2 \in (c_1, 1)$, where c_1 is the same constant in (4.15). Moreover, it guarantees that the slope of ϕ at α_k is greater than the initial slope $\phi'(0)$ times c_2. That is reasonable, because if $\phi(\alpha)$ is strongly negative, we can predict that f can be reduced significantly by moving further along the chosen direction. Now we can introduce and restate the *Wolfe conditions*, which are a combination of the sufficient decrease and curvature conditions, as illustrated in the Figure 4.2

$$f(x_k + \alpha_k d_k) \leqslant f(x_k) + c_1 \alpha \nabla f_k^T d_k, \tag{4.16}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geqslant c_2 \nabla f_k^T d_k, \tag{4.17}$$

with $0 < c_1 < c_2 < 1$.

It can happen that a new step is not be particularly close to the minimizer of ϕ but may still satisfy the Wolfe conditions [118]. In order to overcome this issue, the curvature condition is modified to enforce α_k to lie within a broad neighborhood of a stationary point or local minimizer of ϕ.

$$f(x_k + \alpha_k d_k) \leqslant f(x_k) + c_1 \alpha \nabla f_k^T d_k, \tag{4.18}$$

$$|\nabla f(x_k + \alpha_k d_k)^T d_k| \leqslant -c_2 |\nabla f_k^T d_k|, \tag{4.19}$$

with $0 < c_1 < c_2 < 1$. We refer to these conditions the strong Wolfe conditions as they exclude points that are far from stationary points, effectively enforcing a restriction on how positive the function $\phi'(\alpha_k)$ can be. It is concluded that for every function f which is smooth and bounded below there exist step lengths that satisfy the strong Wolf condition f; see [118, pp.35-36].

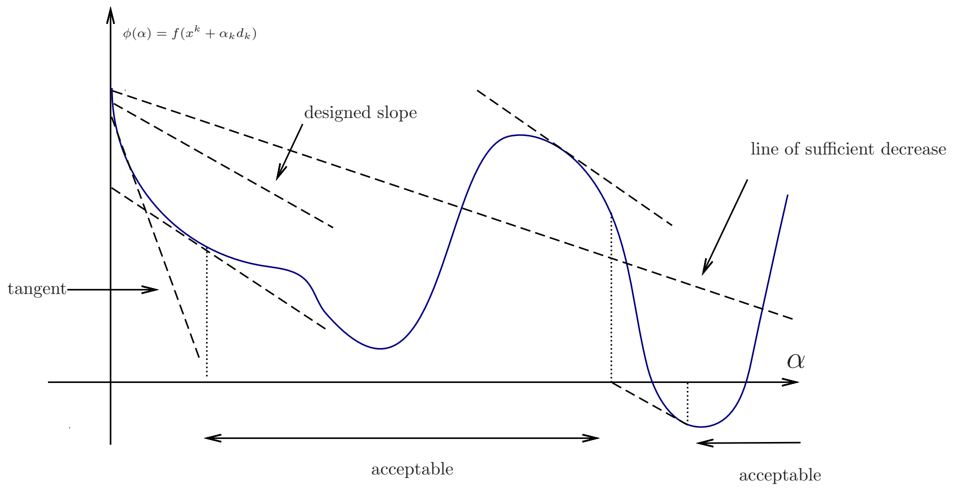

$\phi(\alpha) = f(x^k + \alpha_k d_k)$
designed slope
line of sufficient decrease
tangent
α
acceptable
acceptable

Figure 4.2: Sufficient decrease condition

Although we have mentioned that the sufficient decrease condition (4.16) by itself is not enough to guarantee that the algorithm produces sensible progress along the given direction, it can actually be applied without curvature condition (4.17). This is used in the so-called Backtracking-Armijo line search.

Algorithm 4.1. Given $\alpha_0 > 0$, for example $\alpha_0 = 1$ and let $l = 0$.

Until $f(x_k + \alpha_l d_k) \leqslant f(x_k) + c_1 \alpha_l \nabla f_k^T d_k$

set $\alpha_{l+1} = \tau \alpha_l$, where $\tau \in (0, 1)$,

set $l = l + 1$, and $\alpha_k = \alpha_l$.

As α_k will gradually become small enough to satisfy the sufficient decrease condition, then after a finite number of trials an acceptable step length will be found. The backtracking procedure guarantees that either that the chosen step length α_0 is some fixed value, or else that it is short enough to hold the sufficient decrease condition but not too short.

4.2.2 Convergence of generic line search method

In order to connect all of the previous together, we first need to introduce the *generic line search method* and then the general convergence result [47].

Algorithm 4.2. Given an initial guess x^0, let $k = 0$.

Until convergence, repeat

Compute a descent direction d_k at x^k.

Find a step length α_k using a backtracking-Armijo line search along d_k.

Update x^{k+1} by $x^k + \alpha_k d_k$, and set $k = k + 1$.

Theorem 4.7. Assume that the objective function $f \in C^1$ and that the gradient ∇f is Lipschitz continuous on $\mathbb{R}^n$. Then, the iterates obtained by the *generic line search method*, either

$$\nabla_l f = 0 \text{ for some } l \geqslant 0$$

or

$$\lim_{k\to\infty} f_k = -\infty$$

or

$$\lim_{k\to\infty} \min\left(|(d_k, \nabla_k f)|, \frac{|(d_k, \nabla_k f)|}{\|d_k\|_2} \right) = 0.$$

In other words, the first condition corresponding to finding a first-order stationary point in a finite number of iterations, a sequence of iterates are encountered for which the objective function is unbounded from below, or a normalized slope along the search direction approaches zero. The first two conditions are direct and acceptable consequence, the latter may not. The reason for that, as we can see the convergence of the gradient to zero has not be mentioned, the iterates may not ultimately be first-order critical as it may happen if the search direction and gradient tend to be mutually orthogonal. Thus requiring that d_k to be a descent direction is not included. However, the success of the *generic line search method* is coupled with the descent direction of the line search, that is occur when $d_k \nabla f_k < 0$, it guarantees the search directions are never too close to orthogonality with the gradient. This is in turn satisfied if $d_k = -\nabla f(x^k)$. Which considers the archetypical globally convergent algorithm, the so-called steepest descent method.

4.2.3 Steepest descent method

The method is one of the oldest methods and presents not only the key point for a theoretical analysis but also the standard of reference against which other advanced algorithms are measured. The method is defined by the line search methods (4.12) as

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \tag{4.20}$$

where α_k is a nonnegative scalar minimizing $f(x^k - \alpha_k \nabla f(x^k))$ and $\nabla f(x^k)$ is the gradient of the objective function f. In a short, the global convergence is obtained directly from the following theorem, we eliminate the proof which is clarified in [118, Chapter 3].

Theorem 4.8. Assume that the objective function $f \in C^1$ and that the gradient ∇f is Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates obtained by the *generic line search method* using steepest descent direction, either

$$\nabla_l f = 0 \text{ for some } l \geqslant 0$$

or

$$\lim_{k\to\infty} f_k = -\infty$$

or

$$\lim_{k\to\infty} \nabla f_k = 0.$$

The theorem suggests the globally convergent of the steepest descent method [47]. Although the approach is the standard globally convergent and comparably inexpensive algorithm, is quite slow in practice. Our aim is to incorporate the multigrid approach with the steepest descent features to obtain a method which guarantees a good global convergence and a rapid rate of convergence for variational inequality problems discretized by finite element methods.

4.2.4 Newton's methods

Newton's methods takes multiple forms throughout optimization. First, in the case of solving the nonlinear of equations $F(z) = 0$ ($z \in \mathbb{R}^n$), the Jacobian matrix of F is denoted by $J(z)$. Then, the Newton step d_k at the current point z_k is the solution of the linear system

$$J(z^k)d_k = -F(z^k), \tag{4.21}$$

provided that $J(z^k)$ is non singular. So that d_k is the direction from the point z^k to a zero of the local affine Taylor-series model of F.

However, the Newton step d_k for unconstrained minimization of $f(x)$ is constructed to minimize $f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T H(x^k) d$, a local Taylor series quadratic model of $f(x^k + d)$. Then, d_k is a solution of the linear system

$$H(x^k)d = -\nabla f(x^k), \tag{4.22}$$

Furthermore, for the case when the objective function $f(x)$ subjects to m linear equality constraints $Ax = b$. Then the Newton step d_k should be a minimization of the local Taylor-series quadratic model of f with satisfying the constraints $A(x^k + d_k) = b$, namely, d_k solves the following quadratic program

$$\min_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_k d + \nabla f_k^T d \ \text{ subject to } \ Ad = b - Ax^k, \tag{4.23}$$

where $H_k = H(x^k)$ and $\nabla f_k = \nabla f(x^k)$. Let y^{k+1} denote an estimate of the Lagrange multipliers for the equality constraints, then d_k and y_{k+1} solve the following $n + m$ linear equations

$$\begin{pmatrix} H_k & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} d_k \\ -y^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f_k \\ b - Ax^k \end{pmatrix}, \tag{4.24}$$

provided that the matrix is nonsingular, this can be satisfied if the reduced Hessian $N_A^T H_k N_A$ is positive definite and A has full rank, where N_A is a basis for the null space of A. Further, if $Ax^k = b$, then the second equation in the system becomes $Ad_k = 0$ which means that d_k must belong to the null space of A , see for example [117].

A pure Newton method for zero-finding is defined as $z^{k+1} = z^k + d_k$, with d_k defined by one of (4.21), (4.22) or (4.24) which generates a sequence of Newton iterates $\{z^k\}$ starts with an initial point z^0. The convergence of the method to a solution is quadratic under various conditions that can be quite restrictive, such as starting with an initial point close enough to a solution. However, the convergence from a general starting point can be ensured by performing either the line search approach or the trust region strategy. In the first approach the new iterate is defined by $z^{k+1} = z^k + \alpha_k d_k$, where the positive scalar α is selected to reduce a merit function that measures progress. In unconstrained optimization, the objective function is simply the merit function. While the second approach is based on defining the current iterate within a region which the local model can be trusted, we eliminate the detailed discussion but refer to [118] instead.

In addition, any method uses Newton direction can be called second order methods, the following theorem shows that the generic line search method with the use of Newton direction will usually converge very rapidly.

Theorem 4.9. Assume that the objective function f is twice differentiable and the Hessian $\nabla^2 f(x)$ is Lipschitz continuous in a neighborhood of a solution x^* such that the sufficient conditions are satisfied (4.1). Then, the iterates obtained by the *generic line search method*, with using the Newton direction in which the Hessian $\nabla^2 f(x)$ is positive definite and $\alpha_0 = 1$. Then

(a) $\alpha_k = 1$ for all sufficiently large k,

(b) the sequence $\{x^k\}$ convergence entirely to the solution x^*, and

(c) the rate of convergence is $Q-$quadratic, i.e., there is a constant $k \geqslant 0$ such that

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|_2}{\|x^k - x^*\|_2^2} \leqslant k.$$

The proof of the theorem is given precisely by Wright and Nocedal [118].

As we mentioned that a descent direction of the Newton method depends mainly on the positive definiteness of the Hessian matrix, however, there is a remedy for this, that is by considering the so-called modified Newton methods. This can be obtained in a number of approaches, the precaution is not to replace the matrix B_k in the search direction $d_k = -B_k^{-1}\nabla f(x^x)$ by the Hessian matrix $\nabla^2 f(x^k)$ but by B_k which is (or is close to) $\nabla^2 f(x^k)$ under some assumptions

[118, pp.48-56]. Further, if B_k represents an approximation to the Hessian matrix then we obtain the so-called Quasi-Newton method. We exclude introducing the details instead one can discover them in a wide range of references.

Finally, we preview the different class of search directions which is generated by nonlinear conjugate gradient methods, they take the form

$$d_k = -\nabla f(x^k) + \beta_k d_{k-1},$$

where β is a scalar that guarantees that d_k and d_{k-1} are conjugate, as we now discuss.

4.2.5 Nonlinear conjugate gradient method

Algorithm 2.2 describes a minimization algorithm for the convex quadratic function (2.10). This approach has been successfully adapted by Fletcher and Reeves [38] to minimize general convex functions, or even general nonlinear functions f.

We now describe the adaption to Algorithm 2.2 to handle nonlinear functions. First, the step length α_k (2.27), which is the one-dimensional minimizer of the quadratic function (2.10), is replaced by a suitable line search. In the case that the objective function f is nonlinear, the residual r simply represents the gradient of f. Finally, the scalar β_k is modified, as necessary, to guarantee that d_k is a descent direction at the point x^k. There are two principally known definitions of the scalar β_k, firstly, the Fletcher and Reeves formula

$$\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}, \tag{4.25}$$

and secondly the Polak and Ribére formula

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}, \tag{4.26}$$

These adaptations give rise to the following algorithm for nonlinear optimization.

Algorithm 4.3. Select x^0 as an initial point;

Compute $f_0 = f(x^0)$, $\nabla f_0 = \nabla f(x^0)$;

Set $d_0 = -\nabla f(x^0)$ and $k = 0$;

Do while $\nabla f(x^0) \neq 0$

Evaluate α_k (as described below) and set $x^{k+1} = x^k + \alpha_k d_k$;

Compute ∇f_{k+1};

Select a formula for β_{k+1}; (4.25) or (4.26);

$$d_{k+1} = -\nabla f_{k+1} + \beta_{k+1} d_k; \tag{4.27}$$

$$k = k + 1; \tag{4.28}$$

end(while)

In the above, we note that, the computation of β_{k+1} at each iteration of the algorithm requires only the evaluation of the objective function and its gradient, neither of which are matrix operations, a relatively small number of vectors storage are required. Therefore, the Algorithm 4.3 a practical iterative method for large nonlinear optimization problems. To complete this concern, we require the step length to satisfy the strong Wolfe conditions presented in (4.18) and (4.19). In conclusion, any line procedure with a step length adhering to the strong Wolfe conditions will guarantee that the resulting direction d_k is a descent direction for the function f. For the interested reader, further details are provided in [38, pp.121-122].

4.3 Minimization of constrained problems

In this section we outline approaches are involved in this work for solving constrained optimization problem. First, we describe a certain class of methods used for constrained optimization problems, typically referred to in the literature as projected gradient methods. They are known as primal methods, in that they work directly on the original problem by searching for the optimal solution inside of the feasible region. They are particularly efficient when applied to linearly constrained problems due to their (generally) simple applicability as well as the competitive rates of convergence when directly compared to other methods. Nevertheless, convergence can not be guaranteed for certain problems, particularly problems involving non-linear constraints. As well as requiring a certain procedure to obtain an initial feasible point, computational difficulties may arise during later iterations due to the necessity to stay within the feasible region.

For purely linearly constrained problems, one can consider application of projected gradient methods coupled with an active set strategy, as suggested by Luenberger [75, p.367-371], by only considering active inequality constraints.

We also consider examples of the second order methods which use the Hessian matrix such as interior-points and the penalty barrier multipliers methods. Although these methods are expensive because of evaluating the dense Hessian matrix especially for the large scale problems, they represent more general and effective methods for solving constrained optimization problem in which the first order difficulties can be avoided.

4.3.1 Projection gradient methods

The gradient projection method represents a modified ordinary steepest descent method for unconstrained problems. Therefore these methods solve constrained optimization problems by projecting the negative gradient onto the working space in order to define the direction of movement. The implementation of the projected gradient method is quite simple and it is very effective for large scale problem. Originally proposed by Goldstein [44] and Levitin [72]. Let introduce the general form of constrained optimization problems

$$\min\{f(x) : x \in \Omega\}. \tag{4.29}$$

f is a continuously differentiable function $f : H \to \mathbb{R}$ and Ω is a nonempty closed convex subset of a Hilbert space H, assuming that $H = \mathbb{R}^n$. The projection into Ω is the mapping $P : \mathbb{R}^n \to \Omega$, see [21], defined by

$$P_\Omega(x) = \arg\min \|z - x\| : z \in \Omega, \tag{4.30}$$

where $\|.\|$ is an inner product, the dependence of P on Ω is clear so we use $P(x)$ instead. Given the projection P into Ω, the method of projection gradient is given by the iteration

$$x^{k+1} = P(x^k - \alpha_k \nabla f(x^k)) \quad k = 0, 1, ..., \tag{4.31}$$

where $\alpha_k > 0$ denotes the step size and $\nabla f(x^k)$ the gradient of f at the point x^k.

4.3.2 Optimality conditions and step lengths

We first consider the main necessary and sufficient optimality conditions for constrained problems in its primal formulation. A feasible vector of the problem is a vector that satisfies the constraints, denoted by the domain Ω, of a given problem. A local minimum of f over the set Ω is a vector x^* if, there exists a $\varepsilon > 0$ such that

$$f(x^*) \leqslant f(x) \quad \forall x \in \Omega \text{ with } \|x - x^*\| < \varepsilon.$$

A global minimum of f amounts to a part that no worse than all other feasible vectors, namely

$$f(x^*) \leqslant f(x) \quad \forall x \in \Omega.$$

If the above inequalities are strict for $x \neq x^*$ then the corresponding local or global minimum x^* is said to be strict. In the case when the function f is convex, the optimality conditions amounts to the following proposition,

Proposition 4.10. Optimality Condition

(a) If x^* is a local minimum of f over Ω, then

$$\nabla f(x^*)^T(x - x^*) \geqslant 0, \forall x \in \Omega. \tag{4.32}$$

(b) The condition of part (a) is sufficient to minimize a convex function f over a convex set Ω.

The proof is given in [13]. A vector x^* is also called a stationary point if it satisfies the optimality condition (4.32).

With reference to the step size α_k in this case, it can be selected in several ways, in order to guarantee that the first order optimality conditions are satisfied for any limit point x^* of $\{x^k\}$. For the interested readers, different choices of step length are proposed in [21]. A first practical procedure to determine the step α_k was proposed by Bertsekas [12], commonly referred to as the Armijo procedure. Given β and μ in $(0, 1)$ and $\gamma > 0$, the procedure is defined as

$$\alpha_k = \beta^{m_k}\gamma, \tag{4.33}$$

where m_k is the smallest nonnegative integer such that

$$f(x^{k+1}) \leqslant f(x^k) + \mu(\nabla f(x^k), x^{k+1} - x^k), \tag{4.34}$$

where f is assumed to be continuously differentiable and Ω a convex set.

Here, we focus mainly on the case where the objective function is nonlinear and the set Ω is structured by equations and inequalities, convex set. However, the projected methods can be implemented for a number of different representations of the domain Ω, but with different tools. Examples of these, projection on affine subspaces [21] and projection on a feasible box, used in [28] for the solution of quadratic programming problems subject to box constraints with the so-called Barzilai- Borwein steplengths.

4.3.3 Projection on a convex set

We introduce the main idea of the Calamai procedure [21] for identifying the movement towards the local minimum for linearly constrained problems. His approach generalizes the Armijo procedure (4.33, 4.34) without requiring the bound on the step lengths. Assume that the mapping $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and Ω is a nonempty closed convex set. An iterate x_k is given in Ω by searching the following path,

$$x_k(\alpha) \equiv P(x_k - \alpha \nabla f(x_k)),$$

where $\alpha > 0$ denotes a step size and P the projection into Ω. The next step is defined by $x_{k+1} = x_k(\alpha_k)$, with the step size α_k chosen such that

$$f(x_{k+1}) \leqslant f(x_k) + \mu_1(\nabla f(x_k), x_{k+1} - x_k), \tag{4.35}$$

and

$$\alpha_k \geqslant \gamma_1 \text{ or } \alpha_k \geqslant \gamma_2 \overline{\alpha}_k > 0 \tag{4.36}$$

where $\overline{\alpha}_k > 0$ satisfies

$$f(x(\overline{\alpha}_k)) > f(x_k) + \mu_2(\nabla f(x_k), x(\overline{\alpha}_k) - x_k). \tag{4.37}$$

for constants $\mu_1, \mu_2 \in (0,1)$ and $\gamma_1, \gamma_2 > 0$. A sufficient decrease of the function is forced by condition (4.35) while condition (4.36) guarantees that the step α_k not too small. This is because Dunn [32] shows that if x_k is not a stationary point then (4.37) is not satisfied for all $\overline{\alpha}_k > 0$ sufficiently small. As a result of this, there exists an $\alpha_k > 0$ which satisfies (4.35) and (4.36) provided $\mu_1 \leqslant \mu_2$. This procedure corresponds to the Armijo conditions ((4.33), (4.34)) with $\gamma_1 = \gamma, \gamma_2 = \beta$ and $\mu_1 = \mu_2 = \mu$.

We now consider general algorithm by Calamai [21] for linearly constrained problems. By defining $A(.)$ to be the active set (4.6), we typically look to obtain x^{k+1} by selecting a descent direction d_k orthogonal to the constraints in $A(x^k)$ coupled with an appropriate line search along d_k. We then compute the update $x^{k+1} = x^k + \alpha_k d_k$ and set $A(x_{k+1}) = A(x_k)$ whenever a sufficient reduction is obtained by the line search algorithm for some $\alpha_k \in (0, \sigma_k)$, where σ_k is defined the distance along d_k to the nearest constraint not in $A(x_k)$. Otherwise, we set $x_{k+1} = x_k + \sigma_k d_k$ and $A(x_{k+1}) \subset A(x_k)$. As a consequence of these considerations, the following algorithm is given.

Algorithm 4.4 (Gradient Projection Algorithm)**.** Suppose that Ω is a closed convex set and the initial point $x_0 \in \Omega$ is given. For $k \geqslant 0$ choose either (a) or (b) to compute x_{k+1}.

(a) Let $x_{k+1} = P(x_k - \alpha_k \nabla f(x_k))$ where α_k satisfies (4.35), (4.36) and (4.37).

(b) Find $x_{k+1} \in \Omega$ such that $f(x_{k+1}) \leqslant f(x_k)$ and $A(x_{k+1}) \subset A(x_k)$.

For more detail about convergence analysis, we refer to [21], where the author also provides proof that the projected gradient method forces the sequence of projected gradient to zero. As a consequence, the active and binding constraints (4.7) are identified in a finite number of iterations if the projected gradient method convergences to nondegenerate point of a linearly constrained problem.

4.4 Interior point methods

The invention of interior point approach is credited to Narendra Karmarkar through the work [65] published in 1984. This is as a number of controversial copyright patents introduced in 1988 in pursuit of protecting a code that had been developed at that time. However, a fair amount of research into this area had already been examined by a number of authors prior to this date. In particular, the idea of accessory of the objective function with a penalty term to penalise movements close to the boundary was initially presented in 1955 by Frisch [40]. Later in 1961, a slightly different penalty approach was defined by Carrol in [22], however much of the credit for the theoretical and computational development of the interior point method can be appointed to both Fiacco and McCormick [36, 37], with particular note made to the 1968 monograph [36]. Over the years, a great number of theoretical results relating to interior point approaches have been produced, these include describing the convergence properties as well as suggestions for numerical implementation. For the interested reader, a more detailed account on the history and development of interior point methods is discussed by Shanno in [100, pp.55-64], and a condensed look at classical and recent research about interior point methods is presented by Wright in [117].

The interior point approaches operate by transforming inequality constrained problems into parameterized unconstrained problems via barrier or penalty term. They can be used to determine iteratively the solution to either linear or nonlinear constrained optimization problems. The objective function and constraint functions are denoted by f and g, respectively. Furthermore, we denote by ∇f the gradient of the function f and by $\nabla^2 f$ the Hessian. We consider the convex mathematical program

$$\min_{x \in \mathbb{R}^n} f(x) \tag{4.38}$$

subject to

$$c_i(x) \geqslant 0, \;\; i \in \mathcal{I}, \tag{4.39}$$

$$c_i(x) = 0, \;\; i \in \mathcal{E}, \tag{4.40}$$

with $\mathcal{I}, \mathcal{E}$ are used to denote two finite sets of indices for the inequality and equality constraints, respectively.

4.4.1 Primal barrier methods for constrained optimization

Barrier approaches became popular during the 1960s with particular mention to [36] presented by Fiacco and McCormick . Whilst, such approaches can be viewed as so-called primal interior point methods, they fell out of favour with the introduction of sequential quadratic programming methods. Furthermore as we will describe shortly, these approaches experience drawbacks when compared to so-called primal-dual interior point methods.

The main drawback is that the sequence of points $\{x_\mu\}$ becomes more and more difficult to approach the solution x^* as μ become close to zero because of the nonlinearity of the function $B(x; \mu)$, detailed below.

We now provide a mathematical description of Barrier approaches for inequality constrained problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) \;\text{ subject to }\; c_i(x) \geqslant 0, \; i \in \mathcal{I} \tag{4.41}$$

with $x \in \mathbb{R}^n$ used to denote primal variables. A barrier approach consists of introducing a nonnegative and continuous function $B(x)$ over the domain $\{x \mid c_i(x) > 0\}$ such that $B(x)$ approaches infinity as the boundary of this region is approached from the interior. The barrier function can be defined in a number of ways, however, for this work we consider the so-called logarithmic barrier function as follows

$$B(x, \mu) = -\sum_{i=1}^{n_I} \ln c_i(x), \tag{4.42}$$

where μ is a positive scalar referred to as the barrier parameter. Therefore, we look to determine the solution to the constrained minimization problem (4.38) to (4.40) by considering the following unconstrained minimization problem

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} B(x, \mu), \\ &\text{where} \quad B(x, \mu) = f(x) - \mu \sum_{i=1}^{n_I} \ln c_i(x), \end{aligned} \tag{4.43}$$

The idea behind the method is to start with a fixed parameter μ, say $\mu = 1$, and to solve the unconstrained auxiliary problem. Then the parameter μ is decreased under predefined rule and the auxiliary problem is solved again, and so on. Due to convexity, the unconstrained auxiliary problem (4.43) has a unique solution x_μ, so that as $\mu \to 0$ a smooth curve defined by the set of optimal solutions $\{x_\mu\}$ is generated, commonly referred to as the *central path*.
Then, for a given x^* satisfying the sufficient optimality conditions (Theorem 4.3), the sequence $\{x_\mu\}$ satisfies

$$\lim_{\mu \to 0} x_\mu = x^*, \tag{4.44}$$

with first order Lagrange multiplier estimate (denoted λ^*), the Lagrange barrier functions given as follows

$$\lambda_m := \frac{\mu}{c_j(x_\mu)}, \ \lim_{\mu \to 0} \lambda_m = \lambda_j^*. \tag{4.45}$$

The auxiliary problems are usually solved by Newton's method. However, the freedom in the choice of the penalty function in the classical approaches leads to serious computational difficulties. Firstly, the barrier term ensures that the central path stay away from the region of quadratic convergence, meaning that use of Newton's method may result in extremely short steps. Secondly, the Newton method becomes inefficient and may even leave the feasible region when longer steps are taken to a rapid reduction in the value of μ. Finally, the idea of staying on the central path and solving the auxiliary problems is too restrictive, as discussed in [10]. Advances have since been developed in order to overcome these issues. In particular, these include imposing restrictions on the choice of penalty parameter to a certain class of functions and also relaxing the requirement to remain on the central path. Furthermore, the development of so-called primal-dual interior point methods (to be presented shortly) have been formulated in advances such approaches in recent years, with one such approach (penalty barrier multiplier method) used to good effect in the field of topology optimization [64].

4.4.2 Interior-point for nonlinear programming

Nowadays, interior point methods are considered among the most robust algorithms for large-scale nonlinear programming. Although the main ideas, such as primal-dual steps, are taken directly from linear programming, there are still a number of important challenges to resolve, depending on the problem at hand. Examples of these include the treatment of of nonconvexity, the mechanism for updating the barrier parameter in the presence of nonlinearities and the need to guarantee convergence toward the solution [118].

We consider the following modified formulation of (4.38)-(4.40) for the related analysis of this section

$$\min_{x,s} f(x) \tag{4.46}$$

subject to

$$c_{\mathcal{I}}(x) - s = 0, \quad s \geqslant 0, \tag{4.47}$$

$$c_{\mathcal{E}}(x) = 0, \tag{4.48}$$

where $c_{\mathcal{I}}$ denote a vector of scalar functions $c_i(x)$, $i \in \mathcal{I}$, and similarly for $c_{\mathcal{E}}$. The vector s denotes slack variables which are used to transform the inequality vector $c_{\mathcal{I}}$ into equality. Interior-point method can be derived based on continuation methods or barrier methods. We start with the derivation associated with the continuation approach. The KKT conditions (4.4) for the nonlinear problem (4.46)-(4.48) can be written as

$$\nabla f(x) - A_{\mathcal{E}}^T y - A_{\mathcal{I}}^T z = 0 \tag{4.49}$$

$$SZ - \mu e = 0 \tag{4.50}$$

$$c_{\mathcal{I}}(x) - s = 0, \tag{4.51}$$

$$c_{\mathcal{E}}(x) = 0,, \tag{4.52}$$

with $\mu = 0$, together with

$$s \geqslant 0, \;\; z \geqslant 0. \tag{4.53}$$

In the above, we denote by $A_{\mathcal{E}}(x)$ and $A_{\mathcal{I}}(x)$ the respective Jacobian matrices for the functions $c_{\mathcal{E}}$ and $c_{\mathcal{I}}$, with y and z corresponding to the associated Lagrange multipliers. Furthermore, $e = (1, 1, ..., 1)^T$ and the matrices S and Z used to denote diagonal matrices with entries corresponding to the vectors s and z, respectively.

The second interpolating of interior-point methods is based on barrier methods to the problem. Based on the presentation in Section 4.4.1, we consider the following formulation

$$\min_{x,s} f(x) - \mu \sum_{i=1}^{n_{\mathcal{I}}} \log s_i \tag{4.54}$$

subject to

$$c_{\mathcal{I}}(x) - s = 0, \quad s > 0, \tag{4.55}$$

$$c_{\mathcal{E}}(x) = 0, \tag{4.56}$$

where μ is a positive parameter and $\log(.)$ represents the natural logarithm function. This approach involves finding an approximate solution of the problem (4.54)-(4.56) for a sequence of positive barrier parameters μ_k that reduce to provide an effective approximation to zero for sufficiently large k. By writing down the KKT conditions for the problem (4.54)-(4.56)

$$\nabla f(x) - A_{\mathcal{E}}^T y - A_{\mathcal{I}}^T z = 0 \tag{4.57}$$

$$-\mu S^{-1} e + Z = 0 \tag{4.58}$$

$$c_{\mathcal{I}}(x) - s = 0, \tag{4.59}$$

$$c_{\mathcal{E}}(x) = 0,, \tag{4.60}$$

the similarity between the KKT conditions for the formulation (4.38)-(4.40) are evident. The main difference is in the second equation, which becomes quite nonlinear close to a KKT point as s approaches zero. To aid Newton's method, the rational equation (4.58) can be multiplied on both sides by S to yield a quadratic equation. Such a move is possible due to the structure of S (positive diagonal matrix) and will not result in a different solution. When compared to early barrier methods [37], the main advantage gained from the reformulation (4.46)-(4.48) advanced through the introduction of such variables $s > 0$ is that the resulting method can be started with any initial point x^0. Furthermore, the approach will remain interior (hence the term interior point) with respect to the constraints $s > 0$, $z > 0$, .

4.4.3 Basic interior-point algorithm

We can apply Newton's method to the nonlinear system (4.57)-(4.60) in the variables x, s, y, z to obtain

$$\begin{bmatrix} \nabla^2_{xx} L & 0 & -A^T_{\mathcal{E}}(x) & -A^T_{\mathcal{I}}(x) \\ 0 & Z & 0 & S \\ A_{\mathcal{E}}(x) & 0 & 0 & 0 \\ A_{\mathcal{I}}(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ d_y \\ d_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A^T_{\mathcal{E}}(x) y - A^T_{\mathcal{I}}(x) z \\ Sz - \mu e \\ c_{\mathcal{E}}(c) \\ c_{\mathcal{I}}(x) - s \end{bmatrix}. \tag{4.61}$$

Where L denotes the Lagrangian for (6.13)-(6.15)

$$L(x, s, y, z) = f(x) - y^T c_{\mathcal{E}}(x) - z^T (c_{\mathcal{I}}(x) - s). \tag{4.62}$$

The system (4.61) represents the so-called primal-dual system with solution (or step) $d = (d_x, d_s, d_y, d_z)$. This step is then used to compute the new iterate (x^+, s^+, y^+, z^+) as follows

$$\begin{aligned} x^+ &= x + \alpha_s^{\max} d_x, \quad s^+ = s + \alpha_s^{\max} d_s, \\ y^+ &= y + \alpha_z^{\max} d_y, \quad z^+ = z + \alpha_z^{\max} d_z, \end{aligned} \tag{4.63}$$

where,

$$\begin{aligned} \alpha_s^{\max} &= \max\{\alpha \in (0, 1] : s + \alpha d_s \geqslant (1 - \tau) s\}, \\ \alpha_z^{\max} &= \max\{\alpha \in (0, 1] : z + \alpha d_z \geqslant (1 - \tau) z\}, \end{aligned} \tag{4.64}$$

with $\tau \in (0, 1)$. The straightforward iteration (4.63) represents the basis of modern interior-point methods, with different adoptions developed to handle nonconvexities and nonlinearities. The other major component involves the selecting of the so-called barrier parameters $\{\mu_k\}$ at each iteration. This can be approached in one of two ways. The first is to fix the barrier parameters for a series number of iterations until the KKT conditions (4.57)-(4.60) are satisfied to a prescribed accuracy, as investigated by Fiacco and McCormick [37]. The second approach involves updating the barrier parameter at each iteration. For the interested reader, Nocedal and Wright [118, pp.572-574] provide further details for both approaches as well mechanisms for dealing with issues such as non-convexity and singularity.

The notable observation that can be read for the system(4.61) is that if x^* is a solution satisfying strict complementarity, then for each index i either s_i or z_i maintains strictly positive as the iterates approach x^*. This ensures that the primal-dual matrix in (4.61) remains nonsingular, allowing for a fast rate of convergence to be established, as discussed in [118]. The KKT conditions for the perturbed system (4.57)-(4.60) can be used to define an appropriate error function

$$Er(x,s,y,z;\mu) = \max\{\|\nabla f(x) - A_{\mathcal{E}}(x)^T y - A_{\mathcal{I}}^T z\|, \|Sz - \mu e\|, \|c_{\mathcal{E}}(x)\|, \|c_{\mathcal{I}}(x) - s\|\}, \tag{4.65}$$

for some vector norm $\|.\|$, allowing for the presentation of a primal-dual iteration point algorithm as follows

Algorithm 4.5. Select initial values for x^0 and $s^0 > 0$, and compute initial values of the multipliers y^0 and $z^0 > 0$ from (4.57-4.60). Choose an initial barrier parameter $\mu_0 > 0$ and parameters $\sigma, \tau \in (0,1)$. Set $k = 0$.

```
repeat until satisfying a stopping test for the nonlinear program (6.13)-(6.15),
    repeat until $Er(x,s,y,z;\mu) \leqslant \mu_k$,
    solve (4.61) to get the search direction $d = (d_x, d_s, d_y, d_z)$;
    Evaluate $\alpha_s^{\max}$, $\alpha_z^{\max}$ using (4.64);
    Evaluate $(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1})$ using (4.63)
    Set $\mu_{k+1} = \mu_k$ and $k = k + 1$;
    end
Choose $\mu_k \in (0, \sigma\mu_{k+1})$
end
```

4.4.4 Algorithm development

Algorithm (4.5) has been updated and extended in order to handle nonconvex nonlinear problems which can be started from any initial estimate. Then, the primal-dual system (4.61) is rewritten in the following symmetric form

$$\begin{bmatrix} \nabla^2_{xx}L & 0 & A^T_{\mathcal{E}}(x) & A^T_{\mathcal{I}}(x) \\ 0 & \Sigma & 0 & -I \\ A_{\mathcal{E}}(x) & 0 & 0 & 0 \\ A_{\mathcal{I}}(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ -d_y \\ -d_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A^T_{\mathcal{E}}(x)y - A^T_{\mathcal{I}}(x)z \\ z - \mu S^{-1}e \\ c_{\mathcal{E}}(x) \\ c_{\mathcal{I}}(x) - s \end{bmatrix}, \quad (4.66)$$

where

$$\Sigma = S^{-1}Z. \quad (4.67)$$

Using this formulation reduces the computational work of each iteration because of permitting the use of a symmetric linear equations solver.

The formulation (4.66) with the defined Σ produces also the primal-dual system. However, the primal system can also be obtained by applying Newton' method directly to the optimality conditions (4.57)-(4.60) of the barrier problem and then summarizing the iteration matrix. This also leads to the system (4.66) but with Σ defined by

$$\Sigma = \mu S^{-2}. \tag{4.68}$$

4.4.5 Solving the primal-dual system

In spite of the evaluating cost of the problem functions and their derivatives, the solution of the primal-dual system (4.66) and (4.67) dominates the work of the interior-point iteration. An efficient linear solver is therefore essential in order practically solve large-scale problems. This includes using either iterative techniques or sparse factorization. We will consider iterative approaches preconditioned by multigrid iteration in scope of topology optimization problems, such as a multigrid conjugate gradient method, see Chapter 7.
We first note that, the system can be simplified using fairly standard techniques. In particular, the size of the system may be reduced by eliminating d_s using the second equation in (4.61), resulting in

$$\begin{bmatrix} \nabla^2_{xx}L & A^T_{\mathcal{E}}(x) & A^T_{\mathcal{I}}(x) \\ A_{\mathcal{E}}(x) & 0 & 0 \\ A_{\mathcal{I}}(x) & 0 & -\Sigma^{-1} \end{bmatrix} \begin{bmatrix} d_x \\ -d_y \\ -d_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A^T_{\mathcal{E}}(x)y - A^T_{\mathcal{I}}(x)z \\ c_{\mathcal{E}}(c) \\ c_{\mathcal{I}}(x) - \mu Z^{-1}e \end{bmatrix}, \tag{4.69}$$

This in turn can be reduced further by using the last equation to eliminate d_z which results in the condensed coefficient matrix written as

$$\begin{bmatrix} \nabla^2_{xx}L + A^T_{\mathcal{I}}\Sigma A_{\mathcal{I}} & A^T_{\mathcal{E}}(x) \\ A_{\mathcal{E}}(x) & 0 \end{bmatrix}, \tag{4.70}$$

which is much smaller than the coefficient matrix (4.66) when the number of inequality constraints is large. For all the symmetric forms (4.66), (4.69) and (4.70), the primal-dual system is ill conditioned, since by (4.67) certain elements of Σ converge to zero whilst the others diverge to ∞ as μ approaches zero. However, the Newton direction can be computed accurately by a stable direct factorization method, because of the special form in which this ill conditioning arises. Nevertheless, approximation of the slacks s or multipliers z too close to zero (or when the Hessian $\nabla^2_{xx}L$ or the Jacobian matrix A_E is almost rank deficient) results in damaging errors. Consequently, approaches based in a direct factorizing are commonly used for computing steps for interior point methods.

Iterative solution methods for linear systems can also be used for the computation of the primal-dual step. Here ill conditioning is a serious concern, requiring the use of suitable preconditioner that cluster the eigenvalues of Σ. Furthermore, constructing such a preconditioner is not a difficult task. For instance, one can apply the change of variables $d'_s = S^{-1}d_s$ in the system (4.66) and multiply the second equation by S which transfers the term Σ into $S^T\Sigma S$. Consequently, the eigenvalues of $S^T\Sigma S$ will cluster around μI as $\mu \to 0$ due to (4.67) (along with the associated KKT conditions).

An iterative methods such as GMRES, QMR or LSQR (see [45]) can be applied to one of the symmetric indefinite systems (4.66), (4.69) or (4.70). The conjugate gradient method cannot be applied directly to solve these systems as it is designed for positive definite systems. However, if the dimension of the constraint space is small one can consider a null space approach with the conjugate gradient method applied to the resulting system with positive definite system matrix.

4.4.6 Updating the barrier parameter

As shown in Algorithm 4.5, the sequence of barrier parameters are required to tend to zero as the sequence of iterates $\{x_\mu\}$ approaches the solution of the nonlinear programming problem (4.54)-(4.56). whilst a number of approaches have been developed, it is important to damp the barrier parameter appropriately at each iteration in order to avoid well known issues. Further to reduce the barrier parameter sufficiently will lead to a large number of iterations to advance convergence, whereas the slacks s or multipliers z may converge zero prematurely if reduced too quickly, slowing progress of the solution method.

A popular technique, is the so-called FiaccoMcCormick approach, which proceeds by fixing the barrier parameter until the perturbed KKT conditions (4.57)-(4.60) are satisfied to prescribed tolerance, upon which the parameter is damped as follows

$$\mu_{k+1} = \sigma_k \mu_k, \quad \text{with} \quad \sigma_k \in (0, 1). \tag{4.71}$$

The parameter σ_k can be set as a constant for all iterations. However, it is preferable to let σ_k alternate between two different values (for instance, 0.2 and 0.1), starting with the larger value and alternating to the smaller value when significantly progress is made by the latest iteration towards the solution. Furthermore, using [118] superlinear of convergence can be obtained if σ is allowed to tend to zero close to the solution with the parameter τ in (4.64) set to converge to 1. Whilst, the FiaccoMcCormick approach is straightforward to implement and apply, difficulties may arise with regard to the scaling of the problem as well as the choice of initial point and initial μ_0. Alternatively, adapting strategies may be considered that aim to be more robust when faced with such difficulties. . In contrast to the Fiacco-McCormick approach, barrier parameter is altered at each iteration based on the progress of the algorithm. Typically, a complementarity condition will be imposed of the form

$$\mu_{k+1} = \sigma_k \frac{s_k^T z_k}{n}, \tag{4.72}$$

with n denotes the size of the problem. One choice of σ_k is defined as

$$\sigma_k = 0.1 \min\left(0.05 \frac{1-\xi}{\xi}, 2\right)^3, \text{ where } \xi = \frac{\min_i [s^k]_i [z^k]_i}{(s^k)^T z^k / n}. \tag{4.73}$$

This is based on the deviation of the smallest complementarity product $[s^k]_i [z^k]_i$ from the average, where $[s^k]_i$ denotes the respective ith component of the iterate s^k , and for $[z^k]_i$. For the interested reader, further approaches may be found in [118, p.573].

Chapter 5

A First-Order Multigrid Method for Bound-Constrained Convex Optimization

5.1 Introduction

The aim of this chapter is to design an efficient multigrid method for constrained convex optimization problems arising from discretization of some underlying infinite dimensional problems. Due to problem dependency of this approach, we only consider bound constraints with (possibly) a single equality constraint. As our aim is to target large-scale problems, we want to avoid computation of second derivatives of the objective function, thus excluding Newton like methods. We propose a smoothing operator that only uses first-order information and study the computational efficiency of the resulting method.

Multigrid methods were originally developed for the solution of large systems of linear algebraic equations arising from discretization of partial differential equations. Their practical utility and efficiency were demonstrated by Achi Brandt in his pioneering papers [16, 17]. It has been well-known since the dark ages of multigrid that the methods may loose their superior efficiency when used for slightly different type of problems, namely the linear complementarily problems (LCP) [19, 59, 78]. This is caused by the presence of unilateral obstacles (or box constraints in the optimization formulation of the problem). The fact that the sets of active constraints may vary for different discretization levels, and that the constraints may not even be recognized on very coarse meshes, may lead to poor quality of the coarse level corrections and, in effect, to significant deterioration or even loss of convergence of the method. Various remedies have been proposed by different authors [19, 59, 62, 63, 78]; these usually resulted in "conservative" methods that were often significantly slower than standard methods for linear systems. Finally Kornhuber [69] proposed a truncated monotone multigrid method for LCP problems. This method has the property that as soon as the set of active constraints is correctly identified, the method converges with the same speed as without the presence of the constraints.

Not many attempts have been done to generalize the multigrid technique to the solution of optimization problems. From the successful ones, most focused on unconstrained problems [39, 51, 53, 73, 82]. In this case, the problem can be often identified with a discretized nonlinear PDE and thus techniques of nonlinear multigrid can be used. These techniques use almost exclusively a variant of the Newton method or the Newton-Gauss-Seidel method as a smoother. Hence they require computation and storage of the Hessian of the underlying optimization problem at each iteration. This may be very costly or even prohibitive for some large-scale problems. Moreover, one step of the Newton-Gauss-Seidel method has high computational complexity and cannot be directly parallelized. Therefore, our goal is to use a smoother that only relies on first-order information of the optimization problem. Used only on the finest level, it may be very inefficient, as compared to the Newton method, but this is where the coarse grid correction will help, just as in the linear case.

Treating general (equality or inequality) constraints by multigrid may be difficult, if not impossible, as we may not be able to find the corresponding restriction operators. If the number of constraints is directly proportional to the number of variables (such as for the bound constraints), the restriction operator for these constraints may be based on that for the variables. On the other hand, if the number of constraints is independent of the discretization (e.g., a single equality constraint) then the prolongation/restriction is simply the identity. All other situations are difficult, in our opinion. For this reason, all articles on multigrid for constrained problems either treat the bound-constrained problems or problems with a single equality constraint (e.g., [51, 52, 110]) or assume that a restriction operator for the constraints exists ([83]).

The first, straightforward goal of this Chapter is to extend Kornhuber's technique [69] to nonlinear convex optimization problems with bound constraints. The author is not aware of this generalization in the existing literature. The second goal is to propose a smoothing operator that would only use first-order information, and study the efficiency of the resulting method. Finally, we extend the developed algorithm to problems with an additional equality constraints. We study the behaviour of the proposed algorithms on a number of numerical examples.

The chapter is structured as follows. In Section 2 we introduce three multigrid algorithms for bound-constrained convex optimization problems; first Kornhuber's truncated correction scheme multigrid for bound-constrained quadratic problems [69], then its generalization to convex problems using the full approximation scheme and finally a version of the latter algorithm without truncation. This version is a new though minor modification of an existing method. Section 3 shows that the third algorithm can be easily extended to problems with an additional linear equality constraint that includes all variables. In Section 4 we introduce the first-order smoother—the gradient projection method with a gradient-based line search—and analyse its smoothing properties. Section 5 is devoted to numerical experiments: we, in particular, focus on the dependence of the convergence rate on the number of refinement levels and the number of the smoothing steps.

5.2 Multigrid for bound-constrained optimization

5.2.1 The problem

We consider the bound-constrained nonlinear optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \tag{5.1}$$

subject to

$$\varphi_i \leqslant x_i \leqslant \psi_i, \quad i = 1, \ldots, n$$

where f is a *convex* continuously differentiable nonlinear function and $\varphi_i < \psi_i$ for all i. To guarantee existence of a solution, we assume that either *the function f is coercive* (i.e., $f(x) \to \infty$ as $\|x\| \to \infty$) or that *the feasible set*

$$\mathcal{F} = \{x \in \mathbb{R}^n \mid \varphi_i \leqslant x_i \leqslant \psi_i,\ i = 1, \ldots, n\}$$

is bounded.

In order to define and use a multigrid method, we assume that there is a nested sequence of optimization problems arising from finite element discretizations of some underlying problem:

$$\min_{x^{(k)}\in\mathbb{R}^{n_k}} f_k(x^{(k)}) \tag{5.2}$$

subject to

$$\varphi_i^{(k)} \leqslant x_i^{(k)} \leqslant \psi_i^{(k)}, \quad i \in \mathcal{I}_k\,,$$

where $k = 0, 1, \ldots, j$. Here j is the finest discretization level corresponding to the original problem (5.1) and 0 is the coarsest level. The set $\mathcal{I}_k$ contains indices of n_k finite element vertices on discretization level k and f_k are the discretizations of the same infinite dimensional nonlinear function.

The communication between two consecutive levels $k-1$ and k is maintained by the prolongation and restriction operators $I_{k-1}^{k} : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ and $I_k^{k-1} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k-1}}$, respectively, whereas $I_k^{k-1} = c(I_{k-1}^{k})$. For the specific choice of these operators, see the last section.

For a feasible vector $x^{(k)}$ we introduce the sets of active indices:

$$\begin{aligned} \mathcal{A}_k^{\varphi}(x^{(k)}) &:= \left\{ i \in \mathcal{I}_k \mid x_i^{(k)} = \varphi_i^{(k)} \right\} \\ \mathcal{A}_k^{\psi}(x^{(k)}) &:= \left\{ i \in \mathcal{I}_k \mid x_i^{(k)} = \psi_i^{(k)} \right\} . \end{aligned}$$

We will always assume that

$$\mathcal{A}_k^{\varphi}(x^{(k)}) \cap \mathcal{A}_k^{\psi}(x^{(k)}) = \emptyset \, .$$

Finally, we denote the set of free variables by

$$\mathcal{A}_k^{0} := \mathcal{I}_k \setminus (\mathcal{A}_k^{\varphi}(x^{(k)}) \cup \mathcal{A}_k^{\psi}(x^{(k)})) \, .$$

Let us now introduce a *modified* problem

$$\min_{x^{(k)} \in \mathbb{R}^{n_k}} f_k(x^{(k)}) - v_k^T x^{(k)} \tag{5.3}$$

subject to

$$\varphi_i^{(k)} \leqslant x_i^{(k)} \leqslant \psi_i^{(k)}, \quad i \in \mathcal{I}_k \, ,$$

where $v_k \in \mathbb{R}^{n_k}$, together with the operator

$$x_{\text{new}}^{(k)} = opt(f_k, \varphi^{(k)}, \psi^{(k)}, v_k; x^{(k)}, \varepsilon, \nu)$$

that will play a role of the smoother for (5.3). The input are the problem data and the initial point $x^{(k)}$, the required precision ε and the maximum number of iterations allowed ν. We assume that *opt* is a descent convergent algorithm for the solution of problem (5.3). In Section 5.4 we will discuss the choice of *opt* in detail.

At the end of this section, let us briefly recall the main idea of multigrid methods. For simplicity, we restrict ourselves to quadratic objective functions $f_k(x^{(k)}) = \frac{1}{2}(x^{(k)})^T Q_k x^{(k)} - q_k^T x^{(k)}$, i.e., equivalently, to systems of linear equations $Q_k x^{(k)} = q_k$. We start at the finest level $k = j$ and apply some steps of the smoother *opt* to get an update $x^{(k)}$. We then compute the residuum $r_k = Q_k x^{(k)} - q_k$ for the current iteration. If we now solved the correction equation $Q_k e^{(k)} = r_k$, we would obtain the exact solution in one step as $x_*^{(k)} = x^{(k)} - e^{(k)}$. This would, however, be as costly as the solution of the original problem. So instead we solve the correction equation on the coarser level, $Q_{k-1} e^{(k-1)} = r_{k-1}$ with $r_{k-1} = I_k^{k-1} r_k$, either exactly or approximately by repeating the same procedure on this level. The correction is then interpolated back to level k and a new iteration is computed as $x^{(k)} := x^{(k)} - I_{k-1}^k e^{(k-1)}$. Finally, we again apply a few steps of the smoother. This is the so-called Correction Scheme (CS) multigrid algorithm. It splits the approximation error into two components, smooth and coarse (relative to the current level); the smooth component is reduced by the smoothing operator, while the coarse component by the coarse grid correction.

5.2.2 Truncation

We now recall the idea of truncation introduced by Kornhuber [69]; see also [48]. Consider the sequence of problems (5.2) obtained by standard finite element discretization of some infinite dimensional problem. Denote by $\lambda_i^{(k)}$ the finite element basis function associated with the i-th node on discretization level k.

Let $x^{(j)}$ be an approximate solution of (5.2) on the finest discretization level and $\mathcal{A}_j = (\mathcal{A}_j^{\varphi}(x^{(j)}) \cup \mathcal{A}_j^{\psi}(x^{(j)}))$ the corresponding set of active constraints. We will *truncate* the basis functions $\lambda_i^{(j)}$ by putting them equal to zero at the active nodes $\alpha \in \mathcal{A}_j$:

$$\tilde{\lambda}_i^{(j)}(x_\alpha^{(j)}) = 0 \quad \text{if} \quad \alpha \in \mathcal{A}_j \quad \text{for all } i\,.$$

The idea is to perform the coarse grid correction in next iteration of the multigrid method with the truncated basic functions, instead of the original ones. The basis functions for coarse levels are derived from the truncated basis functions on the finest level. Roughly speaking, we consider the active nodes on the finest level to be fixed by homogeneous Dirichlet boundary condition for the next coarse grid correction. After that, we perform the smoothing with the original basis functions; this may change the set of active nodes, so we update the truncated basis and repeat the procedure. Clearly, once the exact active set is detected, the truncated basis does not change any more and the "truncated" problem reduces to an unconstrained problem, just as in the classic active set strategy.

The way how to implement the truncation depends on the function f_j. For instance, if f_j is a quadratic function $f_j(x^{(j)}) = \frac{1}{2}(x^{(j)})^T Q_j x^{(j)}) + q_j^T (x^{(j)})^T$ then the truncation of the "stiffness" matrix Q_j amounts to putting all rows and columns with indices from $\mathcal{A}_j$ equal to zero, and analogously for q_j. If f_j can still be expressed as a function of Q_j and q_j, such as problems in Examples 5.5.2 and 5.5.4, the truncation is analogous. If f_j is a function defined by means of local stiffness matrices $(A_j)_i$, as in the minimal surface problem in Example 5.5.4, the truncation is performed for all these matrices (again by putting respective rows and columns equal to zero), and the function is evaluated using these truncated matrices.

Let us stress out that the truncation is performed explicitly only on the highest discretization level; on coarser grids it is inherited by means of prolongation/restriction operators. This process is again problem dependent. If f_k is defined by means of the global stiffness matrix Q_k, then $Q_{k-1} = I_k^{k-1} Q_k I_{k-1}^k,\ k = 1, \ldots, j$. If f_k is defined using local stiffness matrices $(A_j)_i$, then $(A_{k-1})_i = I_k^{k-1}(A_k)_i I_{k-1}^k,\ k = 1, \ldots, j$, for all i, etc.

In the following, we denote the truncation operator by trun and will use notation such as $\mathrm{trun}\, Q_j$ and $\mathrm{trun}\, f_j$.

5.2.3 Correction scheme truncated multigrid for quadratic problems

In this section we recall the Correction Scheme (CS) version of the truncated monotone multigrid algorithm introduced by Kornhuber [48, 69]; see also [71, 77, 78]. The analysis of the convergence rate of the monotone multigrid is established by Badea [6]. We assume that the objective function f is quadratic, so that

$$f_k(x^{(k)}) = \frac{1}{2}(x^{(k)})^T Q_k x^{(k)} + q_k^T x^{(k)} \tag{5.4}$$

where Q_k are positive definite matrices of size n_k and $q_k \in \mathbb{R}^{n_k}$. Accordingly, in this section we use the following notation for the smoother:

$$x^{(k)}_{\text{new}} = opt(Q_k, q_k, \varphi^{(k)}, \psi^{(k)}, v_k; x^{(k)}, \varepsilon, \nu)\,.$$

We introduce two additional restriction operators for the bound constraints:

$$R_k^{\varphi}: \quad (R_k^{\varphi} y)_i = \min\{y_j \mid j \in \mathcal{I}_k \cap \text{int supp}\, \lambda_i^{(k-1)}\}, \quad i \in \mathcal{I}_{k-1} \tag{5.5}$$

$$R_k^{\psi}: \quad (R_k^{\psi} y)_i = \max\{y_j \mid j \in \mathcal{I}_k \cap \text{int supp}\, \lambda_i^{(k-1)}\}, \quad i \in \mathcal{I}_{k-1}\,, \tag{5.6}$$

here int supp $\lambda_i^{(k-1)}$ denotes the interior of the support of the (coarse) basis function $\lambda_i^{(k-1)}$ and those includes fine-level nodes "close" to the coarse-level node i.

The following is a reformulation of Algorithm 5.10 from [48] in our notation.

Algorithm 5.1. (truncated CS, V-cycle for quadratic problems)

```
Set $\varepsilon, \varepsilon_0$. Initialize $x^{(j)}$.

for $i = 1 : niter$

      $x^{(j)} := mgm(j, x^{(j)}, q_j, \varphi^{(j)}, \psi^{(j)})$
```

```
        test convergence
end
function $x^{(k)} = mgm(k, x^{(k)}, r_k, \varphi^{(k)}, \psi^{(k)})$
        if $k = 0$
            $x^{(k)} := opt(Q_k, q_k, \varphi^{(k)}, \psi^{(k)}, 0; x^{(k)}, \varepsilon_0, \nu_0)$          (coarsest grid solution)
        else
            $x^{(k)} := opt(Q_k, q_k, \varphi^{(k)}, \psi^{(k)}, 0; x^{(k)}, \varepsilon, \nu 1)$          (pre-smoothing)
            $r_k = q_k - Q_k x^{(k)}$          (residuum)
            if $k = j$
                $Q_k := \mathrm{trun}\, Q_k$          (matrix truncation)
                $r_k := \mathrm{trun}\, r_k$          (residuum truncation)
            end
            $Q_{k-1} = I_k^{k-1} Q_k I_{k-1}^k$          (coarse grid matrix definition)
            $r_{k-1} = I_k^{k-1} r_k$          (residuum restriction)
            $\varphi^{(k-1)} = R_k^{\varphi}(\varphi^{(k)} - x^{(k)})$          (coarse grid bounds)
            $\psi^{(k-1)} = R_k^{\psi}(-\psi^{(k)} + x^{(k)})$          (coarse grid bounds)
            $v^{(k-1)} = mgm(k-1, 0_{n_{k-1}}, r_{k-1}, \varphi^{(k-1)}, \psi^{(k-1)})$          (coarse grid corr.)
            $x^{(k)} := x^{(k)} + I_{k-1}^k v^{(k-1)}$          (solution update)
            $x^{(k)} := opt(Q_k, q_k, \varphi^{(k)}, \psi^{(k)}, 0; x^{(k)}, \varepsilon, \nu_2)$          (post-smoothing)
```

```
    end
```

It is shown in [48] that the above algorithm converges for any initial iterate.

5.2.4 Full approximation scheme truncated multigrid for general problems

In the above CS algorithm, the coarse grid correction is used to correct the error in the current solution. That is, the solution on the coarse level is just a correction of the fine-level error, not an approximate solution of the original problem. For nonlinear problem, it is useful to solve on every level an approximation of the original problem. This is readily obtained by replacing the restricted residuum $I_k^{k-1}(q_k - Q_k x^{(k)})$ in Algorithm 5.1 by the "true" coarse-level right-hand side corrected by the approximation error, i.e., by $q_{k-1} + (Q_{k-1} I_k^{k-1} x^{(k)} - I_k^{k-1} Q_k x^{(k)})$. The solution update is then changed accordingly. This gives rise to the so-called Full Approximation Scheme (FAS) algorithm. The FAS version of Algorithm 5.1 for general nonlinear problems of type (5.2) is given below.

Algorithm 5.2. (truncated FAS, V-cycle for nonlinear problems)

```
Set $\varepsilon, \varepsilon_0$. Initialize $x^{(j)}$.
for $i = 1 : niter$
    $v_j = 0_{n_j \times 1}$
    $x^{(j)} := mgm(f_j, x^{(j)}, v_j, \varphi^{(j)}, \psi^{(j)})$
    test convergence
end
function $x^{(k)} = mgm(f_k, x^{(k)}, v_k, \varphi^{(k)}, \psi^{(k)})$
    if $k = 0$
    $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, v_k; x^{(k)}, \varepsilon_0, \nu_0)$        (coarsest grid solution)
else
    $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, v_k; x^{(k)}, \varepsilon, \nu 1)$        (pre-smoothing)
    $x^{(k-1)} = I_k^{k-1} x^{(k)}$        (solution restriction)
    $g_k = \nabla f_k(x^{(k)})$
    if $k = j$
        $f_k := \mathrm{trun}\, f_k$        (function truncation)
        $g_k := \mathrm{trun}\, g_k$        (gradient truncation)
    end
    $v_{k-1} = I_k^{k-1} v_k + (\nabla f_{k-1}(x^{(k-1)}) - I_k^{k-1} g_k)$        (correction r.h.s.)
    $f_{k-1} = I_k^{k-1} f_k I_{k-1}^k$        (symbolic coarse grid function definition)
    $\varphi^{(k-1)} = R_k^{\varphi}(\varphi^{(k)} - x^{(k)} + I_{k-1}^k x^{(k-1)})$        (coarse grid bounds)
    $\psi^{(k-1)} = R_k^{\psi}(\psi^{(k)} - x^{(k)} + I_{k-1}^k x^{(k-1)})$        (coarse grid bounds)
    $v^{(k-1)} = mgm(f_{k-1}, x^{(k-1)}, v_{k-1}, \varphi^{(k-1)}, \psi^{(k-1)})$        (coarse grid corr.)
    $x^{(k)} := x^{(k)} + I_{k-1}^k (v^{(k-1)} - x^{(k-1)})$        (solution update)
    $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, v_k; x^{(k)}, \varepsilon, \nu_2)$        (post-smoothing)
end
```

Note that the coarse grid function definition $f_{k-1} = I_k^{k-1} f_k I_{k-1}^k$ is only symbolic. As explained in the previous section, it must be defined specifically for each function f. Also, the restriction of the solution using operator I_k^{k-1} may not always lead to the best results and one may prefer, for instance, L_2 projection; see, e.g., [55].

It has been observed in [48, Ex. 7.3.1] that the truncated multigrid algorithm may, in the first iterations, be slower than other algorithms (see also Section 5.5.5). This difference may be significant if we are only interested in a low-accuracy solution of the problem. Therefore, in the next section we propose a "non-truncated" FAS algorithm for nonlinear problems of type (5.2). This algorithm is a minor generalization of the method proposed in the pioneering paper by Hackbusch and Mittelmann [59].

5.2.5 Full approximation scheme multigrid without truncation

We now present a Full Approximation Scheme (FAS) version of Algorithm 5.2 without truncation of the finite element basis. In order to guarantee convergence of the algorithm, we need to modify the definition of coarse grid constraints. As mentioned above, a similar algorithm has been introduced by Hackbusch and Mittelmann [59]. The difference is in the treatment of the constraints on the coarse levels; while Hackbusch and Mittelmann used active sets, we use the restriction operators defined below.

Let us modify the restriction operators $R_k^{\varphi}, R_k^{\psi}$ introduced in the previous section. The motivation for this is two-fold: the exact solution of the initial problem should be a fixed point of the algorithm and a feasible point should remain feasible after the correction step. For given $x^{(k)}$, φ, ψ and some $y \in \mathbb{R}^{n_k}$ and $i \in \mathcal{I}_{k-1}$ the operators are defined as follows

$$(\widetilde{R}_k^{\varphi} y)_i = \left\langle \begin{array}{l} (I_k^{k-1} x^{(k)})_i \textit{ if } \max\{(\varphi_j - x_j^{(k)}) \mid j \in \mathcal{I}_k \cap \operatorname{int}\operatorname{supp} \lambda_i^{(k-1)}\} = 0 \\ \min\{y_j \mid j \in \mathcal{I}_k \cap \operatorname{int}\operatorname{supp} \lambda_i^{(k-1)}\} \textit{ otherwise} \end{array} \right. \tag{5.7}$$

$$(\widetilde{R}_k^{\psi} y)_i = \left\langle \begin{array}{l} (I_k^{k-1} x^{(k)})_i \textit{ if } \min\{(\psi_j - x_j^{(k)}) \mid j \in \mathcal{I}_k \cap \operatorname{int}\operatorname{supp} \lambda_i^{(k-1)}\} = 0 \\ \max\{y_j \mid j \in \mathcal{I}_k \cap \operatorname{int}\operatorname{supp} \lambda_i^{(k-1)}\} \textit{ otherwise}\,. \end{array} \right. \tag{5.8}$$

Figure 5.1 illustrates how operator $(\widetilde{R}^{\varphi}_{k}y)_i$ works. It depicts a segment of a one-dimensional

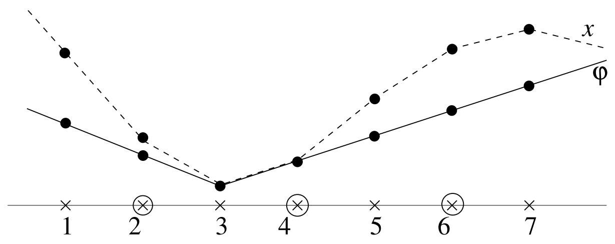


Figure 5.1: A segment of an obstacle and an approximate solution.

mesh with three coarse nodes (circles indexed $2, 4, 6$) and seven fine nodes (crosses indexed $1, \ldots, 7$). The constraints are active at nodes $3, 4$. For the coarse nodes 2 and 4, the first condition applies, as at least one of its neighbours is active. Hence $(\widetilde{R}^{\varphi}_{k}y)_2 = (I^{k-1}_{k}x^{(k)})_2$, $(\widetilde{R}^{\varphi}_{k}y)_4 = (I^{k-1}_{k}x^{(k)})_4$. For node 6 the second condition in the definition of $(\widetilde{R}^{\varphi}_{k}y)_i$ applies, so $(\widetilde{R}^{\varphi}_{k}y)_6 = \min\{y_5, y_6, y_7\}$. Notice that the operators will not be applied directly to functions φ, ψ, rather to their modifications; see the details of the FAS algorithm below.

This strategy of handling the coarse-level constraints is a combination of that of Kornhuber (as in Algorithm 5.1) and the active-set strategy by Hackbush and Mittelmann [59]. We are, however, slightly less conservative than [59]. In their algorithm, both nodes 2 and 4 would be considered active in the coarse-level problem and the corresponding value of x would not be allowed to change, unlike in the FAS algorithm below.

Algorithm 5.3. (FAS, V-cycle for nonlinear problems)

```
Set $\varepsilon, \varepsilon_0$. Initialize $x^{(j)}$.
for $i = 1 : niter$
    $q_j = 0_{n_j \times 1}$
    $x^{(j)} := mgm(j, x^{(j)}, q_j, \varphi^{(j)}, \psi^{(j)})$
    test convergence
end
```

```
function $x^{(k)} = mgm(k, x^{(k)}, q_k, \varphi^{(k)}, \psi^{(k)})$
    if $k = 0$
        $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, q_k; x^{(k)}, \varepsilon_0, \nu_0)$
    else
        $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, q_k; x^{(k)}, \varepsilon, \nu_1)$        (pre-smoothing)
        $x^{(k-1)} = I_k^{k-1} x^{(k)}$
        $q_{k-1} = I_k^{k-1}(q_k - \nabla f_k(x^{(k)})) + \nabla f_{k-1}(x^{(k-1)})$
        $\varphi^{(k-1)} = \widetilde{R}_k^{\varphi}(\varphi^{(k)} - x^{(k)} + I_{k-1}^k x^{(k-1)})$        (coarse grid bounds)
        $\psi^{(k-1)} = \widetilde{R}_k^{\psi}(\psi^{(k)} - x^{(k)} + I_{k-1}^k x^{(k-1)})$        (coarse grid bounds)
        $v^{(k-1)} = mgm(k-1, x^{(k-1)}, q_{k-1}, \varphi^{(k-1)}, \psi^{(k-1)})$
        $x^{(k)} := x^{(k)} + I_{k-1}^k (v^{(k-1)} - x^{(k-1)})$
        $x^{(k)} := opt(f_k, \varphi^{(k)}, \psi^{(k)}, q_k; x^{(k)}, \varepsilon, \nu_2)$        (post-smoothing)
    end
```

5.3 Equality constraints

As explained in the introduction, treating general (equality or inequality) constraints by multigrid may be difficult, if not impossible, as we may not be able to find the corresponding restriction operators. This, however, becomes easy in case of a single equality constraint involving all variables, such as

$$\sum_{i=1}^{n} x_i = \gamma \,. \tag{5.9}$$

Clearly, this constraint will be present in all discretization levels. We only have to guarantee that, having a feasible point with respect to (5.9) before the coarse-grid correction step, it will stay feasible after the correction. Let $\gamma_j = \gamma$ and assume that we are in the discretization level $k > 0$ and that

$$\sum_{i=1}^{n_k} x_i^{(k)} = \gamma_k\,, \tag{5.10}$$

where n_k is the number of variables on the k-th level. Then we define the coarser right-hand side for (5.9) as

$$\gamma_{k-1} = \sum_{i=1}^{n_{k-1}} (I_k^{k-1} x^{(k)})_i\,.$$

If $v^{(k-1)}$ is a solution of the $(k-1)$-level problem, then after the correction step $x^{(k)} := x^{(k)} + I_{k-1}^k (v^{(k-1)} - I_k^{k-1} x^{(k)})$, we obviously get again the equality (5.10).

Notice, however, that when we want to combine the equality constraint with the box constraints, we will have to use Algorithm 5.3 with the *untruncated* restriction operators $(\widetilde{R}_k^{\varphi} y)_i$ and $(\widetilde{R}_k^{\psi} y)_i$. The truncated multigrid in Algorithm 5.2 is not compatible with the equality constraint handled as above.

5.4 Smoothing by the steepest descent method

The choice of the smoothing method is vitally important for any multigrid algorithm. As our main aim is to avoid second derivatives of the function to be minimized, we have to resort to a first-order optimization method. Moreover, our choice of constrained convex minimization further narrows the choice of available algorithms. We have opted for the gradient projection method. Let us try to justify this choice in the next paragraphs.

5.4.1 Steepest descent smoother for unconstrained quadratic problems

Let us start with an unconstrained convex quadratic problem

$$\min_x \frac{1}{2} x^T Q x - q^T x \tag{5.11}$$

or, in other words, with a linear system

$$Qx = q \tag{5.12}$$

and the classic V-cycle multigrid algorithm. If things wouldn't work here, we can hardly expect them to work in the more general setting. One of the most popular smoothers in this case is the Gauss-Seidel (GS) iterative method. Rightly so, its very definition shows that it solves the equations locally, one by one, performing thus the local smoothing of the approximate solution of (5.12).

Looking at the optimization formulation (5.11) of the problem, we can also consider the "most basic" optimization algorithm, the steepest descent (SD) method with line search. Can this be a good smoother? This question was analyzed, e.g., by McCormick [79] who showed that the steepest descent method with exact or "slightly inexact" line search has indeed smoothing properties, as required for the convergence of the standard V-cycle. In case of exact line search, McCormick also gives an explicit bound on the convergence of the V-cycle using the steepest descent method as a smoother. This bound is, however, as many such theoretical bounds, overly pessimistic and far away from the real behaviour of the method (giving estimates of convergence speed such as 0.9995).

We have therefore performed a small experiment with the goal of testing the smoothing property of the steepest descent method and comparing it to the Gauss-Seidel method. Let us introduce some notation. Denote the exact solution on level h by $(x^*)^h$. Consider two discretizations of the underlying differential equation, one on the fine level parameterized by h and one on the coarse level $2h$. Let P be a prolongation operator from the coarse to the fine level. It is well known that the Q-orthogonal projector on the range of P can be written as

$$S^h = P(P^TQ^hP)^{-1}P^TQ$$

while

$$T^h = I - S^h$$

is the projector onto the Q-orthogonal complement of the range of P. With these two projectors, the energy norm of the error $e^h = (x^*)^h - x^h$ of an approximate solution x^h to (5.12) satisfies

$$\|e^h\|^2_{Q^h} = \|S^he^h\|^2_{Q^h} + \|T^he^h\|^2_{Q^h}$$

whereas S^he^h and T^he^h are the "low-frequency" and the "high-frequency" components of the error. It is the goal of the smoother to reduce T^he^h quickly in a few initial (possibly just 1–2) iterations.

In our experiment, the underlying problem was the Poisson problem on a unit square discretized by standard quadrilateral bilinear finite elements. We consider a 32 by 32 fine grid and 16 by 16 coarse grid; the condition number of the fine-grid matrix was 400. The restriction operator is the standard full weighting operator, see, e.g., [20]. We consider the steepest descent method $x^h_{k+1} = x^h_k - s_k(Q^hx^h_k - q^h)$ with exact step length s_k and a method with inexact line search, the details of which are given in the next section. We start the process with a randomly generated vector x^h_0 using the Matlab function `randn`. Assuming that the right-hand side, and thus the solution, is smooth, we will get significant components in both $S^he^h_0$ and $T^he^h_0$.

We first performed 10 iterations of SD and monitored the energy norm of both components of the error, $\|S^he^h\|^2_{Q^h}$ and $\|T^he^h\|^2_{Q^h}$. In Figure 5.2, these are depicted by the full line, the low frequency $\|S^he^h\|^2_{Q^h}$ in blue and the high-frequency $\|T^he^h\|^2_{Q^h}$ in red. Both values are given in logarithmic scale. The left-hand figure is for SD with exact line search while the right-hand side one is for the inexact line search. Also in Figure 5.2, we plot these values for the Gauss-Seidel method; these are denoted by the dashed lines. We can clearly see the smoothing effect

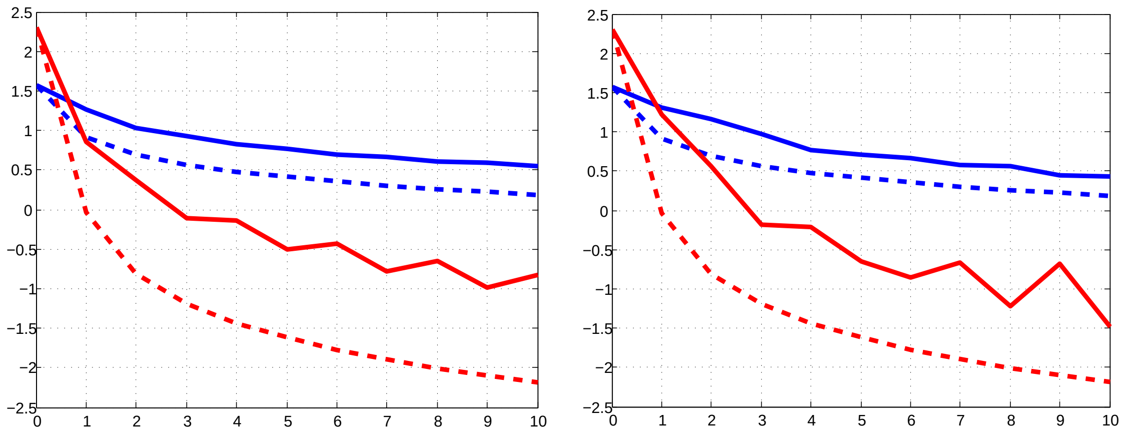


Figure 5.2: First 10 iterations of SD (full line) and GS (dashed line) methods. Blue line depicts low frequency error $\|S^h e^h\|^2_{A^h}$, red line the high frequency error $\|T^h e^h\|^2_{A^h}$. SD with exact line search is on the left, with inexact line search on the right.

of both methods in the first iterations when the red lines quickly drop by orders of magnitude. Although the steepest descent method is not as an efficient smoother as Gauss-Seidel, it still does a good job. After the initial iterations, the smoothing effect slows down and both errors descent proportionally. This can be better seen in Figure 5.3 where we show the error after 100 and after 1000 iterations of both methods. In this figure, we can clearly see the typical zig-zagging of the SD method with exact line search present in the high frequency error. We can also see that inexact line search with its random element breaks this regular zig-zagging and, in effect, makes the method significantly faster.

We could certainly consider other first-order method, for instance the nonlinear conjugate gradients. However, as we will see in the next section and later in the numerical experiments, only a very few (1–5) iterations of the steepest descent method suffice to guarantee a good behaviour of the multigrid algorithm, and there is thus no need for any more sophisticated first-order algorithms.

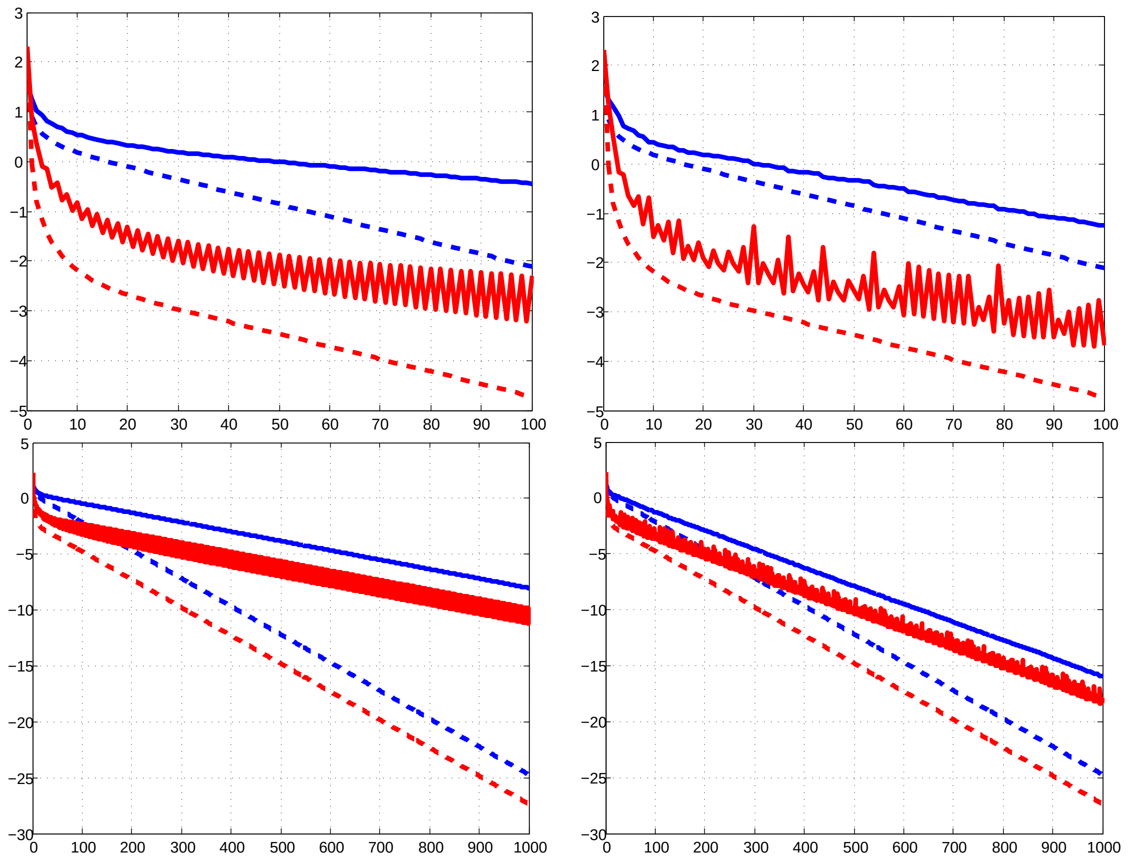


Figure 5.3: First 100 (top) and 1000 (bottom) iterations of SD (full line) and GS (dashed line) methods. Blue line depicts low frequency error $\|S^h e^h\|^2_{A^h}$, red line the high frequency error $\|T^h e^h\|^2_{A^h}$. SD with exact line search is on the left, with inexact line search on the right.

5.4.2 Line search

We have seen in the previous section that exact line search does not bring any significant benefit to the steepest descent method. Moreover, our ultimate goal is to use the projected gradient version of the method for bound-constrained nonlinear convex problems.

A popular—and efficient—choice of the step length is the Barzilai-Borwain method [28]. This, however, leads to a possibly non-monotonous progress of the error. Since we would like to use a very small *fixed* number of SD steps, this method is not suitable. Moreover, its projected gradient version is not fully understood and may lead to a standard Armijo step [28].

Because our problem is convex, the line search can be based solely on the gradient information.

For an unconstrained problem

$$\min_x f(x)$$

with f smooth and convex, we have opted for the following simplified version of Wolfe's method.

Algorithm 5.4. (Steepest descent method with gradient-based line search)

Given an approximate solution x, do until convergence:

```
$x_{\text{new}} = x - s\nabla f(x)$ with $s$ computed by Algorithm 5.5.
```

Algorithm 5.5. (Gradient-based line search for unconstrained problems)

Given an approximate solution x. Choose $s > 0$ and $c > 1$.

```
1. $\gamma = -(\nabla f(x))^T \nabla f(x - s\nabla f(x))$
2. if $\gamma < 0$
      2.1 do until $\gamma^+ > 0$
            2.1.1 $s := cs$
            2.1.2. $\gamma^+ = -(\nabla f(x))^T \nabla f(x - s\nabla f(x))$
      2.2. $s := \frac{1}{c}s$
   else
      2.3. do until $\gamma^+ < 0$
            2.3.1. $s := \frac{1}{c}s$
            2.3.2. $\gamma^+ = -(\nabla f(x))^T \nabla f(x - s\nabla f(x))$
   end
3. return current $s$
```

Clearly, γ (or γ^+) is the directional derivative of f in the steepest descent direction $-\nabla f(x)$ at the trial point $x - s\nabla f(x)$.

Lemma 5.4.1. *Algorithm 5.5 is well-defined and finishes in a finite number of steps. At the new point, we have* $f(x_{\text{new}}) < f(x)$ *and* $\gamma^+ < -(\nabla f(x))^T \nabla f(x_{\text{new}}) < 0$.

Proof. The claim follows immediately from the convexity of f. The algorithm stops with the value of s for which $\gamma^+ < 0$ and such that for the step length cs, the derivative would change sign to $\gamma^+ > 0$. The loop in 2.1 stops when we encounter a positive γ^+, so we have to return one step back in 2.2. $\square$

Let us now move to a generalization of Algorithm 5.4 for convex bound-constrained problem (5.1). For a given feasible x, let us recall the definition of the set of active indices by

$$\mathcal{A}(x) = \{i \mid x_i = \varphi_i \text{ or } x_i = \psi_i\} \ .$$

For this x, we further introduce an operator $[\,\cdot\,]_{\mathcal{A}(x)} : \mathbb{R}^n \to \mathbb{R}^n$ defined component-wise by

$$([\,h\,]_{\mathcal{A}(x)})_i = \left\langle \begin{array}{l} 0 \text{ if } i \in \mathcal{A}(x) \\ h_i \text{ otherwise.} \end{array} \right.$$

Finally, let $[\,\cdot\,]_\Omega$ denotes the (in this case trivial) projection on the feasible set.

Algorithm 5.6. (Gradient projection method with gradient-based line search)

Given a feasible approximate solution $x \in \Omega$, do until convergence

```
$x_{\text{new}} = [x - s\nabla f(x)]_\Omega$ with $s$ computed by Algorithm 5.7.
```

Algorithm 5.7. (Gradient-based line search for constrained problems)

Given a feasible approximate solution $x \in \Omega$. Choose $s > 0$ and $c > 1$.

```
1. $x^+ = [x - s\nabla f(x)]_\Omega, \quad \gamma = -(\nabla f(x))^T [\nabla f(x^+)]_{\mathcal{A}(x^+)}$

2. if $\gamma < 0$
```

```
        2.1 do until $\gamma^+ > 0$

            2.1.1 $s := cs$

            2.1.2. $x^+ = [x - s\nabla f(x)]_\Omega$

            2.1.3. $\gamma^+ = -(\nabla f(x))^T [\nabla f(x^+)]_{\mathcal{A}(x^+)}$

        2.2. $s := \frac{1}{c}s$

    else

        2.3. do until $\gamma^+ < 0$

            2.3.1. $s := \frac{1}{c}s$

            2.3.2. $x^+ = [x - s\nabla f(x)]_\Omega$

            2.3.3. $\gamma^+ = -(\nabla f(x))^T [\nabla f(x^+)]_{\mathcal{A}(x^+)}$

    end

3. return current $s$
```

Lemma 5.4.2. *Algorithm 5.7 is well-defined and finishes in a finite number of steps. At the new point, we have $f(x_{\text{new}}) < f(x)$ and $\gamma^+ < -(\nabla f(x))^T \nabla f(x_{\text{new}}) < 0$.*

Proof. For simplicity, we assume that $\gamma < 0$, so we are in the 2.1.–2.2. branch of the algorithm. The other case would be handled analogously. Assume first that no constraints are active at the initial point x, i.e., $\mathcal{A}(x) = \emptyset$. Once we computed a new point x^+ and found the active set $\mathcal{A}(x^+)$, we can split the search direction $-\nabla f(x)$ into two vectors:

$$-\nabla f(x) = g_A(x) + g_N(x)$$

where

$$(g_A(x))_i = \left\langle \begin{array}{ll} 0 & \text{if } i \in \mathcal{A}(x^+) \\ -(\nabla f(x))_i & \text{otherwise} \end{array} \right.$$

and complementary for $g_N(x)$. Both $g_A(x)$ and $g_N(x)$ are still descent directions, in particular

$$-(\nabla f(x))^T g_A(x) < 0\,.$$

Let

$$\widehat{x} := x - \widehat{s}\nabla f(x)$$

with $\widehat{s}$ chosen such that $\mathcal{A}(\widehat{x}) = \mathcal{A}(x^+)$ (see Figure 5.4, left). If $-(\nabla f(\widehat{x}))^T g_A(x) < 0$ then the function is still descending at $\widehat{x}$ in direction $g_A(x)$ and we do a line search in this direction, i.e., along the manifold defined by active indices. Due to convexity of f, we either have to reach a (finite) point at which the directional derivative changes its sign or when we hit a new constraint and the active set changes; in the latter case we repeat the above argument with the new active set. If $-(\nabla f(\widehat{x}))^T g_A(x) \geqslant 0$ then we know that we went too far, the algorithm stops and returns the previous trial point. The rest follows from Lemma 5.4.1.

Now if $\mathcal{A}(x) \neq \emptyset$ then either $\mathcal{A}(x^+) = \mathcal{A}(x)$ and the above argument applies or some constraints from $\mathcal{A}(x)$ are released at x^+ (Figure 5.4, right). But this would mean that the search direction goes away from these constraints, they can be ignored and, again, the above arguments apply.

In the (unlikely) case $\mathcal{A}(x^+) = \{1, 2, \ldots, n\}$ when all components of the trial point x^+ are active, we have $\gamma^+ = 0$ and the algorithm stops and returns the previous trial point. $\square$

Our preferred choice of the parameters is $c = 2$ and $s = 1$ in the first iteration. In the following iterations of the multigrid V-cycle, the parameter s is chosen as the final one from the previous call of gradient Algorithms 5.5 or 5.7.

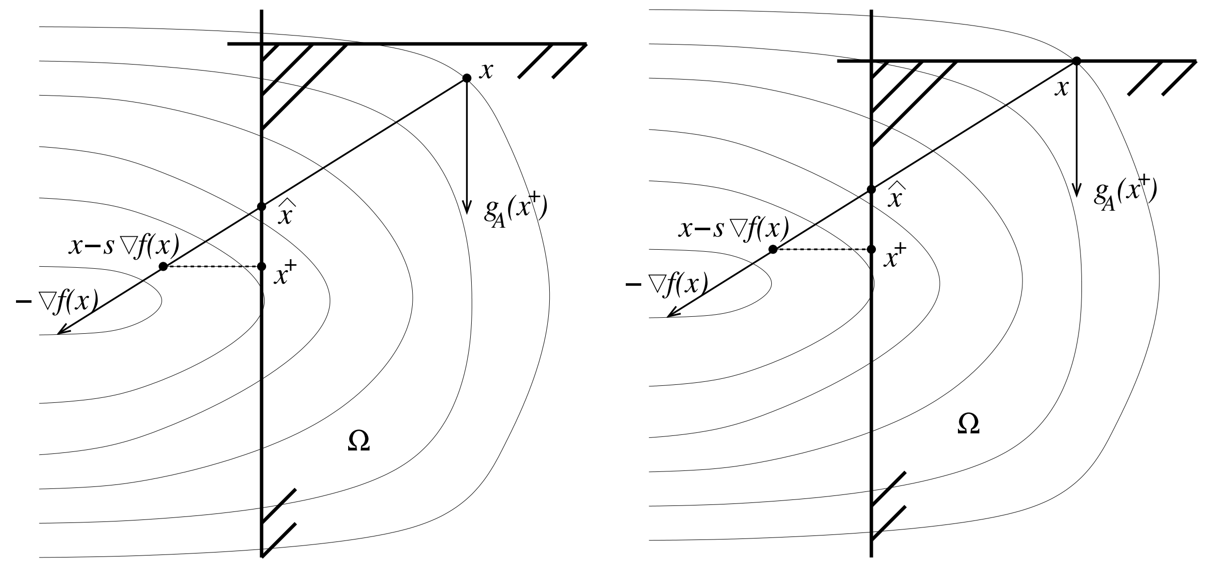


Figure 5.4: Gradient-based line search.

5.5 Numerical experiments

In this section we presents results of our numerical experiments using examples collected from the literature. We will start with a quadratic problem, in order to see the influence of non-linearity on the behaviour of the multigrid method.

For each example we will present the computed asymptotic rate of convergence for different numbers of the smoothing steps, namely for 2,4,6,8,10 smoothing steps (half of them in the pre-smoothing phase, half in the post-smoothing). In the same table, we will give the number of function and gradient calls on the finest level. We will also present a comparison (in the function/gradient calls) with one of the most efficient codes for these problems, the L-BFGS-B by Morales and Nocedal [81].

All examples are defined on the square $\Omega = (0, 1)^2$ in the infinite dimensional setting. We then use regular meshes of square finite elements with bilinear basis functions for their discretization. The initial coarsest mesh (refinement level 0) consists of four elements. We apply $j = 8$ uniform refinement steps to get 9 embedded finite element meshes. That means that, on refinement level k, $k = 0, 1, 2, \ldots$, we have 4^{k+1} finite elements and $(2^{k+1} - 1)^2$ interior nodes, with the finest mesh having 262 144 finite elements and 261 121 interior nodes. The prolongation operators I_{2h}^{h} are based on the nine-point interpolation scheme defined by the stencil $\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$. We use the full weighting restriction operators defined by $I_{h}^{2h} = \frac{1}{4}(I_{2h}^{h})^T$; see, e.g., [60]. The initial point in all experiments was set to a zero vector, even if this was infeasible. The problems on the coarsest level were solved by the same iterative method used as a smoother, however, with high accuracy. In particular, the parameters ε_0 and ν_0 in Algorithms 5.1-5.3 were set to $\varepsilon_0 = 10^{-9}$ and $\nu_0 = 10000$. Recall that ε_0 controls the norm of the scaled gradient or the KKT conditions for constrained problems and ν_0 is a bound on the number of iterations; because our coarsest problems have very low dimensions, this iteration bound was never reached.

The approximate asymptotic convergence rate is computed as $\left(\frac{e_k}{e_2}\right)^{\frac{1}{k-1}}$, where $e_k = \|x^* - x_k\|$, x_k is the last iteration before the algorithm stops and x^* is the "exact" solution as computed either by L-BFGS-B or by the gradient projection method with high accuracy. Every norm in the examples is the Euclidean norm. As a measure of efficiency of the algorithms we will present the number of function and gradient evaluations on the finest level. Of course the work on the lower levels is not free, even though the fine level is dominant, in particular for the nonlinear problems. For instance, in our implementation of the minimum surface problem (which can certainly be improved), the function and gradient evaluation on a fine level was 16 times more expensive than on the coarser level. In most examples, the number of function/gradient evaluations on a a single coarser level is about the same as on the finest level. In the most favourable case when the computational complexity of one function and gradient evaluation is linear in the number of variables, we would expect the effort to compute the function on the coarser level would be about one-quarter the effort on the finer level. This claim is confirmed by a remark concluding Example 5.5.2 presenting exact timings on all levels.

All algorithms were implemented in Matlab. The interface to the L-BFGS-B code was provided by Stephen Becker[I]. For all experiments we used a laptop with Intel Core i7-3570 CPU M 620 at 2.67GHz with 4GB RAM, and MATLAB version 8.0.0 (2012b) running in 64 bit Windows 7.

5.5.1 Example: quadratic obstacle problem

Let us start with the "Spiral problem" from [48]. This is a quadratic optimization problem resulting from the Laplace equation in $\Omega \subset \mathbb{R}^2$:

$$\min_{u \in H_0^1(\Omega)} \mathcal{J}(u) := \frac{1}{2}\int_\Omega \|\nabla u\|^2 \, dx - \int_\Omega F u \, dx$$

subject to

$$\varphi \leqslant u \leqslant \psi, \quad \text{a.e. in } \Omega\,,$$

with $F \in L^2(\Omega)$. We will use the spiral obstacle, as proposed in [48, §7.1.1]:

$$\varphi(x(r,\phi)) = \sin(2\pi/r + \pi/2 - \phi) + \frac{r(r+1)}{r-2} - 3r + 3.6, \quad r \neq 0\,,$$

and $\varphi(0) = 3.6$ with polar coordinates $x(r,\phi) = re^{i\phi}$. The upper bound function ψ is set to infinity and the right-hand side function F to zero. The obstacle function is illustrated in Figure 5.5 (left), together with the solution of the problem for 9 refinement levels.

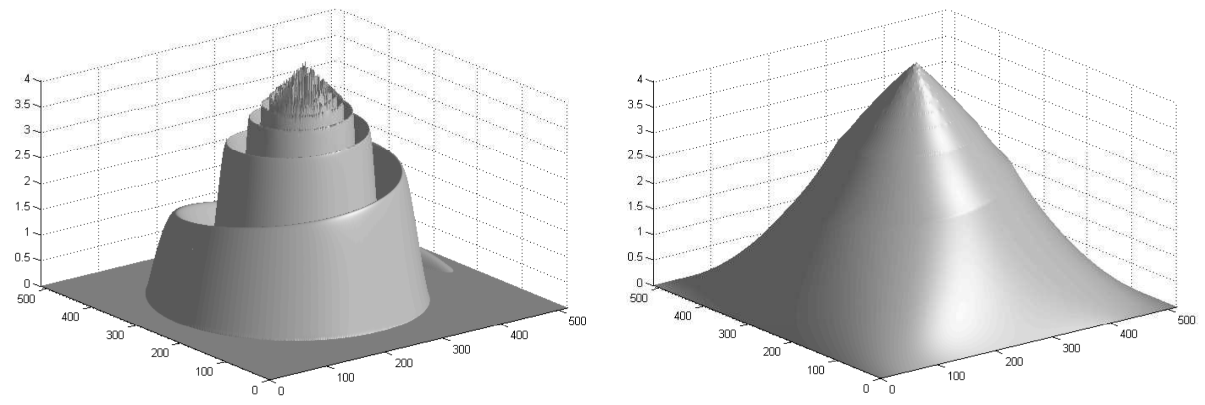


Figure 5.5: Example 5.5.1, nine refinement levels, obstacle (left) and solution (right).

[I]http://www.mathworks.co.uk/matlabcentral/fileexchange/35104-lbfgsb–l-bfgs-b–mex-wrapper

Table 5.1 presents the results of the numerical experiments. It shows the asymptotic rate of convergence (an average from the last 3–5 iterations) and the number of evaluations of the objective function and its gradient on the finest level only. The results are presented for 4–8 discretization levels and refer to Algorithm 5.1 with n pre-smoothing and n post-smoothing steps (GP-n). For comparison, we also show the convergence rate for the algorithm with the projected Gauss-Seidel (GSP) smoother (one pre- and one post-smoothing step). The last row of the table presents the numbers of function evaluations when we solved the finest level problem directly by the gradient projection method. The Gauss-Seidel smoother is clearly superior to GP-1 but its convergence rate can be easily reached by several GP smoothing steps. This increased number of GP steps is size-dependent, as indicated by the examples. However, the results also suggest that the increased number of GP steps is not needed, as the smallest number of function/gradient evaluations is obtained with 1 or 2 smoothing steps. Notice also that one GSP step is much more CPU expensive than one GP step, at least in Matlab implementation which allows for the vectorization of the GP step. The reason for this is that in GSP the projection must be performed for each variable separately, after its update by the Gauss-Seidel inner iteration, while in the GP the whole vector is projected at once.

Table 5.1: Example 5.5.1, asymptotic rate of convergence and number of top-level function evaluations for 4–8 refinement levels. GSP stands for the (1,1) V-cycle with Gauss-Seidel method with projection; GP-n for a (n, n) V-cycle with the GP smoother; "GP only" for gradient projection method solving the full problem on the finest level.

level (vars)	4 (961)		5 (3969)		6 (16129)		7 (65025)		8 (261121)	
smoother	rate	feval	rate	feval	rate	feval	rate	feval	rate	feval
GSP	0.07		0.14		0.22		0.32		0.37	
GP-1	0.18	71	0.33	107	0.50	180	0.80	410	0.86	711
GP-2	0.07	93	0.14	111	0.26	206	0.57	384	0.70	677
GP-3	0.03	92	0.08	142	0.17	239	0.35	387	0.55	806
GP-4	0.02	122	0.05	176	0.12	285	0.31	459	0.53	887
GP-5	0.01	160	0.03	211	0.08	306	0.23	488	0.34	912
GP only		685		2361		9320		34133		127289

The numbers from Table 5.1 are graphically presented in Figure 5.6: it shows the rate of con-

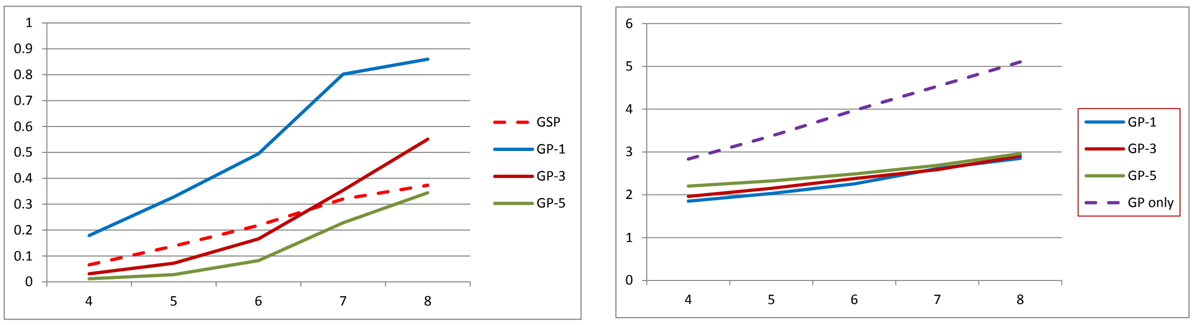


Figure 5.6: Example 5.5.1, rate of convergence (left) and function evaluations (right) for various smoothers as a function of the number of levels.

vergence for the different smoothing steps, as it increases with the number of levels (left-hand figure). The right-hand figure presents the logarithm of function evaluations as a function of the number of levels. We can observe a rapid increase for the "pure" gradient projection method and a much smaller increase for the multigrid algorithm, almost independent of the number of smoothing steps.

5.5.2 Example: non-quadratic obstacle problem

Consider the following optimization problem in $\Omega \subset \mathbb{R}^2$:

$$\min_{u \in H_0^1(\Omega)} \mathcal{J}(u) := \frac{1}{2} \int_\Omega \|\nabla u\|^2 - (ue^u - e^u)\, dx - \int_\Omega Fu\, dx$$

subject to

$$\varphi \leqslant u \leqslant \psi, \quad \text{a.e. in } \Omega\,,$$

with

$$\varphi(x_1, x_2) = -8(x_1 - 7/16)^2 - 8(x_2 - 7/16)^2 + 0.2, \qquad \psi = 0.5$$

and

$$F(x_1, x_2) = \Big(9\pi^2 + e^{(x_1^2 - x_1^3)\sin(3\pi x_2)}(x_1^2 - x_1^3) + 6x_1 - 2\Big)\sin(3\pi x_1)\,.$$

The unconstrained version of the problem is a nonlinear PDE studied in [20, p.105]. Figure 5.7 shows the solution of the unconstrained (top) and the constrained problem (bottom), both in two different views.

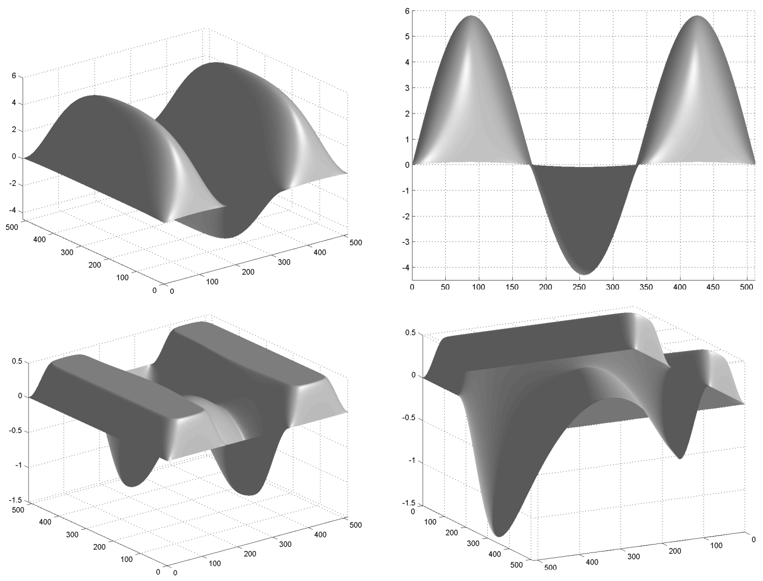


Figure 5.7: Example 5.5.2, solution of the unconstrained problem (top) and the constrained problem (bottom).

Just as in Example 5.5.1, Table 5.2 together with Figure 5.8 present the results of the numerical experiments using Algorithm 5.2. We do not show a comparison of the GP smoother with the projected nonlinear Gauss-Seidel smoother, used, e.g., in [59]. This is because the nonlinear GS algorithm needs the Hessian of the objective function and turns the algorithm into a second-order method. We can use a finite difference approximation of the Hessian but then the resulting code is extremely slow.

We do, however, compare the multigrid algorithm with one of the most efficient codes for bound-constrained nonlinear optimization, the L-BFGS-B code by Morales and Nocedal [81]. We can see that, with increasing size of the problem, the number of function evaluations grows faster in L-BFGS-B (though we should keep in mind that additional work needs to be done on coarse levels of the multigrid algorithm; see the beginning of this section). And we should keep in mind that, unlike in the multigrid algorithm, the function and gradient evaluation is not the only computationally extensive part of the L-BFGS-B code. We do not compare the CPU times, as L-BFGS-B is coded in Fortran.

Table 5.2: Example 5.5.2, asymptotic rate of convergence and number of top-level function evaluations for 4–8 refinement levels. GP-n stands for a (n, n) V-cycle; “GP only” and “L-BFGS-B” for gradient projection method and the L-BFGS-B method, respectively, solving the full problem on the finest level.

level (vars)	4 (961)		5 (3969)		6 (16129)		7 (65025)		8 (261121)	
smoother	rate	feval	rate	feval	rate	feval	rate	feval	rate	feval
GP-1	0.17	62	0.27	81	0.35	93	0.52	127	0.55	166
GP-2	0.12	131	0.21	193	0.29	192	0.42	282	0.50	321
GP-3	0.05	127	0.08	159	0.11	175	0.14	179	0.22	258
GP-4	0.05	171	0.07	205	0.09	249	0.14	284	0.29	384
GP-5	0.03	178	0.04	192	0.08	259	0.08	288	0.15	360
GP only		485		656		2128		5746		12197
L-BFGS-B		59		101		151		257		405

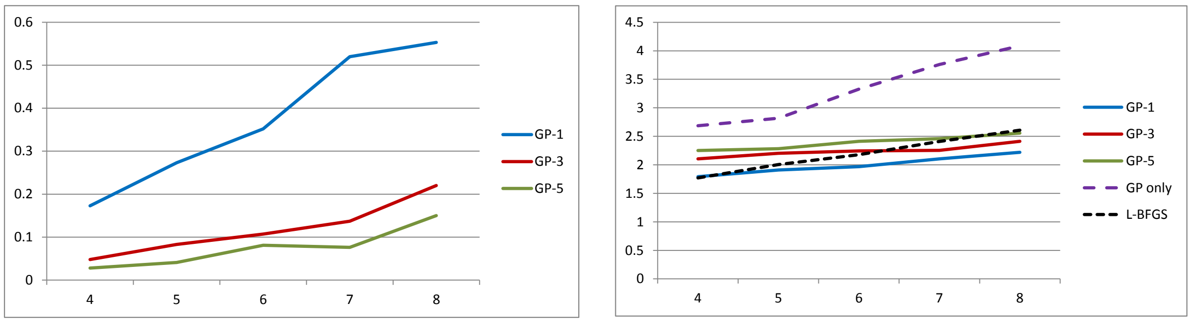


Figure 5.8: Example 5.5.2, rate of convergence (left) and function evaluations (right) for various smoothers as a function of the number of levels.

Remark 5.5.1. To have a better idea about the amount of work required on the coarser levels we present CPU times for the largest problem ($\ell = 9$) with GP-1. Using MATLAB’s tic-toc commands, we measured the cumulative times spent in the iterative method (the smoother) on every level; the times are given in seconds. The time spent on the finest mesh was 6.240, while the time spent on all other meshes was 1.824. The latter number is a sum of $(0.231, 0.036, 0.029, 0.031, 0.043, 0.108$ corresponding to the coarsest up to the second finest mesh, respectively. Recall that the problem on the coarsest mesh is solved to high accuracy.

5.5.3 Example: minimal surface problem

Our next example is the minimal surface problem

$$\min_{u \in H^1(\Omega)} \mathcal{J}(u) := \int_\Omega \sqrt{1 + \|\nabla u\|^2}\, dx$$

subject to

$$u(x_1, x_2) = u_\Gamma(x_1, x_2) \text{ for } (x_1, x_2) \in \partial\Omega$$

$$\varphi \leqslant u \leqslant \psi, \quad \text{a.e. in } \Omega\,,$$

with the boundary function (see [51])

$$u_\Gamma(x_1, 0) = \omega,\ u_\Gamma(1, x_2) = -\omega,\ u_\Gamma(x_1, 1) = -\omega,\ u_\Gamma(0, x_2) = \omega,\ \omega = -\sin(2\pi\xi)$$

and the parabolic lower bound

$$\varphi(x_1, x_2) = -8(x_1 - 0.5)^2 - 8(x_2 - 0.5)^2 + 0.55\,.$$

The upper bound function ψ is set to infinity. The solution is shown in two different views in Figure 5.9.

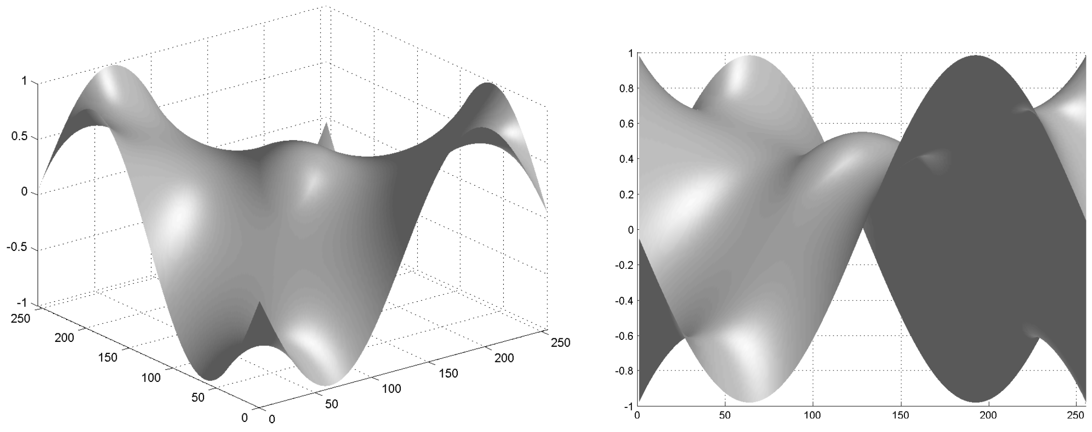


Figure 5.9: Example 5.5.3, solution.

As before, Table 5.3 together with Figure 5.10 present the results of the numerical experiments using Algorithm 5.2. Notice that the function and gradient evaluation for this problem is much more expensive than in the other examples and we were not able to obtain the exact solution to the finest problems in a reasonable time by L-BFGS-B. That is why we only present results for levels 3–7.

Table 5.3: Example 5.5.3, asymptotic rate of convergence and number of top-level function evaluations for 2–6 refinement levels. GP-n stands for a (n, n) V-cycle; "GP only" and "L-BFGS-B" for gradient projection method and the L-BFGS-B method, respectively, solving the full problem on the finest level.

level (vars)	2 (81)		3 (289)		4 (1089)		5 (4225)		6 (16641)	
smoother	rate	feval	rate	feval	rate	feval	rate	feval	rate	feval
GP-1	0.118	46	0.115	47	0.22	62	0.21	72	0.60	141
GP-2	0.079	125	0.055	157	0.13	176	0.18	111	0.43	171
GP-3	0.029	234	0.028	188	0.07	234	0.14	223	0.21	249
GP-4	0.011	156	0.021	205	0.05	165	0.06	170	0.19	276
GP-5	0.004	223	0.010	241	0.04	332	0.09	284	0.13	374
GP only		75		216		685		2361		9320
L-BFGS-B		14		30		126		156		242

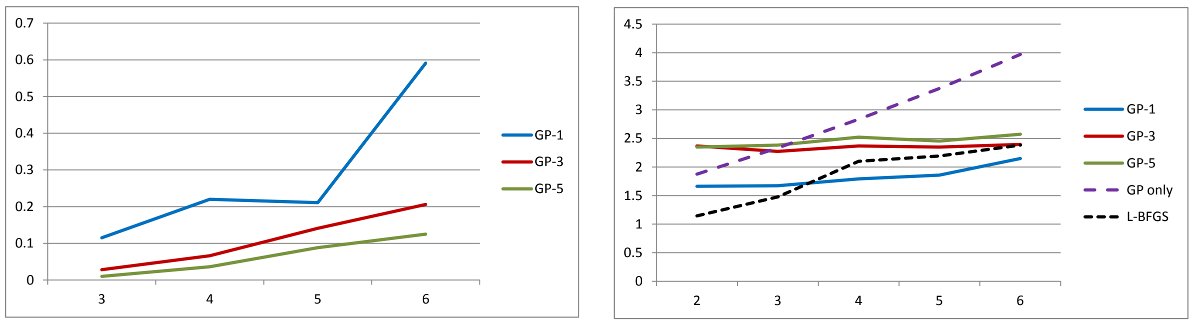


Figure 5.10: Example 5.5.3, rate of convergence (left) and function evaluations (right) for various smoothers as a function of the number of levels.

5.5.4 Example: obstacle problem with an equality constraint

Finally, let us consider an example with an obstacle and an additional equality constraint. The problem stems from the nonlinear PDE

$$\begin{aligned} -\triangle u - u^2 &= f(x) \quad && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned}$$

and can be formulated as the following optimization problem

$$\min_{u \in H_0^1(\Omega)} \mathcal{J}(u) := \frac{1}{2} \int_\Omega \left(\|\nabla u\|^2 - \frac{1}{3} u^3 \right) \, dx - \int_\Omega F u \, dx$$

subject to

$$\begin{aligned} &\varphi \leqslant u \quad \text{a.e. in } \Omega \\ &\int_\Omega u \, dx = 1 \,, \end{aligned}$$

with $F \equiv 0$ and

$$\varphi(x_1, x_2) = -32(x_1 - 0.5)^2 - 32(x_2 - 0.5)^2 + 2.5 \,.$$

Figure 5.11 (left) shows the solution and a comparison with the solution of the same problem without the equality constraint (right). In the unconstrained case, the optimal solution gives $\int_\Omega u \, dx = 0.62$. So, in order to satisfy the equality constraint, the unconstrained solution has been inflated.

Table 5.4 together with Figure 5.12 present the results of the numerical experiments using Algorithm 5.3 with the additional handling of the equality constraint (Section 5.3). The explanation is the same as in the previous examples.

Remark. Notice that, in the presence of the equality constraint, we can no longer use Algorithm 5.5 as a smoother, as the gradient-based line search would not lead to a convergent

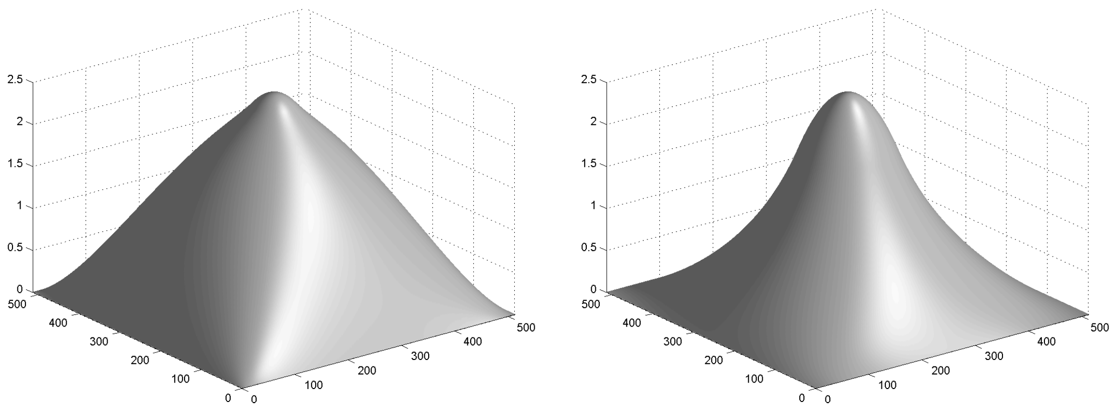


Figure 5.11: Example 5.5.4, solution with (left) and without (right) the equality constraint.

Table 5.4: Example 5.5.4, asymptotic rate of convergence and number of top-level function evaluations for 4–8 refinement levels. GP-n stands for a (n, n) V-cycle; and "GP only" for gradient projection method, solving the full problem on the finest level.

level (vars)	4 (961)		5 (3969)		6 (16129)		7 (65025)		8 (261121)	
smoother	rate	feval	rate	feval	rate	feval	rate	feval	rate	feval
GP-1	0.32	93	0.33	113	0.44	163	0.59	244	0.61	350
GP-2	0.11	88	0.25	120	0.29	129	0.51	183	0.54	182
GP-3	0.09	107	0.14	148	0.25	147	0.4	178	0.44	176
GP-4	0.07	153	0.14	182	0.23	186	0.36	224	0.44	191
GP-5	0.06	137	0.11	185	0.19	202	0.33	204	0.36	223
GP only		388		1586		5907		21372		75258

algorithm. Instead, we use a standard projected gradient method with backtracking Armijo line search. To find the projection on the feasible set, we now have to solve a convex quadratic programming problem. Moreover, this problem has to be solved to a high precision, because we need to identify the active constraints in (5.7)–(5.8). In our implementation, we have used the Gurobi solver to this purpose [56].

5.5.5 To truncate or not to truncate

The tables in [48] (and partly in the previous section) show the clear advantage of truncation: the higher asymptotic rate of convergence as compared to Algorithm 5.3 without truncation. However, a typical user may not be interested in asymptotic rate but in fast convergence in the first iterations. And here Algorithm 5.3 can be the winner. Figure 5.13 presents the convergence

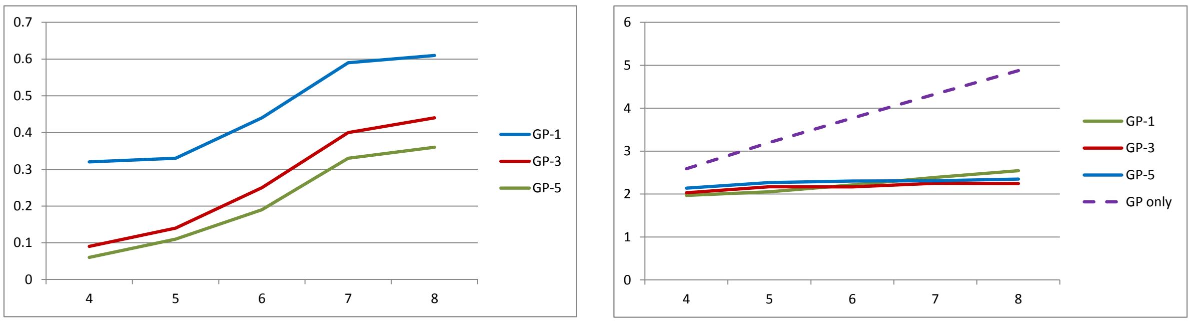


Figure 5.12: Example 5.5.4, rate of convergence (left) and function evaluations (right) for various smoothers as a function of the number of levels.

curves for Example 5.5.1 with 8 refinement levels and $\nu = 5$. The dashed line is for Algorithm 5.3 (no truncation) while the full line for Algorithm 5.2 (truncation). We can see a typical behaviour of the truncated algorithm: it starts slowly, tries to find the exact active set and, once this is found, the algorithm speeds up (for more details, see [48]). However, the total amount of work (represented in this case by the integral of the convergence curve) to reach the required accuracy is actually higher for the asymptotically slower algorithm without truncation.

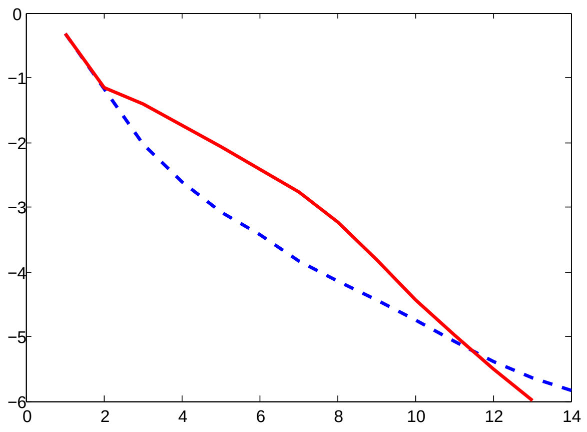


Figure 5.13: Example 5.5.1, nine refinement levels. Convergence curves (iterations vs logarithm of the error) for Algorithm 5.2 with truncation (full line) and Algorithm 5.3 without truncation (dashed line).

5.6 Conclusions

We have presented a version of the multigrid method for convex optimization problems with bound constraints and a possible single linear equality constraint. The method only needs gradient information, unlike similar published algorithms. We have shown that the projected gradient method can serve well as a smoother and that only a very small number of pre- and post-smoothing iterations is needed to obtain an efficient algorithm. The main advantage of the proposed method is thus in its low computational complexity and low memory requirements.

As an interesting by-product for unconstrained problems, we obtained a first-order method able to solve large scale problems efficiently and *to high accuracy*, which is rather untypical in today's realm of first-order methods designed to solve very large scale convex problems though only to some 2–3 digits of accuracy.

The natural question arises about more general constraints. The author devoted significant effort to the generalization of this method to the topology optimization problem (6.35), a convex problem with bound constraints and a single linear equality constraint; however, with two kinds of variables that need to discretized and prolong/restrict on different discretization levels leads to major technical difficulties (explained in more details in the section 7.2). So at the time of writing this approach, we think that more complex constraints could be better handled by traditional optimization methods (SQP or interior point) and multigrid could then be used for the solution of resulting linear systems.

Nevertheless, the class of unconstrained and bound-constrained convex optimization problems is very large and we believe that the presented method, whenever applicable, is one of the most efficient approaches to their solution.

CHAPTER 6

STRUCTURAL OPTIMIZATION PROBLEM

6.1 Linear elasticity

As we will describe in the next section, the goal of topology optimization is to determine the optimal layout of material in a given domain subject to prescribed loading conditions along with constraints on the amount of available material. Therefore, an understanding of elastic bodies subject to deformation and stress is essential in order to provide a suitable description of the problems that we look to solve. This section provides a number of definitions and theory from the field of linear elasticity, describing a linear relationship between stress and strain as a result of applied loads. We begin with describing a number of standard concepts as follows.

Definition 6.1. (Lipschitz Domain)

Assume $\Omega \subset \mathbb{R}^d$ is an open connected domain. A Lipschitz domain is a domain Ω with a Lipschitz boundary $\mathit{\Gamma}$. We say that Ω has a Lipschitz boundary $\mathit{\Gamma}$ if there exist nonnegative real numbers $\alpha, \beta > 0$, such that for each point x^0 on the boundary $\mathit{\Gamma}$, the Cartesian coordinate system can be rotated and translated to x^0, provided that the following statement is satisfied. Let

$$K_{n-1} = \{x \in \mathbb{R}^n | |x_i| < \alpha \text{ for } i = 1, 2, \dots, n-1\},$$

so that K_{n-1} represents an $(n-1)$-dimensional open cube. Then for each $(x_1, \dots, x_n) \in \mathit{\Gamma}$

there exists a function $a(x_1,\dots,x_{n-1}) \in K_{n-1}$ such that

$$a(x_1,\dots,x_{n-1}) = x_n,$$

is Lipschitz continuous. Furthermore, all points $x = (x_1,\dots,x_{n-1},x_n) \equiv (\acute{x}, x_n)$ where $\acute{x} \in K_{n-1}$ and $a(\acute{x}) < x_n < a(\acute{x})+\beta$ are assumed to lie inside Ω, and all the points $x = (\acute{x}, x_n)$, where $\acute{x} \in K_{n-1}$ and $a(\acute{x}) - \beta < x_n < a(\acute{x})$, are assumed to lie outside $\overline{\Omega} = \Omega \cup \mathit{\Gamma}$.

Definition 6.2. (Unit Outer Normal)

Let Ω a Lipschitz domain and let us use the same notations as in Def. 6.1. The vector $\mathbf{v} = (v_1,\dots,v_{n-1},v_n)$ whose components are given by

$$\begin{aligned} v_i &= \frac{1}{p}\frac{\partial a}{\partial x_i} \quad i = 1,\dots,n-1, \\ v_n &= -\frac{1}{p}, \end{aligned}$$

where

$$p = \left(1 + \sum_{i=1}^{n-1}\left(\frac{\partial a}{\partial x_i}\right)^2\right)^{1/2},$$

is called the *unit outer normal* to the boundary $\mathit{\Gamma}$.

Theorem 6.1. (Green Theorem)

Suppose that v_i are the components of the unit outer normal to the Lipschitz boundary $\mathit{\Gamma}$ of a domain Ω, then for $u \in C^1(\overline{\Omega})$ we have

$$\int_\Omega \frac{\partial u}{\partial x_i} = \int_{\mathit{\Gamma}} u v_i dS.$$

For the proof, we refer the interested reader to [84].

Definition 6.3. (Body Force f)

A body force f represents a force acting throughout the body. By writing $f := (f_1, ..., f_d)$, $d = 3$, with each $f_i : \Omega \to \mathbb{R}$ corresponding to the body force in each coordinate, f amounts to the density of forces acting on each volume element of the domain Ω. Examples of body force include the effects of gravitational pull, internal force and also magnetic force.

6.1.1 Stress tensor

Suppose that $\Omega_0 \subset \Omega$ such that $\overline{\Omega}_0 := \Omega_0 \cup \mathit{\Gamma}_0$, where $\mathit{\Gamma}_0$ represents Lipschitz boundary of Ω_0. The stress vector defines the density of the internal forces in the body affecting from the portion $\Omega - \Omega_0$ of the body on the portion $\overline{\Omega}_0$ at the point x, Fig.6.1. Furthermore, suppose that $\mathbf{v}$ is the unit outer normal to $\mathit{\Gamma}_0$ at a point $x \in \mathit{\Gamma}_0$. Then the stress vector depends on the point x and also the direction of the normal $\mathbf{v}$, which is denoted by $t(x, \mathbf{v})$ and represented in terms of the stress tensor τ and the normal vector $\mathbf{v}$.

$$t_i(x, \mathbf{v}) = \sum_{j=1}^{3} v_j \tau_{ji}(x), \quad i = 1, 2, 3, \tag{6.1}$$

where $\tau_{ji}(x)$ denotes a second order tensor at a point x. Therefore, the stress tensor relates the normal vector $\mathbf{v}$ to the stress vector t. In other words, it defines the state of stress at a point inside a material in the deformed placement or configuration. Furthermore, the tensor τ is symmetric, i.e. $\tau_{ji} = \tau_{ij}$ for $i, j = 1, 2, 3$, as shown in [84, p. 24].

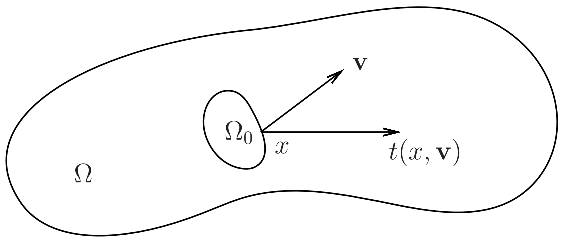


Figure 6.1: Illustration of stress vector

We proceed now to introduce another classical concept of elasticity known as the strain tensor.

6.1.2 Strain tensor

Informally, a local characterization of the displacement field is needed in order to gain insight into the resulting deformation inside the body as a result of prescribed loading conditions.

This can be obtained mathematically by computing the Jacobian matrix of the displacements. However, for engineering purposes, the deformation in the body is measured by the so called strain tensor and defined by

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \qquad i, j = 1, 2, 3, \tag{6.2}$$

which is referred as small strain tensor. The elements of e_{ij} represent the derivative of the displacements u. For the interested reader, the derivation of (6.2) can be found in [84, p. 29-30], where it is shown that the strain tensor amounts to the linear part of the matrix ε with entries defined as follows

$$\varepsilon_{ij} = \frac{1}{2}\left[\sum_{k=1}^{n} \frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j} + \left(\frac{\partial u_i}{\partial x_j}\frac{\partial u_j}{\partial x_i}\right)\right], i, j = 1, 2, 3.$$

This matrix is known as finite strain tensor.

We can follow the strain tensor definition by the generalized Hooke's law, also sometimes referred to as the elasticity law. This states that the stress is related to certain derivatives of displacements, where these derivatives are included in the strain tensor e.

Definition 6.4. (Generalized Hooke's Law) Hooke's Law is defined as a linear relation between the strain tensor e and the stress tensor τ at a point $x \in \Omega$. It expresses a mutually proportional relationship between e and τ and is written mathematically as

$$\tau(u) = E : e(u),$$

or

$$\tau_{ij}(u) = \sum_{k,l=1}^{3} E_{ijkl} e_{ij}(u).$$

This relation does not contain a constant term due to the assumption that the stress tensor disappears when the strain tensor is equal to zero. The symmetry of the tensors τ_{ij} and e_{ij} leads to the following conditions

$$E_{ijkl} = E_{jikl}, \quad E_{ijkl} = E_{ijlk},$$

and

$$E_{ijkl} = E_{klij}.$$

If the constants E_{ijkl} are independent of the choice of the coordinate system, the material is said to be isotropic, otherwise the term anisotropic is used. For this thesis we only consider the farmer case, namely isotropic methods for which we present the following theorem. In this case

$$E_{ijkl} := \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}), \quad 1 \leqslant i, j, k, l \leqslant 3.$$

As before, δ_{ij} represents the Kronecker delta and $\lambda, \mu > 0$ are the Lamé elasticity coefficients.

Theorem 6.2. For isotropic material the generalized Hooke's law is represented as

$$\tau_{ij}(x) = \lambda(x)\delta_{ij}\vartheta(x) + 2\mu(x)e_{ij}(x), \tag{6.3}$$

where

$$\vartheta(x) = \sum_{i=1}^{3} e_{ii}(x).$$

Proof. For the proof, see [84, p. 44]. □

In engineering, instead of Lamé coefficients λ and μ, Young's modulus Y and Poisson's ratio σ are more commonly used. They are derived in [84, pp. 49-50] as

$$\lambda = \frac{Y\sigma}{(1+\sigma)(1-2\sigma)}, \quad \mu = \frac{Y}{2(1+\sigma)}.$$

6.1.3 Equations of equilibrium

Linear elasticity under the conditions of equilibrium reads

$$\text{div}(\tau) + f = 0 \text{ in } \Omega,$$

in which the total forces must sum to zero over the whole domain. Namely, the conditions of equilibrium for the applied forces $f_i,\ i = 1, 2, 3$ on the sphere $B_h \subset \Omega$ of radius h centered at a point $x^0 \in \Omega$ gives

$$\int_{B_h} f_i dx + \int_{\Gamma} t_i(x, \mathbf{v}(x)) d\Gamma = 0, \tag{6.4}$$

where $\Gamma := \partial B_h$ (Lipschitz boundary of the sphere B_h) and we substitute for $t_i(x, \mathbf{v}(x))$ from (6.1). By applying Green's theorem (Theorem 6.1), we obtain that

$$\int_{B_h} f_i dx + \int_{B_h} \frac{\partial \tau_{ji}}{\partial x_j} dx = \int_{B_h} \left(f_i + \frac{\partial \tau_{ji}}{\partial x_j} \right) dx = 0. \tag{6.5}$$

The equation (6.5) is divided by the volume of the sphere B_h and takes the limit as $h \to 0$. On account of the continuity of the integrand, for any $x^0 \in \Omega$, this yields the so-called equations of equilibrium as follows:

$$\frac{\partial \tau_{ji}}{\partial x_j}(x^0) + f_i(x^0) = 0, \quad i = 1, 2, 3. \tag{6.6}$$

Equivalently,

$$\text{div}(\tau) = -f \text{ in } \Omega. \tag{6.7}$$

6.1.4 Lamé equations

We can obtain the Lamé equations directly from the equations of equilibrium (6.6), with $f_i \in C^1(\overline{\Omega})$, after substituting the generalized Hooke's law (6.3) for isotropic material and the strain tensor (6.2). Mathematically speaking, for $\lambda \in C^1(\overline{\Omega})$, $\mu \in C^1(\overline{\Omega})$ and the displacements $u_i \in C^2(\overline{\Omega})$, we obtain the following equations

$$\frac{\partial}{\partial x_j}(\lambda\vartheta\delta_{ij}) + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] + f_i = 0,$$

or

$$\frac{\partial}{\partial x_j}(\lambda\vartheta) + \frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_i}{\partial x_j}\right) + \frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_j}{\partial x_i}\right) + f_i = 0, \quad i = 1, 2, 3. \tag{6.8}$$

Equations (6.8) are known as the general Lamé equations. Moreover, if we assume that λ and μ are constants, then (6.8) maybe presented as

$$(\lambda + \mu)\frac{\partial\vartheta}{\partial x_i} + \mu\triangle u_i + f_i = 0, \quad i = 1, 2, 3, \tag{6.9}$$

where $\triangle u_i = \frac{\partial^2 u_i}{\partial x_j{}^2}$, $\vartheta = \frac{\partial u_j}{\partial x_j} = \text{div}(u)$, and $\frac{\partial\vartheta}{\partial x_i} = \sum_{j=1}^{3}\frac{\partial^2 u_j}{\partial x_i \partial x_j} = \bigtriangledown\text{div}(u)$.
The equations (6.9) hold for an isotropic material and are referred to as the Lamé equations. Equivalently, equation (6.9) can be written as follows

$$(\lambda + \mu)(\bigtriangledown\text{div}(u))^T + \mu\triangle u = -f \text{ in } \Omega. \tag{6.10}$$

6.1.5 The classic formulation of basic boundary value problems of elasticity

Consider the domain Ω with a continuously differentiable boundary $\mathit{\Gamma}$, with $\mu, \lambda \in C^1(\overline{\Omega})$ and $u_i \in C^2(\Omega)$. We can take the general Lamé equations (6.8) into account and find the displacement u that solves these equations with boundary conditions. There are three types of boundary value problem as follows:

(a) First Basic Boundary Value Problem

Let the boundary $\mathit{\Gamma}$ be subjected to the boundary forces $t := (t_1, t_2, t_3)$, $t_i \in C(\mathit{\Gamma})$, $i = 1, 2, 3$. Thus, we bring about the stress vector (6.1),

$$\tau_{ij}(x)v_j(x) = t_i(x), \quad x \in \mathit{\Gamma}.$$

Using Definition (6.4), we use Hooke‘s law for isotropic material in order to yield the first basic boundary equation

$$\lambda\vartheta v_i + 2\mu e_{ij}v_j = t_i \;\; x \in \mathit{\Gamma}, \tag{6.11}$$

which includes both the general Lamé equations (6.8) and also the boundary condition (6.11). Consequently, we look for $u_i \in C^1(\overline{\Omega}) \cap C^2(\Omega), \quad i = 1, 2, 3$, in which the equations (6.8) and (6.11) are satisfied.

(b) Second Basic Boundary Value Problem

In this case, if the prescribed displacements are located on the boundary $\mathit{\Gamma}$, then the boundary condition is written as follows:

$$u(x) = g(x), \quad x \in \mathit{\Gamma} \;\text{ and }\; g_i \in C(\mathit{\Gamma}). \tag{6.12}$$

Then, the aim is to find $u_i \in C^1(\overline{\Omega}) \cap C^2(\Omega), \quad i = 1, 2, 3$, that satisfy (6.8) and (6.12).

(c) Mixed Basic Boundary Value Problem

Let the boundary $\varGamma$ consist of two parts $\varGamma_D$ (Dirichlet boundary) and $\varGamma_N$ (Neumann boundary), or

$$\varGamma = \overline{\varGamma}_D \cup \overline{\varGamma}_N.$$

That is to obtain $u_i \in C^1(\Omega \cup \varGamma_D) \cap C(\Omega \cup \varGamma_N) \cap C^2(\Omega), i = 1, 2, 3$, which are on the Neumann boundary to satisfy (6.11) (Neumann condition), and on the Dirichlet boundary to satisfy (6.12) (Dirichlet condition), where $t_i \in C(\varGamma_N), g_i \in C(\varGamma_D), i = 1, 2, 3$.

Now we can introduce the classic formulation of the basic boundary value problem of elasticity (CFE)

Find $u \in C^1(\Omega)^d$ such that

$$\text{div}(\tau) = -f \quad \text{in} \quad \Omega, \qquad \text{equilibrium equation} \tag{6.13}$$

$$\tau.\mathbf{v} = t(x), \quad x \in \varGamma_N \qquad \text{Neumann boundary condition} \tag{6.14}$$

$$u(x) = g(x), \quad x \in \varGamma_D \quad \text{and} \quad g_i \in C(\varGamma) \qquad \text{Dirichlet boundary condition} \tag{6.15}$$

$$\tau = E : e(u) \qquad \text{Hook's law.} \tag{6.16}$$

We can substitute Hook's law in (6.13) and (6.14) to obtain a formulation in displacements only

(CFE): Find a displacement field $u \in C^1(\Omega)^d$ such that,

$$\begin{aligned} \text{div}(E : e(u)) &= -f \text{ in } \Omega \\ (E : e(u)).\mathbf{v} &= t(x) \quad x \in \varGamma_N, \\ u(x) &= g(x), \quad x \in \varGamma_D. \end{aligned} \tag{6.17}$$

6.1.6 Korn‘s Inequality

Korn‘s inequality is of fundamental importance when analyzing both linear and nonlinear elasticity problems, [84].

Rigid Body movement: In a non deformed body, in which $e_{ij} = 0$, a displacement represents a rigid body rotation and translation. This statement is clarified in the next theorem.

Theorem 6.3. Let $u \in [H^1(\Omega)]^3, \Omega \subset \mathbb{R}^3$. Then $e_{ij}(u) = 0; i, j = 1, 2$ if and only if $u = a + B \times x$. Here $a \in \mathbb{R}^2$ and B represents a rotation matrix.

Denote by

$$P = \{v \in [H^1(\Omega)]^3 \ \ |v = a + B \times x\}, \tag{6.18}$$

a finite dimensional and closed subset in $[H^1(\Omega^*)]^3$, $\Omega^* \subset \Omega$. Now, we can introduce the Korn‘s Inequality definition.

Theorem 6.4. Let $\mathcal{V}$ define a closed subspace of $[H^1(\Omega)]^3$ such that $[H_0^1(\Omega)]^3 \subset \mathcal{V} \subset [H^1(\Omega)]^3$. Let $P_{\mathcal{V}} = P \cap \mathcal{V}$ and let $Q_{\mathcal{V}}$ represent the orthogonal complement of $P_{\mathcal{V}}$ in $\mathcal{V}$. Then for any $v \in Q_{\mathcal{V}}$ the following Korn's inequality holds

$$\int_\Omega e_{ij}(v)e_{ij}(v)d\Omega \geqslant c\|v\|^2.$$

The proof is given in [84, p.79].

6.1.7 Variational formulation of the elasticity problem

A standard approach used when attempting to determine solutions to (6.17) is to consider discretization method based on an appropriately defined weak formulation. The linear forms within the resulting variational problem (as equation) require suitable function spaces in order to describe results with regard to existence and uniqueness of solutions. One of the most benefits of following this approach is that existence theorems for weak solutions will typically be valid under more realistic assumptions than associated theory related to the classical formulation. The suitable function spaces that we require weak solutions to belong to are Sobolev spaces $H^k(\Omega)$ as presented in Definition (2.14). we define solution and test spaces based on prescribed data for the Dirichlet boundary as follows

$$\mathcal{V} := \{u \in (H^1(\Omega))^d | u = g \text{ on } \mathit{\Gamma}_D\}, \quad d = 1, 2, 3, \tag{6.19}$$

$$\mathcal{V}_0 := \{v \in (H^1(\Omega))^d | v = 0 \text{ on } \mathit{\Gamma}_D\}, \quad d = 1, 2, 3, \tag{6.20}$$

Definition 6.5. A function $u \in \mathcal{V}$ is defined as a *weak solution* [84, pp. 87] of the model problem (6.17) of elasticity if

$$u - g \in \mathcal{V}_0 \tag{6.21}$$

$$\int_\Omega [e(u) : E : e(v)]dx = \int_\Omega f_i v_i dx + \int_{\mathit{\Gamma}_N} t_i v_i d\mathit{\Gamma} \text{ for all } v \in \mathcal{V}_0, \tag{6.22}$$

where g represents boundary data.

We shall explain the means of obtaining the weak solution of the boundary value problem of elasticity. Clearly, the condition (6.21) means that $u = g \in \mathit{\Gamma}_D$. By multiplying both sides of the equilibrium equation (6.13) by $v \in \mathcal{V}_0$ and integrating over Ω we have

$$\int_\Omega (\text{div}(\tau))v dx = -\int_\Omega f v dx. \tag{6.23}$$

By applying Green's first formula on the left hand side of (6.23) we arrive at

$$\int_\Omega (\text{div}(\tau))v dx = -\int_\Omega \tau : e(v) dx + \int_{\mathit{\Gamma}} (\tau.\mathrm{v}) v dS. \tag{6.24}$$

By substituting (6.24) in (6.23) we obtain:

$$-\int_{\Omega} \tau : e(v)dx + \int_{\mathit{\Gamma}} (\tau.\mathbf{v})vdS = -\int_{\Omega} fvdx. \tag{6.25}$$

If we are given homogeneous Dirichlet boundary conditions, i.e. $g = 0$ on $\mathit{\Gamma}_D$, we are led to the weak formulation of elasticity (WFE)

$$\int_{\Omega} [e(u) : E : e(v)]dx = \int_{\mathit{\Gamma}_N} tvdS + \int_{\Omega} fvdx \text{ for all } v \in \mathcal{V}_0. \tag{6.26}$$

We bring this into the generic form of a weak formulation, a variational formulation of the elasticity problem can be written in the following form:
Find a function $u \in \mathcal{V}_0$ such that

$$a_E(u, v) = (l, v) \text{ for all } v \in \mathcal{V}_0, \tag{6.27}$$

here $a_E(u, v)$ is the energy bilinear form $a_E(u, v) : \mathcal{V}_0 \times \mathcal{V}_0 \to \mathbb{R}$ for an arbitrary virtual displacement v defined as

$$\begin{aligned} a_E(u, v) &:= \int_{\Omega} e(u) : E : e(v)dx, \\ \text{with} \\ l(u) &:= \int_{\mathit{\Gamma}_N} tudS + \int_{\Omega} fudx, \end{aligned} \tag{6.28}$$

defining the load linear form.

6.1.8 Existence and uniqueness of solutions to variational formulation of the elasticity problem

One can show the existence and uniqueness of the weak solution to the variational formulation VFE of the elasticity problem. That is, under some assumptions, first on the material tensor

E; i.e. at each point E is positive semi definite and in L^∞. Next on continuous dependence of the solution on the given data (u, F, f) such that $u = 0$ on the boundary $\Gamma = \partial\Omega$. Then the following Theorem can be stated as

Theorem 6.5. Under the assumption we mentioned earlier, the VFE always has a solution $u \in \mathcal{V}_0$. Furthermore, this solution is unique.

Proof. See [84, p.91], Theorem 7.2.3. $\square$

6.2 Structural Optimization Problem

The aim of structural optimization is to provide a mathematical framework for engineering design problems, resulting in an optimal structure satisfying a number of given constraints. The focus of optimization for such problems involves minimization of so-called compliance, or (equivalently) maximization of stiffness. Depending on the focus of optimization, the design variables may represent material properties which, in certain cases, leads to areas of void within Ω (as is the case for topology optimization). The discipline of topology optimization offers challenging problems to researchers working in large scale numerical optimization. The results are essentially colors of pixels in a 2d or 3d “pictures”. Hence, in order to obtain high-quality results, i.e., fine pictures capturing all details, a very large number of variables is essential. For a general overview on the field of structural topology optimization, we refer to the survey article by Rozvany et al. [94], and the monograph by Bendsøe and Sigmund [9].

We begin by considering a bounded elastic domain $\Omega \in \mathbb{R}^d$ where $d \in \{2, 3\}$ with a Lipschitz boundary Γ. Recalling related principles from elasticity theory presented earlier in this Chapter, we denote by e, τ and E the small strain (6.2), the stress (6.4) and elasticity tensor, respectively. Using these notations, we consider a general formulation of the structural optimization problem to introduce various aspects of structural design problem, which reads as

$$\inf_{\rho,\alpha} \sup_{u \in \mathcal{V}} -\frac{1}{2} \int_{\Omega(\alpha)} \rho^p e(u) : E : e(u) dx + \int_{\mathit{\Gamma}_N} t.u dx, \tag{6.29}$$

where $\mathit{\Gamma}_N \subset \mathit{\Gamma}$ and $[H_0^1(\Omega)]^d \subset \mathcal{V} \subset [H^1(\Omega)]^d$.

We can obtain the so-called shape optimization problem for differing α and fixed remaining parameters p, E, and ρ. However, for fixed α, the structural optimization leads to different classes of topology optimization problems for selective choices of p, E and d.

- By setting $p \geqslant 1$ and letting E denote an elasticity tensor of isotropic material, we arrive at the so-called solid Isotropic Material with penalization (SIMP) [10], which nowadays is widely popular among engineers.

  In this problem, the parameter ρ plays the role of artificial density. With increasing p, the optimal density tend to either the upper bound (full material E) or to the lower bound (void). Thus, one can use this approach when trying to avoid areas of intermediate density, as well as the presence of anisotropic materials in the optimal structure.

- The variable thickness sheet (VTS) [10, 88] arises in the case where $p = 1$, E denotes an isotropic elasticity tensor, and $d = 2$, which can be viewed as a particular case of SIMP formulation presented above where, ρ corresponds to thickness of a two dimensional isotropic elastic body.

- The free material optimization (FMO) problem [10, 121] is obtained by by setting $p = 1$ and $E(x) \in S^+$, wher S^+ denotes the space of symmetric positive definite matrices of proper dimesion. The design variable is the elastic stiffness tensor E which is a function of the space variable x,for example see [11]. Originally the design variables in FMO are all elements of the elasticity tensor. The optimal E can be reconstructed from the optimal ρ and u.

Furthermore, there are two more approaches that do not fit into the above formulation. These are the topological derivative [103] and the level-set method [112].

In this work the concentration will be on the variable thickness sheet problem which, under the parameters as described, results in the following saddle point problem.

$$\begin{aligned}
&a(u,\rho) := \inf_{\rho\in\mathcal{H}} \sup_{u\in\mathcal{V}_0} -\frac{1}{2}\int_\Omega \rho e(u) : E : e(u)dx + \int_{\Gamma_N} t.udx, \\
&\mathcal{H} = \left\{\rho \in L^\infty(\Omega) \mid 0 \leqslant \underline{\rho} \leqslant \rho(x) \leqslant \overline{\rho} < \infty, \int_\Omega \rho(x)d\Omega \leqslant V\right\} \\
&\mathcal{V}_0 := \{v \in (H^1(\Omega))^2 | v = 0 \text{ on } \Gamma_N\}
\end{aligned} \tag{6.30}$$

6.2.1 Existence and Uniqueness of Variable Thickness Sheet

Proof of existence of solutions to the VTS problem was presented in 1970 by Céa and Malanowski [23]. In their work $a(u,\rho)$ corresponded to an elastic membrane with strictly positive lower thickness $\underline{\rho} > 0$. This is inferred from the property that the admissible thickness function ρ belongs to a bounded, closed and weak* compact set in $L^\infty(\Omega)$ and also the property that the compliance is a lower weak*-semi-continuous function, as it is a function of ρ given through the equilibrium equations [10, p.272].

We can illustrate existence of solutions to the formulation (6.30) by using the following theorem

Theorem 6.6. Problem (6.30) has an optimal solution $(\rho^*, u^*) \in L^\infty(\Omega) \times \mathcal{V}_0$

Proof. Any saddle point of (6.30) corresponds to a solution of (6.30). Such a saddle point exists according to [24] if we can guarantee that

1. The set $\mathcal{H}$ is convex and weak* compact.

2. $a(\rho, .)$ is convex and continuous on $\mathcal{V}_0$ for all $\rho \in \mathcal{H}$,

3. $a(., v)$ is concave and continuous on $L^\infty(\Omega)$-weak* for all $v \in \mathcal{V}_0$.

4. $a(\rho, v)$ is coercive on $\mathcal{V}_0$ for $\rho \in L^\infty(\Omega)$.

The first condition follows from the fact that the admissible thickness function ρ belongs to a closed and bounded and thus weak* compact set in L^∞ therefore satisfying (1). Conditions (2) and (3) hold trivially due to the fact that $a(\rho, .)$ is continuous and quadratic in terms of v, and $a(., v)$ is linear in ρ if v is constant, respectively. Finally, the proof of (4) is based on Korn‘s inequality (6.4), with the proof given in [84, p.85]. $\square$

It can be noticed that, for $\rho > 0$ almost everywhere in Ω, the minimum compliance problem has a unique solution $u \in \mathcal{V}_0$. However, for $\rho \in \mathcal{H}$ there might be several solutions or none.

6.2.2 Discretization of Variable Thickness Sheet Problem

We now give a brief introduction to the discretization of the problem. Assume that Ω consists of m disjoint simplices. By m we denote the number of finite elements and by n the number of nodes (vertices of the elements). The function $\rho(x)$ is assumed to be constant in each element and thus can be characterized by a vector $\rho = (\rho_1, ..., \rho_m)$ of its element values. We approximate the displacement vector $u(x)$ by a continuous function which is bi-linear (linear in each coordinate) on every element. Such a function is given as $u_h = \sum_{i=1}^n \phi_i u_i$ where u_i and ϕ_i denote the respective values of $u \in \mathbb{R}^{d.n}$ and ϕ at the i-th node. For the interested reader, further details maybe found in [26].
With the basis functions $\varphi_j, j = 1, ..., n_k$, we define (3×2) matrices corresponding to the linear strain tensor in $\mathbb{R}^2$, namely

$$\widehat{B}_j := \begin{bmatrix} \partial\varphi_j/\partial x & 0 \\ 0 & \partial\varphi_j/\partial y \\ \frac{1}{2}\partial\varphi_j/\partial y & \frac{1}{2}\varphi_j/\partial x \end{bmatrix}, \quad j \in \mathcal{D}_i, \tag{6.31}$$

where $\mathcal{D}_i$ denotes an index set of nodes belonging to the i-th finite element. By n_g we define the number of Gauss integration points in each element. We denote by $B_{i,k}$ the block matrix that is composed of (3×2) blocks $\widehat{B}_j$ at the j-th position $j \in \mathcal{D}_i$ (computed at the k-th integration point) and zero elsewhere. Consequently, the full dimension of $B_{i,k}$ is $(3 \times 2n)$.

The global stiffness matrix K is a linear combination of element stiffness matrices K_i defined as,

$$K(\rho) = \sum_{i=1}^{m} \rho_i K_i, \quad K_i = \sum_{k=1}^{n_g} B_{i,k}^T E B_{i,k}.$$

Based on this definition, the discretization of the problem (6.29) can be presented as

$$\inf_{\substack{0 \leqslant \rho_i \leqslant \overline{\rho}, \\ \sum_{i=1}^m \rho_i \leqslant V}} \sup_{u \in \mathbb{R}^{2n}} \Pi(\rho, u) := -\frac{1}{2} \sum_{i=1}^{m} \rho_i (K_i u, u) + (f, u). \tag{6.32}$$

The minimum compliance problem can also be formulated from the above problem as

$$\begin{aligned} &\min_{u,\rho} \frac{1}{2} f^T u \\ &\text{subject to} \\ &\qquad 0 \leqslant \rho_i \leqslant \overline{\rho}, \quad i = 1, ..., m \\ &\qquad \sum_{i=1}^{m} \rho_i \leqslant V \\ &\qquad K(\rho) u = f. \end{aligned} \tag{6.33}$$

This problem depends on the so called variational principle as presented in [67]. For a given ρ, the displacement u satisfying the equilibrium equation $K(\rho)u = f$ corresponds to a minimizer of $\Pi(\rho, u)$. The compliance $f^T u$ is finite if and only if $\Pi(\rho, .)$ is bounded above,when $f^T u = \max_u \Pi(\rho, u)$.

In the next section, we introduce theory asserting weak convergence for finite element solutions. We omit the presentation of the convergence analysis for the associated finite element discretization of variable thickness sheet problem since this is discussed and analyzed by Petersson in [89].

6.2.3 Weak Convergence Result

Following [42], the rather weak and general assumptions for the displacement approximation are stated as follows

(i) For any sequence $\{u_h\}$ such that $u_h \in \mathcal{V}_0^h$ and $u_h \to u$ weakly in $H^1(\Omega)$, it satisfies that $u \in \mathcal{V}_0$.

(ii) Let a set $\chi \subset \mathcal{V}_0$ exists such that $\overline{\chi} = \mathcal{V}_0$, $\overline{\chi}$ is the $H^1(\Omega)$ closure of χ, and there exists an interpolation operator $\pi_h : \chi \to \mathcal{V}_{0h}$ such that $\pi_h u \to u$ strongly in $H^1(\Omega)$ as $h \to 0^+$ $\forall u \in \chi$.

The assumptions (i) and (ii) are satisfied for almost all classic appropriate finite element interpolations [89]. Therefore it is expected that the finite element solutions hold for any element type, which has conformed in the displacement analysis problem, when it is integrated with constant thickness approximation for an element [89].

Theorem 6.7. Let $(u_h^*, \rho_h^*) \in \mathcal{V}_0^h \times \mathcal{H}^h$ as $h \to 0$ and suppose that (i) and (ii) are satisfied. Then there exists a subsequence, indicated again by (u_h^*, ρ_h^*) with elements $(u^*, \rho^*) \in \mathcal{V}_0 \times \mathcal{H}$ such that

$$u_h^* \to u^*, \text{ weakly in } H^1(\Omega),$$

and

$$\rho_h^* \to \rho^*, \text{ weakly}^* \text{ in } L^\infty(\Omega).$$

If $\rho_{low} > 0$ and any such limit pair belongs to $\mathcal{V}_0 \times \mathcal{H}$, then the convergence of the whole sequence $\{u_h^*\}$ holds strongly in $H^1(\Omega)$ to the unique $u^* \in \mathcal{V}_0$.

Furthermore, the weak* convergence of ρ_h^* to ρ^* in $L^\infty(\Omega)$ means that

$$\int f\rho_h^* \to \int f\rho^*, \forall f \in L^1(\Omega), \tag{6.34}$$

and the result of this could be a sequence of discrete thickness solutions that may exploit oscillatory behavior with an increasingly refined FE mesh , even when there is a unique and smooth exact optimal thickness. However, useful information can still be extracted from this result. In particular, if $\rho_{low} > 0$ then the rapid oscillations are restricted to appear only in the region where the optimality condition is active.

In [89], Petersson showed the existence of solutions of the discretized variable thickness sheet problem and extended the mathematical analysis of mixed FEM of Stokes flow problem to prove strong FE-convergence, existence and the convergence of the variable thickness sheet problem. Furthermore, with reference to [102], one can expect solutions to be mesh independent, without the necessity of introducing restriction methods (such as a perimeter control), in more general step SIMP setting of $p > 1$.

6.2.4 Minmax formulation of the variable thickness sheet problem

The minimum compliance problem presented in (6.33) is not only nonconvex in variables (u, ρ) due to the equilibrium equations presented in the third constraint but is also inherently large scale. Both of these concerns can be addressed by considering two approaches, both of which result in equivalent convex programming problems.

The first approach involves using the equilibrium equation to eliminate u by writing $u = A^{-1}(\rho)f$. Here we need to assume that $\underline{\rho} > 0$ to ensure that the stiffness matrix is nonsingular, which results in the following convex formulation in the variable ρ only

$$\begin{aligned} \min_{\rho} \quad & f^T A^{-1}(\rho) f \\ \text{subject to:} \quad & \sum_{i=1}^{m} \rho_i \leqslant 1, \\ & 0 < \underline{\rho} \leqslant \rho_i \leqslant \overline{\rho}. \end{aligned} \tag{6.35}$$

The convexity is a consequence of the positive definiteness of the matrix $A^{-1}(\rho)$ and the set of constraints for a suitable choice of lower bound $\underline{\rho}$ is nonempty, convex and compact, see [10, pp.272-274].

The second approach involves deriving the dual formulation of (6.35) in terms of the variable u only. This result in an unconstrained problem and includes the minimization of a nondifferentiable function $F(u, \centerdot)$. This function is defined based on a summation of terms, with each term determined based on the maximum of two convex quadratic functions. The resulting formulation reads as follows

$$\min_{u\in\mathbb{R}^n, \lambda\in\mathbb{R}} \left\{ F(u,\lambda) := \lambda V - f^T u + \sum_{e=1}^{N} \max\left\{ (\frac{1}{2}u^T A_e u - \lambda)\underline{\rho}, (\frac{1}{2}u^T A_e u - \lambda)\overline{\rho} \right\} \right\}. \tag{6.36}$$

which can be viewed in a similar manner to the solution of truss topology design problem [8]. Here, we note that the objective function $F(u, \lambda)$ is a nonsmooth convex function, with the relationship between the original problem (6.33) and (6.36), denoted respectively by (P) and (D), is given in the latter theorem [8].

In the case where only a zero lower bound is imposed on the densities (i.e. free upper bound), the resulting problem (denoted $(D)_s$) is simplified to

$$\min_{u\in\mathbb{R}^n} \max_{i=1,\ldots,m} \{\frac{V}{2} u^T A_i u - f^T u\}, \tag{6.37}$$

where each term $u^T A_i u$ corresponds the energy of the element number i.

Thus we can write (6.37) as the following quadratically constrained minimization problem:

$$\begin{aligned} &\min_{\alpha \in \mathcal{R}, u \in \mathcal{R}^n} \alpha - f^T u \\ &\text{subject to} \\ &\qquad \frac{V}{2} u^T A_i u - f^T u \geqslant 0 \;\; i = 1, ..., m, \end{aligned} \tag{6.38}$$

Theorem 6.8. The solution of the minimization problem (P) equals the minimization of (D) multiplied by -1, or

$$\min(P) = -\min(D).$$

The proof is omitted, but can be found in [8].

CHAPTER 7

PRIMAL-DUAL INTERIOR-POINT MULTIGRID METHOD FOR TOPOLOGY OPTIMIZATION

7.1 Introduction

In this chapter, an interior point method for the structural topology optimization is proposed. The linear systems arising in the method are solved by the conjugate gradient method preconditioned by geometric multigrid. The resulting method is then compared with the so-called optimality condition method, an established technique in topology optimization. This method is also equipped with the multigrid preconditioned conjugate gradient algorithm. We conclude that, for large scale problems, the interior point method with an inexact iterative linear solver is superior to any other variant studied in the Chapter.

In this chapter we only consider the discretized, finite dimensional topology optimization problem as it given in the previous chapter. For its derivation and for general introduction to topology optimization, see, e.g., [10].

We will consider the basic problem of topology optimization (6.33): minimization of compliance under equilibrium equation constraints and the most basic linear constraints on the design variables:

$$\min_{\rho\in\mathbb{R}^m,\, u\in\mathbb{R}^n} \frac{1}{2} f^T u \tag{7.1}$$

subject to

$$\begin{aligned} & K(\rho)u = f \\ & \sum_{i=1}^m \rho_i = V \\ & \rho_i \geqslant 0, \quad i = 1,\dots,m \\ & \rho_i \leqslant \overline{\rho}, \quad i = 1,\dots,m \end{aligned}$$

where $K(\rho) = \sum_{i=1}^m \rho_i K_i$, $K_i \in \mathbb{R}^{n\times n}$ and $f \in \mathbb{R}^n$. We assume that K_i are symmetric and positive semidefinite and that $\sum_{i=1}^m K_i$ is sparse and positive definite. We also assume that the data $V \in \mathbb{R}$ and $\overline{\rho} \in \mathbb{R}$ is chosen such that the problem is strictly feasible. For further reference, we will call the design variable ρ the *density*.

The most established and commonly used optimization methods to solve this problem are the Optimality Conditions (OC) method ([10, p.308]) and the Method of Moving Asymptotes (MMA) by Svanberg [106]. In both methods, the computational bottleneck consists of the solution of a large scale linear system with a sparse symmetric positive definite matrix (the equilibrium equation). This is traditionally used by a direct solver, such as the Cholesky decomposition. Recently, several authors proposed the use of iterative solvers, mostly preconditioned Krylov subspace solvers, such as Conjugate Gradients (CG), MINRES or GMRES. These have one big advantage which is specific for their use within optimization algorithms: in the early (or even not-so-early) stages of the optimization method, only a very low accuracy of the linear solver is needed. They also have one big disadvantage: in the late stages of the optimization method, the linear solvers become very ill-conditioned and thus a vanilla iterative method can come into extreme difficulties.

It is therefore essential to use a good preconditioner for the Krylov subspace method. The difficulty lies in the fact that as we approach the optimal solution of the topology optimization problem, the condition number of the stiffness matrices increases significantly. In fact, it is only controlled by an artificial lower bound on the variable—if this bound was zero, the stiffness matrix would be singular. Wang et al. [113] studied the dependence of the condition number on the variables and concluded that it is a combination of the ratio of maximum and minimum density and the conditioning of a corresponding problem with constant density. Consequently, they proposed a rescaling of the stiffness matrix combined with incomplete Cholesky preconditioner. The rescaling results in constant order of condition number during the optimization iterations. For large scale example still hundreds of MINRES iterations are needed and hence the authors use recycling of certain Krylov subspaces from previous iterations of the optimization method. Recently, Amir et al. [3] proposed a multigrid preconditioner for the systems resulting from OC or MMA methods and demonstrated that the resulting linear system solver keeps its efficiency also for rapidly varying coefficient of the underlying PDE, i.e., rapidly varying ρ in (7.1).

While OC and MMA methods are the most popular methods in topology optimization, they may not be the most efficient. The basic problem (7.1) is convex (more exactly, it is equivalent to a convex problem) and we may thus expect interior point methods to be highly efficient (see, e.g., [118]). Indeed, Jarre et al. [64] proposed an interior point method for the truss topology optimization problem that is equivalent to the discretized problem (7.1), with the exception that the stiffness matrix may be dense. They reported high efficiency of the method and ability to solve large scale problems; they also proved convergence of the proposed method. Maar and Schulz [76] studied interior point methods for problem (7.1) with sparse stiffness matrices and proposed to use a multigrid preconditioner for the GMRES method to solve the arising indefinite linear systems.

A new comprehensive numerical study of optimization methods for topology optimization can be found in [93]. The authors compare the efficiency of different methods, including general purpose optimization solvers such as SNOPT [41].

In this chapter we follow the path outlined by Jarre et al. [64] and by Maar and Schulz [76]. We use the same interior point method as in [64] and, unlike in [76], reduce the linear systems to obtain positive definite matrices. This allows us to use standard conjugate gradient method preconditioned by standard V-cycle multigrid. We further use the same linear solver in the OC method (in the same way suggested in [3]) to get a comparison with our interior point method. We will see that in both cases the inexact multigrid preconditioned CG method leads to a very efficient optimization solver. Most notably, in case of the interior point method we obtain an approximately constant number of CG iterations needed to solve the full problem which is independent of the size of the problem. In case of the OC method, the total number of OC iterations is increasing with the problem size; however, for a given problem size, the number of CG steps per one linear systems remains almost constant, and very low, in all OC iterations, notwithstanding the condition number of the stiffness matrix.

7.2 Can direct multigrid from Chapter 5 be used for topology optimization problem?

The natural question arises about using direct multigrid from Chapter 5 for more general problems, in particular, for the solution of the problem (7.1). The author devoted significant effort to the generalization of this method to handle this problem by reformulating the problem analogously to those problems solved in Chapter 5. As a convex problem in ρ only with a single equality constraint and bound constraints on ρ_i, see equation (6.35).

However, with two kinds of variables ρ and u that need to be discretized and prolonged/restricted on different discretization levels with two different prolongation/restriction operators this seems to be not an easy task. Namely, the variables u and ρ are defined, respectively, as node-based and element-based and it turns out that the interplay of these two kinds of discretization brings major technical difficulties, in particular, with upper and lower bounds on the thicknesses of elements. For instance, how to treat bounds on ρ, $\underline{\rho} \leqslant \rho_i \leqslant \overline{\rho}$ **?** If we have part of the fine mesh Figure 7.1, which contains variable on upper bound $\overline{\rho}$, on lower bound $\underline{\rho}$ and in between ρ_i, after restriction, we can take the average but then the prolonged correction will again be the average. But, it needed to keep the variables which are on the upper bounds (active constraints) to stay on these bounds after the correction step. The question is how to define the restriction/prolongation operator in this case? As it is, the correction would have to be zero. We tried to avoid this by applying slope constraints on the thickness variables, that is by enforcing pointwise bounds on the density slopes, for more details see Petersson et al. [90]. But then theses difficulties are just postponed to coarser meshes. Perhaps the remedy is to have the ρ variables associated with nodes not with elements and use linear basis functions for ρ and quadratic for displacements u. Then the restriction/prolongation operator will be defined for linear and quadratic elements.

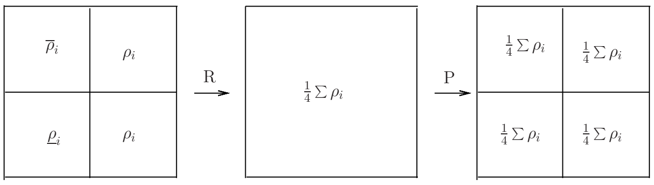

$\overline{\rho}_i$
ρ_i
$\underline{\rho}_i$
ρ_i
R
$\frac{1}{4}\Sigma\rho_i$
P
$\frac{1}{4}\Sigma\rho_i$
$\frac{1}{4}\Sigma\rho_i$
$\frac{1}{4}\Sigma\rho_i$
$\frac{1}{4}\Sigma\rho_i$

Figure 7.1: Prolongation (P) and restriction (R) for a part of a fine mesh

Further effort was made to generalize the proposed method to the nonsmooth unconstrained formulation of topology optimization. This can be obtained by deriving the dual formulation of (7.1) in terms of the variable u only, the resulting formulation reads as follows

$$\min_{u\in\mathbb{R}^n,\lambda\in\mathbb{R}}\left\{F(u,\lambda) := \lambda V - f^T u + \sum_{i=1}^{N}\max\left\{(\frac{1}{2}u^T A_i u - \lambda)\underline{\rho}, (\frac{1}{2}u^T A_i u - \lambda)\overline{\rho}\right\}\right\}, \quad (7.2)$$

and this analogous to the formulation of truss topology design problem [8]. For example, in the case where only a zero lower bound is imposed on the densities (i.e. free upper bound), the resulting problem is simplified to (see [8])

$$\min_{u\in\mathbb{R}^n}\max_{i=1,\dots,m}\{\frac{V}{2}u^T A_i u - f^T u\}, \quad (7.3)$$

where each term $u^T A_i u$ corresponds the energy of the element number i.

In literature, there are multigrid methods for nonsmooth unconstrained optimization problems (NSO), [49, 50]. However all these methods assume that the nonsmooth function is separable, in particular, a sum of one-dimensional nonsmooth functions (such as $\sum |x_i|$). This is not our case, as the function xA_ix in (7.3) is multivariate.

In these approaches, one needs to find a neighborhood of the given point in which the function is smooth (Figure 7.2), then the function can be transfered to smooth problem with bounds

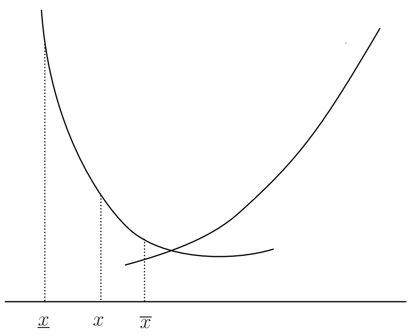


Figure 7.2: Finding bounds on x on a smooth branch of the function.

$(\underline{x}, \overline{x})$, however, this can be done efficiently for one dimensional functions. For the function (7.3) is as costly as solution of the full problem. Also, it seems that more general constraints may increase the complexity of the formulas for constraint restriction.

7.3 Newton systems for KKT conditions

Let $\mu \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$, $\varphi \in \mathbb{R}^m$ and $\psi \in \mathbb{R}^m$ denote the respective Lagrangian multipliers for constraints in (7.1). The Karusch-Kuhn-Tucker (KKT) first order optimality conditions for

(7.1) can be written as

$$
\begin{aligned}
-\mathrm{Res}^{(1)} &:= K(\rho)u - f = 0 && (7.4)\\
-\mathrm{Res}^{(2)} &:= \sum_{i=1}^{m} \rho_i - V = 0 && (7.5)\\
-\mathrm{Res}^{(3)} &:= -\frac{1}{2}u^T K_i u - \lambda - \varphi_i + \psi_i = 0, \quad i = 1,\dots,m && (7.6)\\
&\varphi_i \rho_i = 0, \quad i = 1,\dots,m && (7.7)\\
&\psi_i(\overline{\rho} - \rho_i) = 0, \quad i = 1,\dots,m && (7.8)\\
&\rho_i \geqslant 0, \quad \overline{\rho} - \rho_i \geqslant 0, \quad \varphi_i \geqslant 0, \quad \psi \geqslant 0 && (7.9)
\end{aligned}
$$

We will perturb the complementarity constraints (7.7) and (7.8) by barrier parameters $s_1, s_2 > 0$:

$$
\begin{aligned}
-\mathrm{Res}^{(4)} &:= \varphi_i \rho_i - s_1 = 0, \quad i = 1,\dots,m && (7.10)\\
-\mathrm{Res}^{(5)} &:= \psi_i(\overline{\rho} - \rho_i) - s_2 = 0, \quad i = 1,\dots,m && (7.11)
\end{aligned}
$$

and apply Newton's method to the system of nonlinear equations (7.4), (7.5), (7.6), (7.10), (7.11). In every step of the Newton method, we have to solve the linear system

$$
\begin{bmatrix}
K(x) & 0 & B(u) & 0 & 0\\
0 & 0 & e^T & 0 & 0\\
B(u)^T & e & 0 & I & -I\\
0 & 0 & \Phi & X & 0\\
0 & 0 & -\Psi & 0 & \widetilde{X}
\end{bmatrix}
\begin{bmatrix}
d_u\\ d_\lambda\\ d_\rho\\ d_\varphi\\ d_\psi
\end{bmatrix}
=
\begin{bmatrix}
\mathrm{Res}^{(1)}\\ \mathrm{Res}^{(2)}\\ \mathrm{Res}^{(3)}\\ \mathrm{Res}^{(4)}\\ \mathrm{Res}^{(5)}
\end{bmatrix}
. \tag{7.12}
$$

Here $B(u) = (K_1 u, K_2 u, \dots, K_m u)$, e is a vector of all ones and

$$
X = \mathrm{diag}(\rho), \quad \widetilde{X} = \mathrm{diag}(\overline{\rho} - \rho), \quad \Phi = \mathrm{diag}(\varphi), \quad \Psi = \mathrm{diag}(\psi)
$$

are diagonal matrices with the corresponding vectors on the diagonal.

Because the last two equations only involve diagonal matrices, we can eliminate d_φ and d_ψ:

$$d_\varphi = X^{-1}(\mathrm{Res}^{(4)} - \Phi d_t) \tag{7.13}$$

$$d_\psi = \widetilde{X}^{-1}(\mathrm{Res}^{(5)} - \Psi d_t)\,. \tag{7.14}$$

This will reduce the system (7.12) to

$$\begin{bmatrix} K(x) & 0 & B(u) \\ 0 & 0 & e^T \\ B(u)^T & e & -(X^{-1}\Phi + \widetilde{X}^{-1}\Psi) \end{bmatrix} \begin{bmatrix} d_u \\ d_\lambda \\ d_\rho \end{bmatrix} = \begin{bmatrix} \mathrm{Res}^{(1)} \\ \mathrm{Res}^{(2)} \\ \widetilde{\mathrm{Res}}^{(3)} \end{bmatrix} \tag{7.15}$$

with

$$\widetilde{\mathrm{Res}}^{(3)} = \mathrm{Res}^{(3)} - X^{-1}\mathrm{Res}^{(4)} + \widetilde{X}^{-1}\mathrm{Res}^{(5)}\,.$$

We can now follow two strategies. Firstly, we can solve the system (7.15) as it is, i.e., an indefinite system of dimension $m + n + 1$. To simplify things, we can still eliminate the multipliers φ and ψ as

$$\varphi_i = s_1/\rho_i, \quad \psi_i = s_2/(\overline{\rho} - \rho_i), \quad i = 1, \dots, m$$

to get

$$\begin{bmatrix} K(x) & 0 & B(u) \\ 0 & 0 & e^T \\ B(u)^T & e & -(s_1 X^{-2} + s_2 \widetilde{X}^{-2}) \end{bmatrix} \begin{bmatrix} d_u \\ d_\lambda \\ d_\rho \end{bmatrix} = \begin{bmatrix} \mathrm{Res}^{(1)} \\ \mathrm{Res}^{(2)} \\ \widetilde{\mathrm{Res}}^{(3)} \end{bmatrix}. \tag{7.16}$$

Remark. System (7.16) could be obtained directly as a Newton system for optimality conditions of the following "penalized" problem with penalty parameters s_1 and s_2:

$$\begin{aligned}\min_u \frac{1}{2} f^T u + s_1 \sum_{i=1}^m \log \rho_i + s_2 \sum_{i=1}^m \log(\overline{\rho} - \rho_i) \\ \text{s.t.} \quad K(\rho)u = f, \quad \sum_{i=1}^m \rho_i = V\,;\end{aligned}$$

see, e.g., [118, Ch.19.1].

Secondly, we can further reduce the Newton system (7.15). As the (3,3)-block matrix in (7.15) is diagonal, we will compute the Schur complement to the leading block to get

$$Z \begin{bmatrix} d_u \\ d_\lambda \end{bmatrix} = \mathrm{Res}^{(Z)}\,, \tag{7.17}$$

with

$$Z = \begin{bmatrix} K(x) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B(u) \\ e^T \end{bmatrix} (X^{-1}\Phi + \widetilde{X}^{-1}\Psi)^{-1} \begin{bmatrix} B(u)^T & e \end{bmatrix} \tag{7.18}$$

and

$$\mathrm{Res}^{(Z)} = \begin{bmatrix} \mathrm{Res}^{(1)} \\ \mathrm{Res}^{(2)} \end{bmatrix} + \begin{bmatrix} B(u) \\ e^T \end{bmatrix} (X^{-1}\Phi + \widetilde{X}^{-1}\Psi)^{-1} \widetilde{\mathrm{Res}}^{(3)}\,. \tag{7.19}$$

The remaining part of the solution, d_ρ, is then computed by

$$d_\rho = (X^{-1}\Phi + \widetilde{X}^{-1}\Psi)^{-1} \left(\widetilde{\mathrm{Res}}^{(3)} - B^T \mathrm{Res}^{(1)} - e\mathrm{Res}^{(2)} \right)\,. \tag{7.20}$$

7.4 Interior point method

Once we have derived the Newton systems, the interior point algorithm is straightforward (see, e.g., [118, Ch.19]). The details of the single steps of the algorithm will be given in subsequent paragraphs.

7.4.1 The algorithm

Denote $z = (u, \lambda, \rho, \varphi, \psi)^T$. Set $\rho_i = V/m,\ i = 1, \ldots, m,\ u = K(\rho)^{-1}f,\ \lambda = 1,\ \varphi = e,\ \psi = e$. Set $s_1 = 1,\ s_2 = 1,\ \sigma_{s_1}, \sigma_{s_2} \in (0, 1)$. Do until convergence:

1. Solve either system (7.15) or (7.17) and compute the remaining components of vector d from (7.12).

2. Find the step length α.

3. Update the solution

$$z = z + \alpha d\,.$$

4. If the stopping criterium for the Newton method is satisfied, update the barrier parameters

$$s_1 = \sigma_{s_1} \cdot s_1, \quad s_2 = \sigma_{s_2} \cdot s_2\,.$$

   Otherwise, keep current values of s_1 and s_2.

   Return to Step 1.

7.4.2 Barrier parameter update

We use a fixed update of both parameters s_1 and s_2 with

$$\sigma_{s_1} = \sigma_{s_2} = 0.2\,.$$

This update leads to long steps and, consequently, small number of interior point iterations. The value of the update parameter is a result of testing and leads, in average, to the smallest overall number of Newton steps. A more sophisticated version of the algorithm, with an adaptive choice of the barrier parameters s_1 and s_2 can be found in [64].

7.4.3 Step length

We cannot take the full Newton step

$$z_{\text{new}} = z + d$$

because some variables could become infeasible with respect to the inequality constraints (7.9). We thus need to shorten the step in order to stay strictly feasible with some "buffer" to the boundary of the feasible domain. A simple step-length procedure is described below (see also [118, Ch.19.2]).

Find α_l such that $\rho_i + (d_\rho)_i > 0$ for $i \in \{j : (d_\rho)_j < 0\}$ and α_u such that $\rho_i + (d_\rho)_i < \overline{\rho}$ for $i \in \{j : (d_\rho)_j > 0\}$ using the following formulas:

$$\alpha_l = 0.9 \cdot \min_{i:(d_\rho)_i<0} \left\{ -\frac{\rho_i}{(d_\rho)_i} \right\}, \quad \alpha_u = 0.9 \cdot \min_{i:(d_\rho)_i>0} \left\{ \frac{\overline{\rho} - \rho_i}{(d_\rho)_i} \right\}.$$

The constant 0.9 guarantees the shortening of the step in the interior of the feasible domain. Now take the smaller of these numbers and, in applicable, reduce it to 1:

$$\alpha = \min\{\alpha_l, \alpha_u, 1\}.$$

A more sophisticated (and complicated) line-search procedure is described in [64].

It is worth noticing that for a properly chosen initial barrier parameter and its update, the step-length reduction is almost never needed; this was, at least, the case of our numerical examples and our choice of the parameters.

7.4.4 Stopping rules

Following [64], we terminate the Newton method whenever

$$\frac{\|\mathrm{Res}^{(1)}\|}{\|f\|} + \frac{\|\widetilde{\mathrm{Res}}^{(3)}\|}{\|\varphi\| + \|\psi\|} \leqslant \tau_{\mathrm{NWT}}.$$

The full interior point method is stopped as soon as both parameters s_1 and s_2 are smaller than a prescribed tolerance:

$$\max\{s_1, s_2\} \leqslant \tau_{\mathrm{IP}}\,. \tag{7.21}$$

In our numerical experiments, we have used the values $\tau_{\mathrm{NWT}} = 10^{-1}$ and $\tau_{\mathrm{IP}} = 10^{-8}$.

Remark 7.4.1. A more established criterium for terminating the interior point algorithm would be to stop whenever all (scaled) residua are below some tolerance, i.e.,

$$\frac{\|\mathrm{Res}^{(1)}\|}{\|f\|} + \frac{\|\widetilde{\mathrm{Res}}^{(3)}\|}{\|\varphi\| + \|\psi\|} + \frac{\varphi^T\rho}{\|\varphi\|\|\rho\|} + \frac{\psi^T(\overline{\rho} - \rho)}{\|\varphi\|\|\rho\|} \leqslant \tau_{\mathrm{IP}}.$$

This criterium, however, leads to almost the same results as (7.21), hence we opted for the simpler and more predictable one.

Remark 7.4.2. The parameter τ_{NWT} is kept constant in our implementation, unlike in classic path-following methods. We will return to this point later in Section 7.10.1.

7.5 Optimality Conditions method

One of the goals of this chapter is to compare the interior point method with the established and commonly used Optimality Condition (OC) method. We will therefore briefly introduce the basic algorithm and its new variant. For more details about its derivation and for general introduction to OC, see [109, pp.57-61]. However, for an overview, we refer to ([10, p.308]) and the references therein.

7.5.1 OC algorithm

Assume for the moment that the bound constraints in (7.1) are not present. Then the KKT condition (7.6) would read as

$$-u^T K_i u + \lambda = 0\,, \quad i = 1, \ldots, m\,.$$

(For conveenience, we multiplied λ from (7.6) by $-\frac{1}{2}$.) Multiplying both sides by ρ_i, we get

$$\rho_i \lambda = \rho_i u^T K_i u\,, \quad i = 1, \ldots, m$$

which leads to the following iterative scheme:

$$\rho_i^{\mathrm{NEW}} = \frac{1}{\lambda}\, \rho_i u^T K_i u\,, \quad i = 1, \ldots, m\,.$$

The new value of ρ is then projected on the feasible set given by the bound constraints. The value of λ should be chosen such that $\sum_{i=1}^m \rho_i^{\mathrm{NEW}} = V$ and is obtained by a simple bisection algorithm. Hence we obtain the following algorithm called the OC method:

Algorithm OC Let $\rho \in \mathbb{R}^m$ be given such that $\sum_{i=1}^m \rho_i = V$, $\rho \geqslant 0$. Repeat until convergence:

1. $u = (K(\rho))^{-1} f$
2. $\overline{\lambda} = 10000$, $\underline{\lambda} = 0$
3. While $\overline{\lambda} - \underline{\lambda} > \tau_\lambda$
   (a) $\lambda = (\overline{\lambda} + \underline{\lambda})/2$
   (b) $\rho_i^{\mathrm{NEW}} = \min\left\{\rho_i \dfrac{u^T K_i u}{\lambda}, \overline{\rho}\right\}, \quad i = 1, \ldots, m$

(c) $\rho = \rho^{\mathrm{NEW}}$

(d) if $\sum_{i=1}^{m} \rho_i > V$ then set $\underline{\lambda} = \lambda$; else if $\sum_{i=1}^{m} \rho_i \leqslant V$ then set $\overline{\lambda} = \lambda$

The value of the bisection stopping criterium τ_λ has been set to 10^{-11}.

Notice that, due to positive semidefiniteness of K_i, the update in step 3(b) is always non-negative and thus the lower-bound constraint in the original problem (7.1) is automatically satisfied.

The basic version of the OC method converges (there are no known counter examples) but is extremely slow. The reason for this is that, from the very first iterations, the method is zig-zagging between two clusters of points. However, the following two modifications lead to a substantial improvement. To the best of our knowledge, the second modification called Averaged OC is new.

7.5.2 Damped OC

Algorithm DOC Let $\rho \in \mathbb{R}^m$ be given such that $\sum \rho_i = V$, $\rho \geqslant 0$. Repeat:

1. $u = (K(\rho))^{-1} f$
2. $\rho_i^{\mathrm{NEW}} = \min \left\{ \rho_i \dfrac{(u^T K_i u)^q}{\lambda}, \overline{\rho} \right\}, \quad i = 1, \ldots, m$
3. $\rho = \rho^{\mathrm{NEW}}$

Here q is called the damping parameter; the typical choice is $q = 1/2$. This version of the method is widely used among the structural engineers.

7.5.3 Averaged OC

Let us define an operator $OC(\cdot)$ as a result of one step of the standard OC algorithm.

Algorithm AOC Let $\rho \in \mathbb{R}^m$ be given such that $\sum \rho_i = V$, $\rho \geqslant 0$. Repeat:

1. $\rho^{(1)} = OC(\rho)$

2. $\rho^{(2)} = OC(\rho^{(1)})$

3. $\rho = \frac{1}{2}(\rho^{(1)} + \rho^{(2)})$

Numerical experiments suggest that Algorithm AOC is slightly faster than Algorithm DOC. This modification seems to be new, at least we did not find it in the existing literature.

7.6 Numerical experiments

7.6.1 Example shape1 (Figure 7.6)

Consider an example with 200 finite elements (size of the vector ρ) and 440 degrees of freedom (size of the vectors f and u). The exact solution (computed to a high precision by an interior point method) is denoted by ρ^*, the exact objective value by $c^* := f^T u$. Below we will show the behaviour of Algorithms OC, DOC and AOC.

The stopping criterion for all algorithms is based on the error in the objective function (compliance), the algorithms are stopped when $\frac{|c^* - c_k|}{|c^*|} < 1e - 9$ or when the number of iterations exceeds 1000. The other error that is computed is $\frac{\|\rho^* - \rho_k\|}{\|\rho^*\|}$.

The tables below show the iteration history. Here $\kappa_c = |c^* - c_k|/|c^* - c_{k-1}|$ and $\kappa_\rho = \|\rho^* - \rho_k\|/\|\rho^* - \rho_{k-1}\|$ are the "rates of convergence". In the following tables we use $t := \rho$.

7.6.1.1 Algorithm OC

```
Iteration   Compliance    |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
       1    253.47217433
       2    233.24746587   2.46e-001   0.6945   4.52e-001   0.9925
```

```
   3     230.52437432   2.31e-001   0.9408   3.64e-001   0.8055
   4     228.91736820   2.22e-001   0.9629   4.10e-001   1.1265
   5     228.02518620   2.18e-001   0.9786   3.62e-001   0.8831
   6     227.20043213   2.13e-001   0.9798   4.05e-001   1.1188
   7     226.56896077   2.10e-001   0.9842   3.56e-001   0.8789
   8     225.95733738   2.07e-001   0.9844   4.02e-001   1.1301
   9     225.43710253   2.04e-001   0.9866   3.51e-001   0.8717
  10     224.94610469   2.01e-001   0.9871   4.00e-001   1.1395
 ...
  19     221.75671967   1.84e-001   0.9912   3.30e-001   0.8492
  20     221.50875407   1.83e-001   0.9928   3.86e-001   1.1699
  21     221.22309611   1.81e-001   0.9917   3.27e-001   0.8458
 ...
  99     210.47702091   1.24e-001   0.9957   2.45e-001   0.7614
 100     210.40499587   1.24e-001   0.9969   3.20e-001   1.3086
 101     210.30553582   1.23e-001   0.9957   2.43e-001   0.7597
 ...
 199     204.52162766   9.22e-002   0.9972   1.91e-001   0.6924
 200     204.48849378   9.20e-002   0.9981   2.76e-001   1.4407
 201     204.44055943   9.17e-002   0.9972   1.90e-001   0.691
 ...
 499     197.86982900   5.67e-002   0.9983   1.31e-001   0.6241
 500     197.86053782   5.66e-002   0.9991   2.09e-001   1.5998
 501     197.84281336   5.65e-002   0.9983   1.30e-001   0.6241
 ...
 998     193.14548314   3.14e-002   0.9992   1.47e-001   1.4917
 999     193.13679828   3.14e-002   0.9985   9.83e-002   0.6697
1000     193.13208387   3.14e-002   0.9992   1.47e-001   1.4911
```

As we can see, the method is very slow even for this small example. The zig-zagging is nicely seen on the norm $\|t^* - t_k\|$. Moreover, although the method is monotone in the objective function, it is not monotone in the error of the variables.

7.6.1.2 Algorithm DOC

```
Iteration   Compliance    |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
     1      218.72897274
     2      201.87454284   7.80e-002   0.4644   4.04e-001   0.7230
     3      194.34554447   3.78e-002   0.4848   2.80e-001   0.6942
     4      190.76755024   1.87e-002   0.4950   1.90e-001   0.6790
     5      189.07387760   9.68e-003   0.5171   1.31e-001   0.6902
     6      188.27402055   5.41e-003   0.5589   9.46e-002   0.7194
     7      187.87401806   3.28e-003   0.6054   7.10e-002   0.7505
     8      187.65628900   2.11e-003   0.6452   5.51e-002   0.7767
     9      187.52944308   1.44e-003   0.6796   4.40e-002   0.7987
    10      187.45201932   1.02e-003   0.7122   3.60e-002   0.8184
   ...
    20      187.29101325   1.64e-004   0.8859   1.06e-002   0.9133
    40      187.26488433   2.40e-005   0.9167   2.25e-003   0.9290
    60      187.26120697   4.39e-006   0.9195   5.06e-004   0.9270
    80      187.26053907   8.27e-007   0.9201   1.08e-004   0.9245
   100      187.26041363   1.57e-007   0.9203   2.20e-005   0.9229
   120      187.26038985   2.98e-008   0.9204   4.37e-006   0.9219
   140      187.26038532   5.67e-009   0.9203   8.54e-007   0.9213
   160      187.26038446   1.08e-009   0.9201   1.65e-007   0.9210
   161      187.26038445   9.90e-010   0.9201   1.52e-007   0.9210
```

A signifigant change, the dampened version of the method converges to a very high accuracy solution in just 161 iterations. Monotonic behaviour of both errors.

7.6.1.3 Algorithm AOC

```
Iteration   Compliance    |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
      1     198.81943433
      2     188.24949007   1.88e-002   0.0856   2.29e-001   0.2798
      3     187.48380356   4.25e-003   0.2259   9.82e-002   0.4286
      4     187.35659827   1.83e-003   0.4306   5.83e-002   0.5942
      5     187.31813681   1.10e-003   0.6003   4.13e-002   0.7083
      6     187.29973325   7.48e-004   0.6813   3.14e-002   0.7600
      7     187.28861782   5.36e-004   0.7175   2.45e-002   0.7819
      8     187.28119974   3.96e-004   0.7373   1.95e-002   0.7928
      9     187.27598748   2.96e-004   0.7496   1.55e-002   0.7989
     10     187.27220953   2.25e-004   0.7579   1.25e-002   0.8021
    ...
     20     187.26129840   1.74e-005   0.7796   1.37e-003   0.7974
     30     187.26046171   1.47e-006   0.7820   1.33e-004   0.7889
     40     187.26039090   1.26e-007   0.7823   1.21e-005   0.7851
     50     187.26038483   1.08e-008   0.7822   1.06e-006   0.7835
     60     187.26038431   9.18e-010   0.7797   9.22e-008   0.7832
```

Even faster convergence of the Averaged OC method (we should keep in mind that one AOC iteration involves two basic OC steps, so 161 iterations of DOC should be compared to 110 OC steps in AOC).

7.6.2 Example shape2 (Figure 7.6)

Consider an example with 3200 finite elements (size of the vector t) and 6560 degrees of freedom (size of the vectors f and u).

Below we use the same notation as in the previous example.

7.6.2.1 Algorithm OC

```
Iteration   Compliance    |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
      1     76.31086103
      2     68.80078947   3.07e-001    0.6829   7.27e-001   0.9298
      3     67.44530725   2.82e-001    0.9162   5.64e-001   0.7751
      4     66.20329382   2.58e-001    0.9162   6.10e-001   1.0815
      5     65.79580256   2.50e-001    0.9700   4.90e-001   0.8032
      6     65.33836762   2.42e-001    0.9653   5.43e-001   1.1090
      7     65.12840538   2.38e-001    0.9835   4.47e-001   0.8234
      8     64.86298014   2.32e-001    0.9788   4.99e-001   1.1156
      9     64.72435435   2.30e-001    0.9887   4.15e-001   0.8325
     10     64.54082056   2.26e-001    0.9848   4.67e-001   1.1232
    ...
     19     63.80510518   2.12e-001    0.9961   3.64e-001   0.9091
     20     63.74749223   2.11e-001    0.9948   3.94e-001   1.0804
     21     63.70923353   2.11e-001    0.9966   3.63e-001   0.9228
    ...
     99     62.44074286   1.86e-001    0.9989   3.45e-001   0.9189
    100     62.43255097   1.86e-001    0.9992   3.75e-001   1.0876
    101     62.42162194   1.86e-001    0.9989   3.44e-001   0.9184
    ...
    199     61.66464903   1.72e-001    0.9991   3.29e-001   0.9049
    200     61.65937020   1.72e-001    0.9994   3.63e-001   1.1044
    201     61.65169202   1.71e-001    0.9991   3.29e-001   0.9047
    ...
    499     60.21293724   1.44e-001    0.9994   2.96e-001   0.8856
    500     60.20985774   1.44e-001    0.9996   3.34e-001   1.1286
    501     60.20541185   1.44e-001    0.9994   2.96e-001   0.8855
    ...
```

```
     998     58.76061052   1.17e-001    0.9997   2.99e-001   1.1643
     999     58.75793470   1.16e-001    0.9996   2.57e-001   0.8584
    1000     58.75603704   1.16e-001    0.9997   2.99e-001   1.1645
```

Even slower convergence than for the small example, the same bad behaviour.

7.6.2.2 Algorithm DOC

```
Iteration    Compliance     |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
       1     65.68832914
       2     60.22364871   1.44e-001    0.5816   7.70e-001   0.9231
       3     57.57783205   9.41e-002    0.6517   7.09e-001   0.9211
       4     56.09489206   6.59e-002    0.7004   6.52e-001   0.9195
       5     55.18398385   4.86e-002    0.7373   6.00e-001   0.9200
       6     54.58807732   3.73e-002    0.7669   5.54e-001   0.9230
       7     54.18045408   2.95e-002    0.7921   5.14e-001   0.9280
       8     53.89218199   2.40e-002    0.8144   4.80e-001   0.9337
       9     53.68246117   2.00e-002    0.8342   4.51e-001   0.9389
      10     53.52553338   1.71e-002    0.8512   4.25e-001   0.9430
     ...
      20     52.92452056   5.64e-003    0.9123   2.52e-001   0.9499
      30     52.75383695   2.40e-003    0.9203   1.50e-001   0.9494
      40     52.68275031   1.05e-003    0.9207   9.09e-002   0.9531
      50     52.65255899   4.74e-004    0.9277   5.93e-002   0.9629
     ...
     100     52.63009636   4.69e-005    0.9668   2.08e-002   0.9847
     200     52.62775371   2.43e-006    0.9744   5.14e-003   0.9873
     300     52.62763961   2.65e-007    0.9815   1.65e-003   0.9901
     400     52.62762841   5.25e-008    0.9856   6.91e-004   0.9924
     500     52.62762634   1.33e-008    0.9867   3.44e-004   0.9935
     600     52.62762583   3.53e-009    0.9868   1.82e-004   0.9938
```

```
...
695      52.62762570   9.92e-010    0.9863   1.01e-004   0.9939
```

Again, a much better behaviour. However, the rate of convergence apparently grows with the number of variables. Note, however, that DOC gets in just three iterations a better solution than OC in 1000 iterations. Furthermore, a useful solution is obtained in just about 50 iterations.

7.6.2.3 Algorithm AOC

```
Iteration    Compliance     |c* - c_k|  kappa_c ||t* - t_k|| kappa_t
       1     59.02479757
       2     54.60065521   3.75e-002    0.3084   5.51e-001   0.7629
       3     53.64328387   1.93e-002    0.5148   4.44e-001   0.8047
       4     53.27418237   1.23e-002    0.6366   3.74e-001   0.8421
       5     53.08421638   8.68e-003    0.7062   3.19e-001   0.8537
       6     52.96575399   6.42e-003    0.7406   2.73e-001   0.8558
       7     52.88421044   4.88e-003    0.7588   2.34e-001   0.8556
       8     52.82502057   3.75e-003    0.7693   2.00e-001   0.8548
       9     52.78064600   2.91e-003    0.7752   1.71e-001   0.8543
      10     52.74665132   2.26e-003    0.7778   1.46e-001   0.8543
    ...
      20     52.63940359   2.24e-004    0.8324   4.15e-002   0.9204
      30     52.63085101   6.13e-005    0.8985   2.34e-002   0.9527
      40     52.62881436   2.26e-005    0.9087   1.48e-002   0.9565
      50     52.62809980   9.01e-006    0.9150   9.61e-003   0.9589
    ...
    100      52.62763912   2.56e-007    0.9466   1.60e-003   0.9710
    200      52.62762583   3.55e-009    0.9612   1.82e-004   0.9815
    ...
    232      52.62762570   9.82e-010    0.9600   1.01e-004   0.9817
```

Again, this is the fastest method.

7.7 Multigrid conjugate gradient method

In both optimization algorithms introduced above, we repeatedly need to solve systems of linear equations. In this section, we will introduce an efficient iterative method that seems to be most suitable for these problems. Throughout this section, we assume that we want to solve the problem

$$Az = b \tag{7.22}$$

where $b \in \mathbb{R}^n$ and A is a $n \times n$ symmetric positive definite matrix.

7.7.1 Multigrid method for linear systems

Recall first the Correction Scheme (CS) version of the multigrid algorithm (see, e.g., [57]). Let *opt* denote a convergent iterative algorithm for (7.22):

$$z_{\text{new}} = opt(A, b; z, \varepsilon, \nu)\,,$$

where, on input, z is the initial approximation of the solution, ε is the required precision and ν the maximum number of iterations allowed. This will be called the *smoother*. A typical example is the Gauss-Seidel iterative method.

Assume that there exist ℓ linear operators $I_k^{k-1} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k-1}}$, $k = 2, \dots, \ell$, with $n := n_\ell > n_{\ell-1} > \cdots > n_2 > n_1$ and let $I_{k-1}^k := (I_k^{k-1})^T$. These are either constructed from finite element or finite difference refinements of some original coarse grid (geometric multigrid) or from the matrix A (algebraic multigrid); see [20] for details.

Define the “coarse level” problems

$$A_k z_k = b_k, \quad k = 1, \ldots, \ell - 1$$

with

$$A_{k-1} = I_k^{k-1}(A_k) I_{k-1}^k, \quad b_{k-1} = I_k^{k-1}(b_k), \quad k = 2, \ldots, \ell\,.$$

Algorithm MG (V-cycle correction scheme multigrid)

```
Set $\varepsilon, \varepsilon_0$. Initialize $z^{(\ell)}$.
for $i = 1 : niter$
    $z^{(\ell)} := mgm(\ell, z^{(\ell)}, b_\ell)$
    test convergence
end
function $z^{(k)} = mgm(k, z^{(k)}, r_k)$
    if $k = 0$
        $z^{(k)} := opt(A_1, b_1; z^{(k)}, \varepsilon_0, \nu_0)$                    (coarsest grid solution)
    else
        $z^{(k)} := opt(A_k, b_k; z^{(k)}, \varepsilon, \nu_1)$                      (pre-smoothing)
        $r_{k-1} = I_k^{k-1}(r_k - A_k z^{(k)})$                              (restricted residuum)
        $v^{(k-1)} = mgm(k-1, 0_{n_{k-1}}, r_{k-1})$                        (coarse grid correction)
        $z^{(k)} := z^{(k)} + I_{k-1}^k v^{(k-1)}$                              (solution update)
        $z^{(k)} := opt(A_k, b_k; z^{(k)}, \varepsilon, \nu_2)$          (post-smoothing)
    end
```

7.7.2 Multigrid preconditioned conjugate gradient method

Although the multigrid method described above is very efficient, an even more efficient tool for solving (7.22) may be the preconditioned conjugate gradient (CG) method, whereas the preconditioner consist of one step of the V-cycle multigrid method. The algorithm is described below (see, e.g., [45]).

Algorithm PCG

```
Given initial $z$, set $r := Az - b$
$y := mgm(\ell, 0_n, r)$
Set $p := -y$
for $i = 1 : niter$
    $\alpha := \frac{r^T y}{p^T A p}$
    $z := z + \alpha p$
    $\tilde{r} := r + \alpha A p$
    $\tilde{y} := mgm(\ell, 0_n, \tilde{r})$
    $\beta := \frac{\tilde{r}^T \tilde{y}}{r^T y}$
    $p := -y + \beta p$
    $r := \tilde{r}, y := \tilde{y}$
```

```
        test convergence
    end
```

7.8 Multigrid conjugate gradients for IP and OC methods

The main goal of this section (and of the whole chapter) is to study the effect of the multigrid preconditioned CG method in the IP and OC algorithms. We will also compare them to their counterparts, IP and OC with direct solvers.

The details on discretization and the choice of prolongation and restriction operators will be given in Section 7.9.

7.8.1 Multigrid conjugate gradients for IP

Our goal is to solve the linear systems arising in the Newton method, by the conjugate gradient method preconditioned by one V-type multigrid step. We can choose one of the three equivalent systems to solve, namely the full system (7.12), the reduced saddle-point system (7.15) and the so-called augmented system (7.17). We prefer the last one for the following reasons.

- The matrix Z in (7.17) is positive definite and we can thus readily apply the standard conjugate gradient method together with the standard V-cycle as a preconditioner. We could, of course, use GMRES or MINRES for the indefinite systems in (7.12) and (7.15), however, the multigrid preconditioner, in particular the smoother, would become more complicated in this case; see [76], who used so-called transforming smoothers introduced by Wittum [116].
- In order to use the multigrid preconditioner, we have to define prolongation/restriction operators for the involved variables. This can be easily done in case of the system (7.17) that only involves the displacement variable $u \in \mathbb{R}^n$ plus one additional variable λ, the

Lagrangian multiplier associated with the volume constraint; see the next Section 7.9 for details.

If, on the other hand, we decided to solve (7.12) or (7.15), we would have to select an additional restriction operator for the variables associated with the finite elements; this operator should then be "compatible" with the nodal-based restriction operator. This is a rather non-trivial task and can be simply avoided by choosing system (7.17).

The matrix Z from (7.17) is positive definite, sparse and typically has an arrow-type sparsity structure: it is banded apart from the last full row and column; see Figure 7.3-left. The bandwidth grows, approximately, with the square root of the problem size. At the same time, the number of non-zeros in each row is always the same, notwithstanding the problem size.

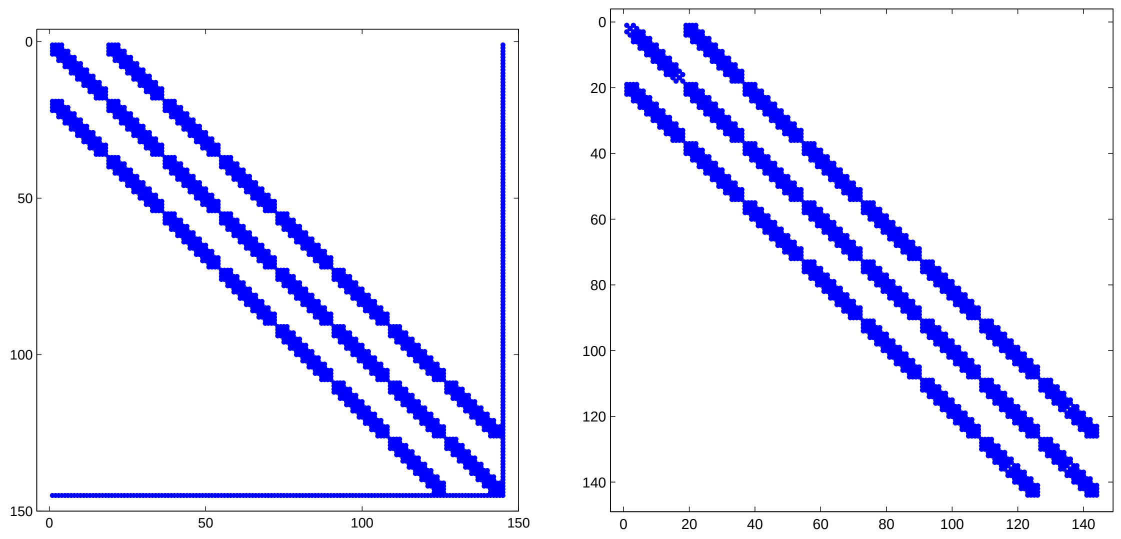


Figure 7.3: Typical sparsity structure of matrix Z from the augmented system (7.17) (left) and of the stiffness matrix K (right)

Stopping rule It is a big advantage of iterative methods, over direct solvers, that they allow us to control the precision of the approximate solution and stop whenever even a low required precision is reached. In our implementation, the PCG method is stopped whenever

$$\|r\| \; \|b\| \leqslant 10^{-2} \tag{7.23}$$

where r is residuum and b the right-hand side of the linear system, respectively. In this way we only compute an approximate Newton direction; it is shown, e.g., in [29] that the resulting method converges once the approximate Newton direction is "close enough" (though not infinitesimally close in the limit) to the exact solution of the Newton system. Furthermore, for convex quadratic programming problems, Gondzio [46] has shown that when the PCG method is stopped as soon as $\|r\| \leqslant 0.05s$ (s being the barrier parameter), the theoretical complexity of the interior point method is the same as with the exact linear solver. Inexact iterative solvers in the context of other optimization problems and algorithms were further studied, e.g., in [27, 68, 80, 107].

In our case, the value of 10^{-2} proved to be a good compromise between the overall number of Newton steps and the overall number of PCG iterations within the IP method. With this stopping criterium, the IP methods requires, typically, 2–4 PCG iterations in the initial and in many subsequent IP steps. Only when we get close to the required accuracy, in the last 2–3 IP steps, the conditioning of the matrix Z increases significantly and so does the number of PCG steps, typically to 10–30; see the next section for detailed numerical results.

7.8.2 Multigrid conjugate gradients for OC

Within the OC algorithm, the multigrid CG method will be used to solve the discretized equilibrium equation $Ku = f$. Recall that K is assumed to be a positive definite matrix. Moreover K is very sparse and, if a reasonably good numbering of the nodes is used, banded. A typical non-zero structure of K is shown in Figure 7.3-right: it is exactly the same as for the matrix in (7.17) in the IP method, apart from the additional last column and row in the augmented matrix in (7.17).

The only degrees of freedom in the resulting algorithm are the stopping criteria for the OC method and for the multigrid CG method.

The overall stopping criterium As the dual information is not readily available, so far the only practical (and widely used) stopping criterium for the OC method is the difference in the objective function value in two subsequent iterations. Needless to say that, unless we have an estimate for the rate of convergence, this criterium can be misleading and may terminate the iteration process long before some expected approximation of the optimum has been reached. Nevertheless, many numerical experiments suggest that this criterium is not as bad as it seems and serves its purpose for the OC method.

Hence the OC method is typically stopped as soon as

$$|f^T u_k - f^T u_{k-1}| \leqslant \tau_{\text{OC}} \tag{7.24}$$

where k is the iteration index. In our numerical experiments we have used $\tau_{\text{OC}} = 10^{-5}$; this value has been chosen such that the OC results are comparable to the IP results, in the number of valid digits both in the objective function and in the variables; see Section 7.10 for more details.

Stopping criterium for the multigrid CG method As already mentioned above, one of the advantages of an iterative method is the fact that an exact solution to the linear system is not always needed. In such a case, we can stop the iterative method after reaching a relatively low accuracy solution. The required accuracy of these solutions (such that the overall convergence is maintained) is well documented and theoretically supported in case of the IP method; it is, however an unknown in case of the OC method; see [3] for detailed discussion. Clearly, if the linear systems in the OC method are solved too inaccurately, the whole method may diverge or just oscillate around a point away from the solution.

We have opted for the following heuristics that guarantees the (assumed) overall convergence of the OC method. Notice that the OC method is a feasible descent algorithm. That means that every iteration is feasible and the objective function value in the k-th iteration is smaller than that in the $(k-1)$-st iteration. Hence

- we start with $\tau = 10^{-4}$;
- if $f^T u_k > f^T u_{k-1}$, we update $\tau := 0.1\,\tau$.

In our numerical tests, the update had to be done only in few cases and the smallest value of τ needed was $\tau = 10^{-6}$. Recall that this is due to our relatively mild overall stopping criterium (7.24). In the next section, we will see that this heuristics serves its purpose, as the number of OC iterations is almost always the same, whether we use an iterative or a direct solver for the linear systems.

7.9 Numerical experiments

This section contains detailed results of three numerical examples. All codes were written entirely in MATLAB. Notice, however, that when we refer to a direct solver for the solution of linear system, we mean the backslash operation in MATLAB which, for our symmetric positive definite systems, calls the CHOLMOD implementation of the Cholesky method [25]. This implementation is highly tuned, very efficient and written in the C language. So whenever we compare CPU times of the iterative solver with the direct solver, we should keep this in mind. These comparisons are given solely to show the tendency in the CPU time when increasing the problem size. All problems were solved on an Intel Core i5-3570 CPU at 3.4GHz with 8GB RAM, using MATLAB version 8.0.0 (2012b) running in 64 bit Windows 7.

In all examples, we use square finite elements with bilinear basis functions for the displacement variable u and constant basis functions for the thickness variable ρ, as it is standard in topology optimization. The prolongation operators I_{k-1}^k for the variable u are based on the nine-point interpolation scheme defined by the stencil $\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$; see, e.g., [57]. When solving the linear system (7.17) in the interior point method, we also need to prolong and restrict the single additional variable λ; here we simply use the identity.

The examples are solved with isotropic material with Young's modulus equal to 1 and Poisson's ratio 0.3. The physical dimensions of the computational domain are given by the coarsest mesh, whereas the coarse level element has dimension 1×1. The value of the force is always equal to (0,-1). The upper bound on the variable ρ is set to $\overline{\rho} = 2$.

The meaning of the captions in the following tables:

problem... the first two numbers describe the dimension of the computational domain, the last number is the number of mesh refinements

variables... number of variables in the linear systems

feval... total number of function evaluations (equal to the number of linear systems solved)

total CG iters... total number of CG iterations in the optimization process

solver CPU time... total CPU time spent in the solution of linear systems

average CG iters... average number of CG iterations per one linear system

7.9.1 Example 1

We consider a square computational domain with the coarsest mesh consisting of 2×2 elements. All nodes on the left-hand side are fixed, the right-hand middle node is subject to a vertical force; see Figure 7.4. We use up to nine refinements levels with the finest mesh having 262 144 elements and 525 312 nodal variables (after elimination of the fixed nodes).

Table 7.1 presents the results of the interior point method. We can see that, with increasing size of the problem, the total number of CG iterations is actually decreasing. This is due to our specific stopping criterium explained in the previous section. We also observe that the average number of CG iterations per linear system is very low and, in particular, *is not increasing with the problem size*, the result of the multigrid preconditioner.

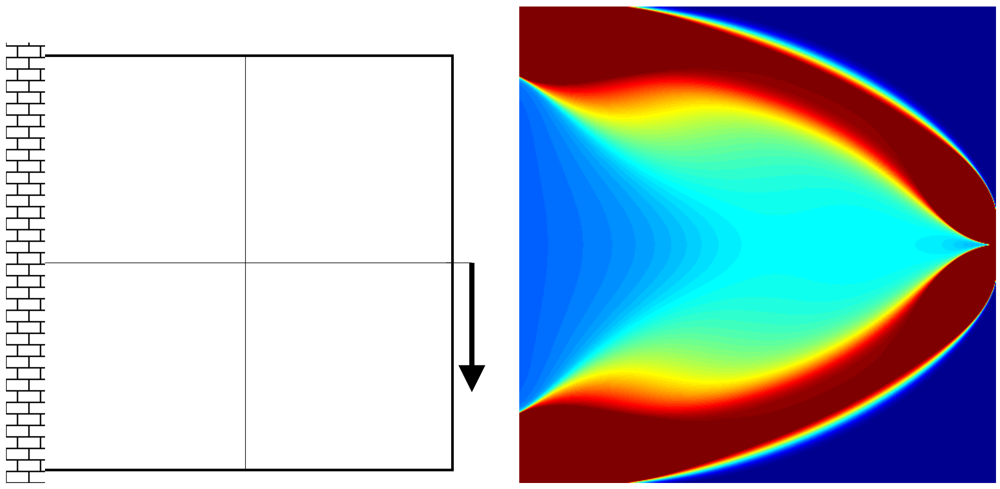


Figure 7.4: Example 1, initial setting with coarsest mesh and optimal solution.

Table 7.1: Example 1, interior point method with iterative solver

problem	variables	feval	total CG iters	solver CPU time	average CG iters
223	145	31	253	0.18	8.16
224	545	30	281	0.44	9.37
225	2 113	29	197	0.91	6.79
226	8 321	28	139	2.79	4.96
227	33 025	27	119	12.7	4.41
228	131 585	25	104	45.8	4.16
229	525 313	27	85	156.0	3.15

Let us now compare these results with those for the OC method where the linear system is just the equilibrium equation; see Table 7.2. As expected, the number of OC iterations (and thus the number of linear systems and the total number of CG iterations) *grows* with the size of the problem. Also in this case the average number of CG iterations is almost constant, notwithstanding the size of the problem.

The comparison of the interior point method with the OC method is graphically presented in Figure 7.5 (left). Here we can see, in the log-log scale, the total CPU time spent in the linear solver, growing with the size of the problem. While initially worse than the OC method, the interior point method grows slower and soon catches up and overtakes the OC method. For both methods, the growth is almost linear for the larger problems, so that we can estimate the growth in the CPU time as a polynomial function cn^d of the problem dimension n. For the

Table 7.2: Example 1, OC method with iterative solver

problem	variables	feval	total CG iters	solver CPU time	average CG iters
223	144	19	56	0.04	2.95
224	544	33	100	0.14	3.03
225	2 112	55	164	0.65	2.98
226	8 320	85	254	4.84	2.99
227	33 024	111	332	30.8	2.99
228	131 584	119	362	133.0	3.04
229	525 312	123	368	636.0	2.99

interior point method, the degree $d = 0.907$ while for the OC method $d = 1.09$. This means that the overall computational complexity of the IP method with inexact Newton and inexact multigrid CG methods is slightly *sublinear*. For the OC method, it is just a bit worse than linear.

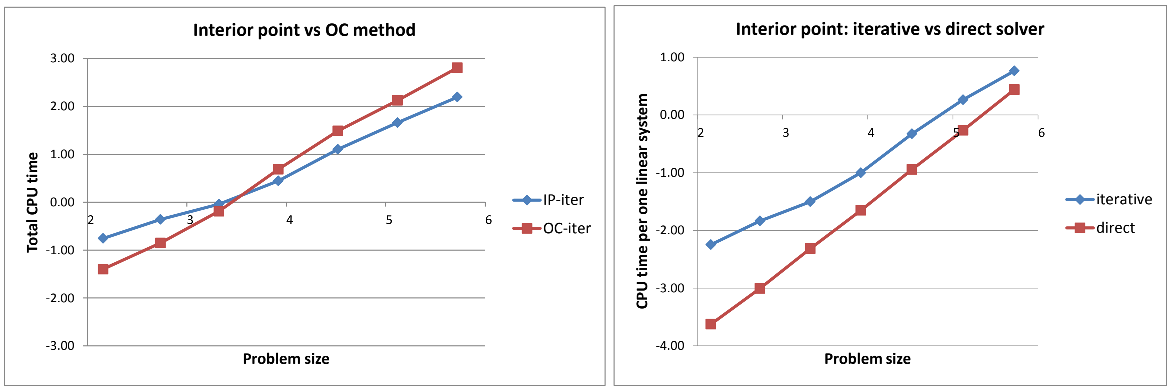


Figure 7.5: Example 1, *left*: total CPU time spent in the iterative linear solver for the interior point and the OC method; *right*: interior point method, total CPU time spent in the iterative linear solver and in the direct solver.

In Figure 7.5 (right) we compare the iterative solver used in the interior point method with a direct Cholesky solver (see the warning at beginning of this section!). We can clearly see that the time for the (C coded) direct solver grows quicker than for the (MATLAB coded) iterative solver.

7.9.2 Example 2

The next example is similar to the previous one, only the computational domain is “longer” in the horizontal direction; the coarsest mesh consists of 4×2 elements. It is well known that the conditioning of this kind of examples grows with the slenderness of the domain. As before, all nodes on the left-hand side are fixed, the right-hand middle node is subject to a vertical force; see Figure 7.6. Again, we use up to nine refinements levels with the finest mesh having 524 288 elements and 1 050 624 nodal variables (after elimination of the fixed nodes).

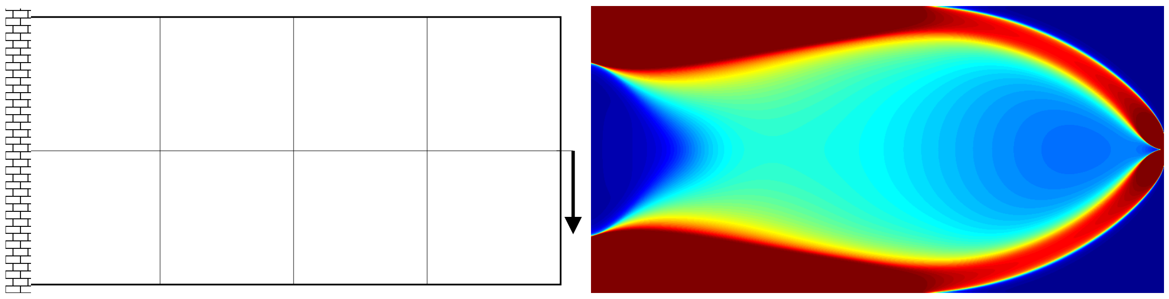


Figure 7.6: Example 2, initial setting with coarsest mesh and optimal solution.

We first show the results of the interior point method in Table 7.3. Just as in the previous example, the total number of CG iterations is decreasing with the increasing size of the problem. Again, the average number of CG iterations per linear system is very low and not increasing. Compare this with the OC solver results in Table 7.4. In this case, we only consider eight

Table 7.3: Example 2, IP method with iterative solver

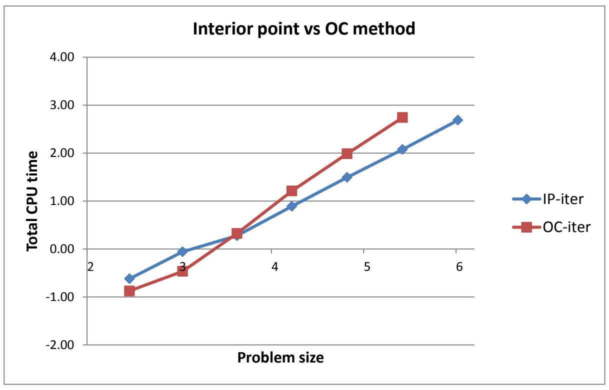
problem	variables	feval	total CG iters	solver CPU time	average CG iters
423	288	33	265	0.24	8.03
424	1 088	32	342	0.87	10.69
425	4 224	31	207	1.89	6.68
426	16 640	30	160	7.77	5.33
427	66 048	29	139	31.1	4.79
428	263 168	27	123	119.0	4.56
429	1 050 624	27	101	385.0	3.74

refinement levels, as the largest problem would take too much time on our computer. Contrary to the previous example, the average number of CG iterations is slightly increasing due to the worse conditioning.

Table 7.4: Example 2, OC method with iterative solver

problem	variables	feval	total CG iters	solver CPU time	average CG iters
423	288	39	117	0.13	3.00
424	1 088	45	144	0.34	3.20
425	4 224	77	262	2.10	3.40
426	16 640	123	423	16.2	3.44
427	66 048	157	542	97.1	3.45
428	263 168	165	739	552	4.48

Figure 7.7 (left) gives the comparison of the interior point with the OC method. We can see even more clearly than in the previous example the faster growth of the OC method. When we calculate the degree of the assumed polynomial function cn^d of the problem dimension n from the larger examples, we will obtain $d = 0.944$ for the interior point method (so a linear growth) and $d = 1.28$ for the OC method.



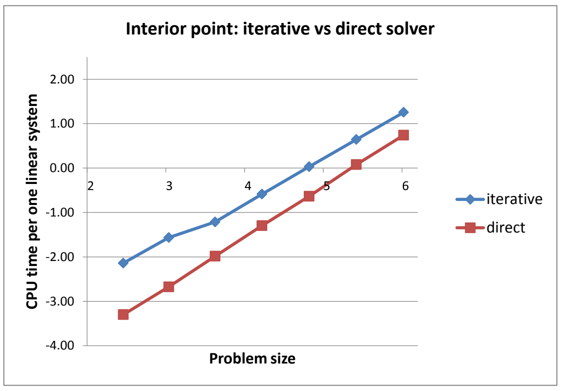


Figure 7.7: Example 2, *left*: total CPU time spent in the iterative linear solver for the interior point and the OC method; *right*: interior point method, total CPU time spent in the iterative linear solver and in the direct solver.

Figure 7.7 (right) compares the iterative solver used in the interior point method with the Cholesky solver (see the beginning of this section), giving the same picture as in the previous example.

Finally in Figure 7.8 we compare the average number of CG steps per linear system in the interior point and the OC solver. We can see that while the graph is decreasing for the IP method, it is slowly increasing in case of the OC method. The reason for that is that, in this example, we had to decrease the stopping criterium for the CG solver in the OC method, in order to guarantee its convergence (see Section 7.8.2 for explanation).

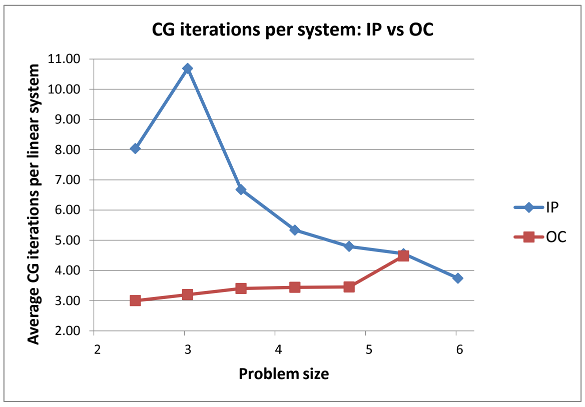


Figure 7.8: Example 2, average number of CG iterations per linear system for the interior point and the OC method.

7.9.3 Example 3

The computational domain for our final example is a rectangle, initially discretized by 8×2 finite elements. The two corner points on the lower edge are fixed and a vertical force is applied in the middle point of this edge; see Figure 7.9. We use up to eight refinement levels with the finest mesh having 262 144 elements and 568 850 nodal variables (after elimination of the fixed nodes).

The results of the interior point method are shown in Table 7.5. Yet again, the total number of CG iterations is decreasing with the increasing size of the problem and the average number of CG iterations per linear system is very low and not increasing. The negative complexity factor is caused by the exceptional difficulties of the CG method in the last interior point step in problem 823.

Table 7.3 presents the results of the OC method. As in Example 2, the average number of CG iterations is increasing due to the worse conditioning.

Table 7.5: Example 3, IP method with iterative solver

problem	variables	feval	total CG iters	solver CPU time	average CG iters
822	170	33	284	0.21	8.61
823	594	31	383	0.78	12.35
824	2210	32	121	0.60	3.78
825	8514	31	166	3.41	5.35
826	33410	26	140	14.8	5.38
827	132354	26	133	78.8	5.12
828	526850	25	121	217.0	4.84

Table 7.6: Example 3, OC method with iterative solver

problem	variables	feval	total CG iters	solver CPU time	average CG iters
822	170	23	69	0.04	3.00
823	594	37	147	0.21	3.97
824	2210	57	267	1.16	4.68
825	8514	75	374	7.40	4.99
826	33410	99	495	51.8	5.00
827	132354	111	665	290.0	5.99
828	526850	113	677	1250.0	5.99

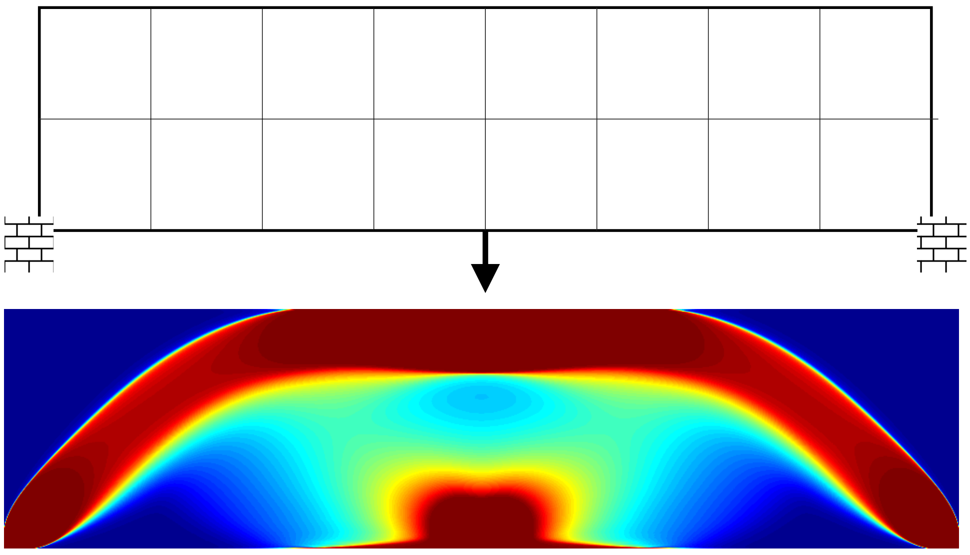


Figure 7.9: Example 3, initial setting with coarsest mesh and optimal solution.

Figure 7.10 (left) compares of the interior point with the OC method. Yet again, the interior point method is a clear winner, both in the absolute timing as in the growth tendency. Calculating the degree of the assumed polynomial function cn^d of the problem dimension n from the larger examples, we get $d = 1.09$ for the interior point method and $d = 1.24$ for the OC method.

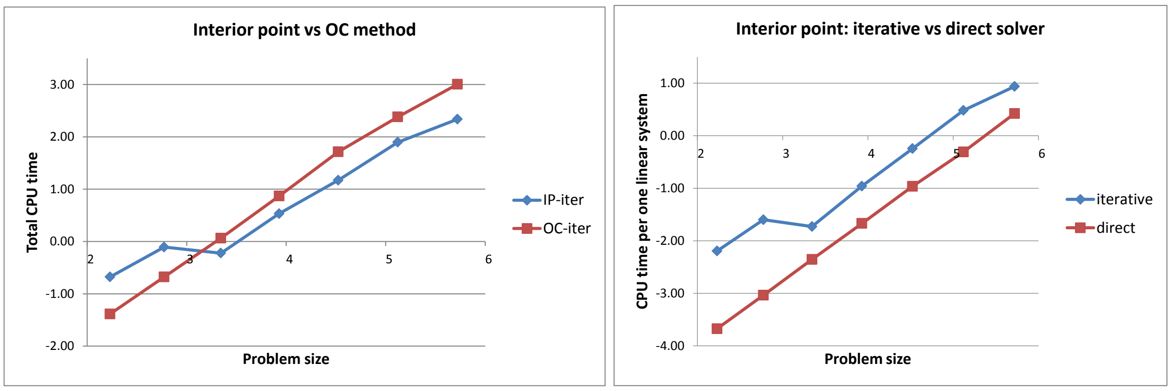


Figure 7.10: Example 3, *left*: total CPU time spent in the iterative linear solver for the interior point and the OC method; *right*: interior point method, total CPU time spent in the iterative linear solver and in the direct solver.

In Figure 7.10 (right) we compare the iterative solver used in the interior point method with the Cholesky solver (see the beginning of this section). Finally in Figure 7.11 we compare the average number of CG steps per linear system in the interior point and the OC solver. We can see that while the graph for the IP method has a decreasing tendency, it is increasing in case of the OC method. As before, the reason for that is that we had to decrease the stopping criterium for the CG solver, in order to guarantee its convergence (see Section 7.8.2).

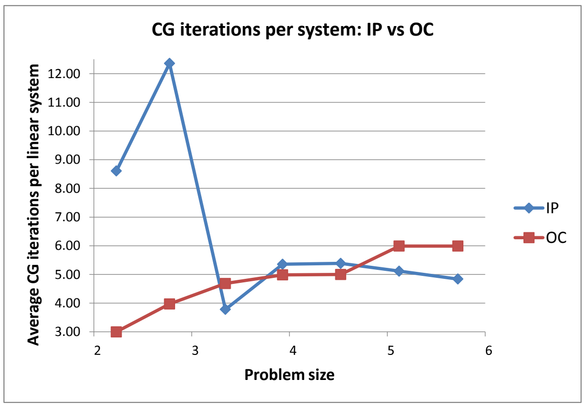


Figure 7.11: Example 3, average number of CG iterations per linear system for the interior point and the OC method.

7.10 How exact is ‘exact’?

7.10.1 Interior point method

In this approach we are using slightly nonstandard stopping criteria within the interior point method. In particular, with the decreasing barrier parameters s_1, s_2 we do *not* decrease the stopping tolerances τ_{NWT} and τ_{CG} for the Newton method and for the conjugate gradients, respectively, although both is required for the theoretical convergence proof. In Figure 7.12 we try to give a schematic explanation. Here we depict the feasible region and three points ρ_1, ρ_2, ρ_3 on the central path, corresponding to three values of the barrier parameter $r_1 > r_2 > r_3$. The exact solution lies in the corner of the feasible region. The circle around each of these points depict the region of stopping tolerance of the Newton method, once we get within, the Newton method will stop. The radius of these circles is decreasing, even though τ_{NWT} is kept constant. The idea is now obvious: it is “better” to stay within the tolerance circle of ρ_3 rather than to get very close to ρ_2.

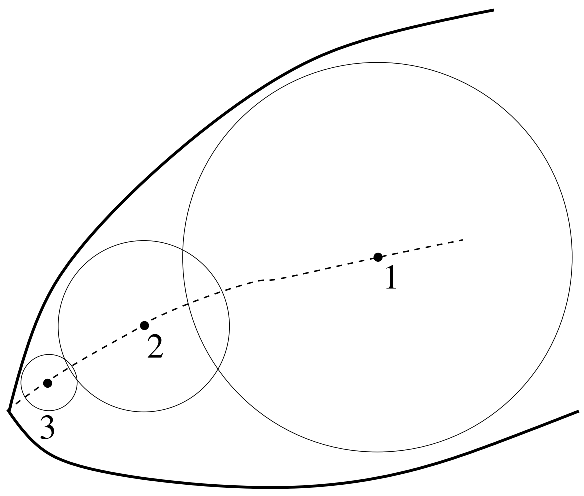


Figure 7.12:

In the lemma below, ρ^* is a point on the central path corresponding to a barrier parameter s_1 and ρ an approximation of ρ^* resulting from inexact Newton method. We will show that, even with a fixed stopping criterium for the Newton method, ρ must converge to ρ^* with s going to zero. For simplicity of notation, we will just verify it for the lower bound complementarity part of $\widetilde{\mathrm{Res}}^{(3)}$.

Lemma 7.10.1. *Let $\rho^* > 0$ satisfies the perturbed scaled complementary condition*

$$\frac{\varphi_i \rho_i^* - s_1}{\rho_i} = 0, \quad i = 1, \dots, m \tag{7.25}$$

and let $\rho > 0$ be an approximation of ρ^* satisfying

$$\|z\| \leqslant \tau, \quad z_i = \frac{\varphi_i \rho_i - s_1}{\rho_i} \tag{7.26}$$

with some $\tau > 0$. Then there is an $\varepsilon > 0$ depending on s_1 and τ such that $\|\rho^* - \rho\| \leqslant \varepsilon$. Moreover, if s_1 tends to zero then also ε tends to zero.

Proof. From (7.25) we have that $\varphi_i = \frac{s_1}{\rho_i^*}$ and thus (7.26) can be written as

$$\sum_{i=1}^{m} \left(\frac{s_1}{\rho_i^*} - \frac{s_1}{\rho_i} \right)^2 \leqslant \tau^2$$

which is, in particular, means that

$$\left|\frac{s_1}{\rho_i^*} - \frac{s_1}{\rho_i}\right| \leqslant \tau, \quad i = 1, \dots, m\,,$$

i.e.,

$$\frac{|\rho_i^* - \rho_i|}{\rho_i^* \rho_i} \leqslant s_1 \tau, \quad i = 1, \dots, m\,.$$

Clearly, when s_1 tends to zero, ρ must tend to ρ^*. $\square$

How good solution can we get when replacing the ("exact") direct solver by an inexact iterative method for the solution of the Newton systems? We may expect that, with the ever decreasing barrier parameter, the inexact version will get into numerical difficulties sooner than the exact one. Table 7.7 answers this question. In topology optimization, the important variable is ρ, the "density". With lower bound equal to zero, the quality of the solution may be characterized by the closeness of components of ρ to this lower bound (that is, in examples where the lower bound is expected to be reached, such as in Example 1 with sufficiently fine discretization). In Table 7.7 we display the smallest component of ρ, denoted by $\rho_{\min}$ for Example 1 with 6 refinements levels, i.e., example 226 from Table 7.2. The meaning of other columns in Table 7.7 is the following:

barrier. . . the smallest value of the barrier parameters s_1, s_2 before the interior point algorithm was terminated;

IP,NWT,CG. . . the total number of iterations of the interior point method, the Newton method and conjugate gradients, respectively;

Cholesky. . . the linear system was solved by the CHOLMOD implementation of the Cholesky method;

CG tol fixed. . . the linear system was solved by the multigrid preconditioned conjugate gradient method with a fixed stopping criterium $\|r\|\ \|b\| \leqslant 10^{-2}$; see (7.23);

CG tol decreasing... as above but with a variable stopping criterium $\|r\|\,\|b\| \leqslant \tau_{\text{CG}}$, where τ_{CG} is initially equal to 10^{-2} and is then multiplied by 0.5 after each major iteration of the interior point method.

Table 7.7: Number of iterations and error in the IP solution for different values of τ_{IP} and three different linear solvers.

		Cholesky		CG tol fixed			CG tol decreasing		
τ_{IP}	IP	NWT	$\rho_{\min}$	NWT	CG	$\rho_{\min}$	NWT	CG	$\rho_{\min}$
10^{-8}	12	28	$1.6 \cdot 10^{-5}$	28	139	$1.8 \cdot 10^{-5}$	28	587	$1.6 \cdot 10^{-5}$
10^{-10}	15	34	$1.0 \cdot 10^{-7}$	35	291	$2.7 \cdot 10^{-7}$	34	4285	$1.0 \cdot 10^{-7}$
10^{-12}	18	40	$1.0 \cdot 10^{-9}$	72	2832	$2.4 \cdot 10^{-9}$	40	10042	$1.5 \cdot 10^{-9}$
10^{-14}	21	46	$6.4 \cdot 10^{-12}$	296	63674	$1.9 \cdot 10^{-11}$	53	23042	$1.9 \cdot 10^{-11}$
10^{-16}	24	52	$6.2 \cdot 10^{-14}$	489	88684	$1.4 \cdot 10^{-13}$	82	52042	$1.4 \cdot 10^{-13}$

We can see that all three algorithms were able to solve the problem to very high accuracy. However, both versions of the CG method had problems with very low values of the barrier parameter. The "CG tol fixed" version needed very high number of the Newton steps, while the "CG tol decreasing" version needed very high number of the CG steps to reach the increased accuracy. (Notice that the maximum number of CG iterations for one system was limited to 1000.) On the other hand, for barrier parameter equal to 10^{-8} (our choice in the numerical examples above), both inexact solvers were on par with the exact one and, due to the lower accuracy required and thus lower number of CG steps, the "CG tol fixed" version is the method of choice.

7.10.2 OC method

In the OC method, we have to solve the equilibrium problem with the stiffness matrix $K(\rho)$; that means, $K(\rho)$ must not be singular. A common way how to approach this is to assume that ρ is strictly positive, though very small. Typically, one would modify the lower bound constraint to $0 < \underline{\rho} \leqslant \rho_i$, $i = 1, \ldots, m$ with $\underline{\rho} = 10^{-6}$, for instance. Once the OC method is terminated, all values of ρ with $\rho_i = \underline{\rho}$ are set to zero. This is usually considered a weakness of the OC method, because we do not exactly solve the original problem, only its approximation (see [1]). Somewhat surprisingly, in the examples we solved using our MATLAB code, the value of $\underline{\rho}$ could be actually very low, such as $\underline{\rho} = 10^{-30}$. The stiffness matrix $K(\rho)$ will, consequently, become extremely ill-conditioned (in the above case the condition number will be of order 10^{30}), nevertheless, CHOLMOD does not seem to have a problem with that and the OC method converges in about the same number of iterations as if we set $\underline{\rho} = 10^{-6}$.

The main question is how does the quality of the solution depends on the heuristic stopping criterium (7.24). Our next Table 7.8 sheds some light on this. We solve the example 226 from Table 7.2 for various values of the stopping criterium τ_{OC} and two different values of the lower bound $\underline{\rho}$. We then compute, pair-wise, the norm of the difference of these solution. Notice that the stopping criterium $\tau_{\mathrm{OC}} = 10^{-50}$ and, in this case, the OC method has been terminated after 5000 iterations (i.e., 10000 solutions of the linear system). For instance, the maximum norm of the difference between the solutions with $\tau_{\mathrm{OC}} = 10^{-5}$ and $\tau_{\mathrm{OC}} = 10^{-50}$ is $\|\rho_{-5} - \rho_{-50}\|_\infty = 0.126$, while $\|\rho_{-9} - \rho_{-50}\|_\infty = 0.016$. Notice that the norm is not scaled, e.g., by the dimension of ρ, hence the numbers are relatively large. Also, to get a clearer picture, we used a direct linear system solver.

Table 7.8: The norm of difference of two OC solutions ρ for various values of the stopping criterium $\tau_{\mathrm{OC}} = 10^{-5}, 10^{-7}, 10^{-9}, 10^{-50}$, and for two values of the lower bound $\underline{\rho} = 10^{-7}$ and $\underline{\rho} = 10^{-17}$. Upper triangle shows the 2-norm, lower triangle the infinity norm.

lower bound		10^{-7}				10^{-17}			
	τ_{OC}	-5	-7	-9	-50	-5	-7	-9	-50
10^{-7}	-5	0	1.09	1.19	1.26	2e-6	1.10	1.18	1.26
	-7	0.114	0	0.116	0.281	1.09	0.013	0.110	0.281
	-9	0.123	0.01	0	0.190	1.19	0.103	0.009	0.190
	-50	0.126	0.025	0.016	0	1.26	0.271	0.198	4e-6
10^{-17}	-5	2e-7	0.114	0.123	0.126	0	1.10	1.19	1.26
	-7	0.116	0.001	0.009	0.024	0.116	0	0.094	0.271
	-9	0.123	0.009	8e-4	0.017	0.123	0.008	0	0.198
	-50	0.126	0.025	0.016	3e-7	0.126	0.024	0.017	0

7.10.3 Interior point versus OC method

We again solve example 226 from Table 7.2, this time by the interior point method with an exact linear solver and various stopping parameters τ_{IP}. In Table 7.9, these solutions are compared (in two different norms), to the 'exact' solution obtained in the previous section by 5000 iterations of the OC method with $\underline{\rho} = 10^{-17}$. Comparing these numbers to those in Table 7.8, we can see that the IP method delivers very good solution already for our standard value $\tau_{\text{IP}} = 1.00.10^{-8}$; this is comparable to OC solution with $\tau_{\text{OC}} = 1.00.10^{-7}$. Moreover, decrease of τ_{IP} leads to a rapid decrease of the error, unlike in the OC method.

Table 7.9: Two different norms of the error of the IP method in variable ρ for different values of the stopping parameter τ_{IP}. As an 'exact' solution ρ^* we take the OC solution after 5000 iterations with lower bound $\underline{\rho} = 10^{-17}$.

τ_{IP}	$\|\rho - \rho^*\|_2$	$\|\rho - \rho^*\|_\infty$
$1.00.10^{-8}$	$2.47.10^{-1}$	$2.49.10^{-2}$
$1.00.10^{-10}$	$9.60.10^{-3}$	$1.50.10^{-3}$
$1.00.10^{-12}$	$2.07.10^{-4}$	$4.95.10^{-5}$
$1.00.10^{-14}$	$1.70.10^{-6}$	$4.66.10^{-7}$

Finally, we compare the quality of the solution by evaluating the optimal objective value. Notice first that all solutions generated by both the OC and the IP method are feasible, hence when comparing the objective values it holds "the lower the better". In Table 7.10 we give the objective values computed by two versions of the IP algorithm and by the OC method, all with two different stopping criteria. In particular,

IP-Chol-n stands for the IP method with Cholesky factorization and with $\tau_{\text{IP}} = 10^{-n}$.

IP-mgm-n stands for the IP method with the multigrid preconditioned CG method and with $\tau_{\text{IP}} = 10^{-n}$.

OC-n stands for the OC method with Cholesky factorization and with $\tau_{\text{OC}} = 10^{-n}$.

We solved problems 223–229; the row IP-Chol-14 shows the "exact" solution where the last displayed digit would not change if we increased the precision (the value of $\tau_{\text{IP}} = 10^{-14}$ was found experimentally). The remaining rows show the computed objective values with incorrect digits in boldface and underlined (an approximation $\tilde{z}$ to z has p correct significant digits if $\tilde{z}$ and z round to the same number to p significant digits). Looking at the table, we can make several conclusions:

- The number of correct significant digits is decreasing with the increasing size of the problem. This holds for all tested algorithms. This is despite the fact that the stopping criterion in the IP method — the satisfaction of the KKT conditions — takes into account the problem size.
- IP-Chol is always better than IP-mgm with the same stopping criterion. This is due to the inexact solution of the linear system. This fact is more obvious for smaller problems.
- The IP method (in particular IP-Chol) is very predictable—by decreasing τ_{IP} by two orders, we typically gain two more correct significant digits.

Table 7.10: Objective function (compliance) values on different levels computed by the IP and OC methods for Example 1.

problem	223	224	225	226	227	228	229
IP-Chol-14	5.99863328	6.20894142	6.42788583	6.64721163	6.86709911	7.08741189	7.30790336
IP-mgm-8	5.99863**745**	6.20899**584**	6.428**04879**	6.647**32131**	6.867**45856**	7.08**821162**	7.31**305766**
IP-Chol-8	5.998633**55**	6.20894**390**	6.4279**0129**	6.647**29076**	6.867**45909**	7.08**833561**	7.31**171044**
IP-mgm-10	5.998632**57**	6.2089**6690**	6.427**99051**	6.647**34688**	6.867**17581**	7.087**55905**	7.308**12138**
IP-Chol-10	5.9986332**9**	6.2089414**4**	6.427885**94**	6.647212**20**	6.86710**169**	7.0874**1881**	7.3079**3294**
OC-5	5.99863**664**	6.2089**6326**	6.4279**1540**	6.647**29898**	6.867**28948**	7.08**771830**	7.308**26717**
OC-7	5.998633**36**	6.208941**81**	6.42788**669**	6.64721**318**	6.86710**229**	7.0874**2640**	7.307**97924**

7.11 Conclusions

Based on the results of our numerical experiments, we make the following conclusions.

- The interior point method clearly outperforms the OC method on large-scale problems. The larger the problem, the bigger the difference. This is independent of the fact whether direct or iterative solver is used for the linear system.

- The inexact multigrid preconditioned CG method outperforms even a very sophisticated direct solver, at least for large to very-large scale problems. This holds for both, the interior point and the OC method. Especially if the method is coded in C because the most time is spent in smoother symmetric Gauss Seidel (SGS). Namely, SGS needs computing $triu$ and $tril$ (upper and lower triangle, respectively) which are MATLAB commands. It would be much faster if these commands are implemented in C.

- Very large problem can be implemented by using the current idea of the inexact multigrid preconditioned CG method on parallel computers. However, neither SGS nor Cholesky can not be easily parallelized; then the real benefit of the parallel approach might be to have a smoother can be easily parallelized. The question is, what is a good parallelized smoother? There is a nice paper by Adam et. al. [2] says that the best smoother which easily parallelizable and comparable to SGS is the polynomial smoother, in particular, Chybeshev with cubic polynomials. This idea has not implemented in the current thesis because we implemented everything on series computers not on parallel computers, then the difference can not been seen but one can rely on the literature from standard multigrid.

- Also in the OC method, the multigrid preconditioned CG algorithm is predictable and very stable, both with respect to the size of the problem and of the OC iteration (and thus of the condition number of the stiffness matrix). Perhaps rather surprisingly, not more than 10 CG iterations are needed, even when high precision of the OC method is required. This is the effect of the multigrid preconditioner: notice that in [113] the authors report about 100–200 CG steps needed (with a different preconditioner) and thus propose to use so-called recycling of the Krylov subspaces, in order to accelerate CG convergence speed. This is just not needed here, given the very low number of CG steps.

- The behaviour of the interior point method is very predictable. More surprisingly, also the behaviour of the chosen iterative method, the multigrid preconditioned conjugate gradients, is also very predictable and *independent on the size of the problem.*

- The OC method has one noticeable advantage to the interior point method. It can quickly identify the "very strongly" active constraints, those with large Lagrangian multiplier. Due to the projection of variables on the feasible set, the active variables are then exactly equal to the bounds. Contrary to that, the interior point method only approaches the boundary. This may be particularly significant in case of lower bounds, when the user has to decide which values are cut off and considered zero (and thus interpreted as void). Clearly, the lower bound for the OC method has to be positive but it can be set very low (e.g., 10^{-17}) and is then exactly reached.

From the above, it seems to be obvious to recommend the interior point method with multigrid preconditioned CG solver as the method of choice for large scale topology optimization problem. However, we should keep in mind that the use of multigrid is rather restricted by the assumed existence of regularly refined finite element meshes. This is easily accomplished when using "academic" examples with regular computational domains such as squares, rectangles, prisms and unions of these. For geometrically complex domains appearing in practical examples, multigrid may not be so suitable or may even be unusable. In these cases, we can resort to domain decomposition preconditioners. In [70] it was shown that, in connection with the interior point method, they also lead to very efficient techniques for topology optimization problems.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This dissertation has investigated the use of multigrid methods in certain classes of optimization problems, with emphasis on structural, namely topology, optimization. In the first part, we have investigated the solution bound constrained optimization problems arising in discretization by the finite element method, such as elliptic variational inequalities. For these problems we have proposed a "direct" multigrid approach which is a generalization of existing multigrid methods for variational inequalities. We have proposed a nonlinear first order method as a smoother that reduces memory requirements and improves the efficiency of the resulting algorithm, as documented on several numerical examples.

In the second part of the thesis, we intended to apply the same direct multigrid approach to the solution of the topology optimization problem. It turns out that this problem is not suitable for the direct approach, due to the presence of two different types of variables. Instead, we have proposed to use an interior point method as an "outer" solver of the discretized optimization problem. The large scale systems of linear equations (the Newton equations) arising in the interior point method are then solved by standard linear multigrid techniques, adopted for the Newton equations. More precisely, we use the multigrid preconditioned conjugate gradient method. The resulting algorithm turns out to be very efficient and clearly outperforms the interior point method using a state-of-the art direct solver. Moreover, the behaviour of the multigrid conjugate gradient method within the interior point algorithm is very stable, predictable and, most of all, independent of the problem size.

In the future work, we would like to address the question of finding a larger class of optimization problems approachable by the first order direct multigrid method introduced in the first part of the thesis. However, on of the results of our study is the fact that the direct approach is limited to problems with relatively simple constraints. For more complicated optimization problems, we propose to use a linear multigrid method within a standard optimization solver such as the interior point method employed in this thesis. Furthermore, to attack even more general problems for which geometric multigrid cannot be used and which, perhaps, do not arise from the discretization of infinite dimensional problems, the use of algebraic multigrid within a standard optimization solver should be investigated. The main benefit would again be the independence of the solver efficiency on the size of the problem.

REFERENCES

[1] W. Achtziger. Local stability of trusses in the context of topology optimization part I: exact modelling. *Structural Optimization*, 17(4):235–246, 1999.

[2] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus gauss–seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

[3] O. Amir, N. Aage, and S. Lazarov. On multigrid-cg for efficient topology optimization. *Structural and Multidisciplinary Optimization*, 49(5):815–829, 2014.

[4] O. Axelsson. A generalized conjugate gradient, least square method. *Numerische Mathematik*, 51(2):209–227, 1987.

[5] O. Axelsson. *Iterative solution methods*. Cambridge university press, 1996.

[6] L. Badea. Global convergence rate of a standard multigrid method for variational inequalities. *IMA Journal of Numerical Analysis*, page drs054, 2013.

[7] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[8] A. Ben-Tal and M. Bendsøe. A new method for optimal truss topology design. *SIAM Journal on Optimization*, 3(2):322–358, 1993.

[9] M. Bendsøe. *Optimization of structural topology, shape, and material*. Berlin: Springer-Verlag, 1995.

[10] M. Bendsøe and O. Sigmund. *Topology Optimization. Theory, Methods and Applications*. Springer-Verlag, Heidelberg, 2003.

[11] Martin P Bendsoe, JM Guedes, Robert B Haber, P Pedersen, and JE Taylor. An analytical model to predict optimal material properties in the context of optimal structural design. *Journal of Applied Mechanics*, 61(4):930–937, 1994.

[12] D. Bertsekas. On the goldstein-levitin-polyak gradient projection method. *Automatic Control, IEEE Transactions on*, 21(2):174–184, 1976.

[13] D. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[14] A. Borzì and V. Schulz. Multigrid methods for pde optimization. *SIAM review*, 51(2):361–395, 2009.

[15] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, New York, 2004.

[16] A. Brandt. Multi-level adaptive technique (mlat) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pages 82–89. Springer, 1973.

[17] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[18] A. Brandt. *Multigrid techniques: 1984 guide with applications to fluid dynamics*. Gesellschaft für Mathematik und Datenverarbeitung, 1984.

[19] A. Brandt and C. W. Cryer. Multigrid algorithms for the solution of linear complementarity problems arising from free boundary problems. *SIAM journal on scientific and statistical computing*, 4(4):655–684, 1983.

[20] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*, volume 72. Society for Industrial and Applied Mathematics, 2000.

[21] H. Calamai and J. Moré. Projected gradient methods for linearly constrained problems. *Math. Programming*, 39(1):93–116, 1987.

[22] W. Carroll. The created response surface technique for optimizing nonlinear, restrained systems. *Operations Research*, 9(2):169–184, 1961.

[23] J. Céa and K. Malanowski. An example of a max-min problem in partial differential equations. *SIAM Journal on Control*, 8(3):305–316, 1970.

[24] J. Cea and V. Murthy. *Optimization: Theory and algorithms*. Springer-Verlag Berlin, Germany, 1978.

[25] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22, 2008.

[26] Ph. G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics SIAM, Philadelphia, PA, 2002. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].

[27] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust region methods*. Society for Industrial and Applied Mathematics SIAM, 2000.

[28] Y. Dai and R. Fletcher. Projected barzilai-borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

[29] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM journal on Numerical Analysis*, 9:400–408, 1982.

[30] T. Dreyer, B. Maar, and V. Schulz. Multigrid optimization in applications. *Journal of Computational and Applied Mathematics*, 120(1):67–84, 2000.

[31] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. Clarendon press Oxford, 1986.

[32] J. C. Dunn. Global and asymptotic convergence rate estimates for a class of projected gradient processes. *SIAM Journal on Control and Optimization*, 19(3):368–400, 1981.

[33] H. C. Elman. *Iterative methods for large, sparse, nonsymmetric systems of linear equations*. PhD thesis, Yale University New Haven, Conn, 1982.

[34] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1(5):922–927, 1961.

[35] R. P. Fedorenko. The speed of convergence of one iterative process. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 4(3):559–564, 1964.

[36] A. V. Fiacco and G. P. McCormick. *Nonlinear programming: Sequential unconstrained minimization techniques*. John Wiley and Sons, Inc., New York-London-Sydney, 1968.

[37] A. V. Fiacco and G. P. McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*, volume 4. Society for Industrial and Applied Mathematics, 1990.

[38] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[39] E. Frandi and A. Papini. Coordinate search algorithms in multilevel optimization. *Optimization Methods and Software*, 29(5):1020–1041, 2014.

[40] KR Frisch. The logarithmic potential method of convex programming. *Memorandum, University Institute of Economics, Oslo*, 5(6), 1955.

[41] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM journal on optimization*, 12(4):979–1006, 2002.

[42] R. Glowinski. *Numerical methods for nonlinear variational problems*, volume 4. Springer, 1984.

[43] D. Goldfarb and Z. Wen. A line search multigrid method for large-scale convex optimization, 2007.

[44] A. Goldstein. Convex programming in Hilbert space. *Bull. Amer. Math. Soc.*, 70:709–710, 1964.

[45] G. H. Golub and Ch. F. Van Loan. *Matrix computations*. JHU Press, 2012.

[46] J. Gondzio. Interior point method for convex quadratic programming. *SIAM journal on Optimization*, 23(3):1510–1527, 2013.

[47] N. Gould. An introduction to algorithms for continuous optimization, 2006.

[48] C. Gräser and R. Kornhuber. Multigrid methods for obstacle problems. *J. Comput. Math.*, 27(1):1–44, 2009.

[49] C. Gräser and R. Kornhuber. Nonsmooth newton methods for set-valued saddle point problems. *SIAM Journal on Numerical Analysis*, 47(2):1251–1273, 2009.

[50] C. Gräser, U. Sack, and O. Sander. Truncated nonsmooth newton multigrid methods for convex minimization problems. In *Domain Decomposition Methods in Science and Engineering XVIII*, pages 129–136. Springer, 2009.

[51] S. Gratton, M. Mouffe, A. Sartenaer, P. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. *Optimization Methods & Software*, 25(3):359–386, 2010.

[52] S. Gratton, M. Mouffe, P. L. Toint, and M. Weber-Mendonça. A recursive-trust-region method for bound-constrained nonlinear optimization. *IMA Journal of Numerical Analysis*, 28(4):827–861, 2008.

[53] S. Gratton, A. Sartenaer, and P. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *Society for Industrial and Applied Mathematics Journal on Optimization*, 19(1):414–444, 2008.

[54] A. Greenbaum. *Iterative methods for solving linear systems*, volume 17. Society for Industrial and Applied Mathematics SIAM, 1997.

[55] Ch. Gross and R. Krause. On the convergence of recursive trust-region methods for multiscale nonlinear optimization and applications to nonlinear mechanics. *SIAM Journal on Numerical Analysis*, 47(4):3044–3069, 2009.

[56] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2014.

[57] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer-Verlag Berlin, 1985.

[58] W. Hackbusch. Multigrid methods for fem and bem applications. *Encyclopedia of Computational Mechanics*, 2003.

[59] W. Hackbusch and H. D. Mittelmann. On multi-grid methods for variational inequalities. *Numerische Mathematik*, 42:65–76, 1983.

[60] W. Hackbusch and U. Trottenberg. Multigrid methods: proceedings of the conference, held at koeln-porz, november 23-27, 1981. *Lecture Notes in Mathematics*, (960), 1986.

[61] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. 1952.

[62] R. H. W. Hoppe. Multigrid algorithms for variational inequalities. *SIAM Journal on Numerical Analysis*, 24(5):1046–1065, 1987.

[63] R. H. W. Hoppe. Two-sided approximations for unilateral variational inequalities by multi-grid methods. *Optimization*, 18(6):867–881, 1987.

[64] F. Jarre, M. Kočvara, and J. Zowe. Optimal truss design by interior-point methods. *SIAM Journal on Optimization*, 8(4):1084–1107, 1998.

[65] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[66] C. T. Kelley. *Iterative methods for optimization*, volume 18. Society for Industrial and Applied Mathematics, 1987.

[67] M. Kočvara and M. Stingl. Solving nonconvex sdp problems of structural optimization with stability control. *Optimization Methods and Software*, 19(5):595–609, 2004.

[68] M. Kočvara and M. Stingl. On the solution of large-scale SDP problems by the modified barrier method using iterative solvers. *Mathematical Programming*, 109(2–3):413–444, 2007.

[69] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities I. *Numerische Mathematik*, 69:167–184, 1994.

[70] M. Kočvara, D. Loghin, and J. Turner. Constraint interface preconditioning for topology optimization problems. *SIAM journal on Scientific Computing*, 2015. Sumitted.

[71] R. H. Krause. *Monotone multigrid methods for Signorini's problem with friction*. PhD thesis, Freie Universität Berlin, Universitätsbibliothek, 2001.

[72] E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

[73] R. M. Lewis and S. G. Nash. A multigrid approach to the optimization of systems governed by differential equations. *AIAA paper*, 4890:2000, 2000.

[74] R. M. Lewis and S. G. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM Journal on Scientific Computing*, 26(6):1811–1837, 2005.

[75] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 116. Springer, 2008.

[76] B. Maar and V. Schulz. Interior point multigrid methods for topology optimization. *Structural and Multidisciplinary Optimization*, 19(3):214–224, 2000.

[77] J. Mandel. Étude algébrique d'une méthode multigrille pour quelques problèmes de frontière libre. *Comptes rendus des séances de l'Académie des sciences. Série 1, Mathématique*, 298(18):469–472, 1984.

[78] J. Mandel. A multilevel iterative method for symmetric, positive definite linear complementarity problems. *Applied Mathematics and Optimization*, 11(1):77–95, 1984.

[79] S. F. McCormick. Multigrid methods for variational problems: general theory for the V-cycle. *SIAM Journal on Numerical Analysis*, 22(4):634–643, 1985.

[80] S. Mizuno and F. Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Mathematical Programming*, 84(1):105–122, 1999.

[81] J. L. Morales and J. Nocedal. Remark on Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):7, 2011.

[82] S. G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000.

[83] S. G. Nash. Convergence and descent properties for a class of multilevel optimization algorithms. Technical report, Tech. Rep., Department of Systems Engineering and Operations Research, George Mason University, Fairfax, VA, 2010.

[84] J. Neċas and I.Hlaváċek. *Mathematical theory of elastic and elastico-plastic bodies: An introduction*. J. Neċas and I.Hlaváċek, Elsevier Amsterdam, 1981.

[85] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Science+ Business Media, 2006.

[86] J. Outrata, M. Kocvara, and J. Zowe. *Nonsmooth approach to optimization problems with equilibrium constraints: theory, applications and numerical results*, volume 28. Springer Science & Business Media, 2013.

[87] Ch. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[88] J. Petersson. On stiffness maximization of variable thickness sheet with unilateral contact. *Quarterly of applied mathematics*, 54(3):541–550, 1996.

[89] J. Petersson. A finite element analysis of optimal variable thickness sheets. *SIAM Journal on Numerical Analysis*, 36(6):1759–1778, 1999.

[90] J. Petersson and O. Sigmund. Slope constrained topology optimization. *International Journal for Numerical Methods in Engineering*, 41(8):1417–1434, 1998.

[91] B. D. Reddy. *Introductory functional analysis: with applications to boundary value problems and finite elements*. New York, 1998.

[92] J-F Rodrigues. *Obstacle problems in mathematical physics*. Elsevier, 1987.

[93] S. Rojas-Labanda and M. Stolpe. Benchmarking optimization solvers for structural topology optimization. *Structural and Multidisciplinary Optimization*, pages 1–21, 2015. DOI 10.1007/s00158-015-1250-z.

[94] G. I. N. Rozvany. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. *Structural and Multidisciplinary Optimization*, 21(2):90–108, 2001.

[95] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of computation*, 37(155):105–126, 1981.

[96] Y. Saad. Overview of krylov subspace methods with applications to control problems. 1989.

[97] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003.

[98] Y. Saad and M. H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Mathematics of Computation*, 44(170):417–424, 1985.

[99] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[100] D. Shanno. Who invented the interior-point method?

[101] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[102] O. Sigmund and J. Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16(1):68–75, 1998.

[103] J. Sokolowski and A. Zochowski. Topological derivative in shape optimization topological derivative in shape optimization. In *Encyclopedia of Optimization*, pages 3908–3918. Springer, 2009.

[104] R. Stainko. An adaptive multilevel approach to the minimal compliance problem in topology optimization. *Communications in Numerical Methods in Engineering*, 22(2):109–118, 2006.

[105] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1):281–309, 2001.

[106] K. Svanberg. The method of moving asymptotes- a new method for structural optimization. *International journal for numerical methods in engineering*, 24(2):359–373, 1987.

[107] K.C. Toh. Solving large scale semidefinite programs via an iterative solver on the augmented systems. *SIAM Journal on Optimization*, 14(3):670–698, 2004.

[108] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid.* Academic Press, 2000.

[109] J. A. Turner. *Application of domain decomposition methods to problems in topology optimisation.* PhD thesis, University of Birmingham, 2015.

[110] M. Vallejos. MGOPT with gradient projection method for solving bilinear elliptic optimal control problems. *Computing*, 87(1-2):21–33, 2010.

[111] M. Vallejos and A. Borzì. Multigrid optimization methods for linear and bilinear elliptic optimal control problems. *Computing*, 82(1):31–52, 2008.

[112] M. Y. Wang, X. Wang, and D. Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1):227–246, 2003.

[113] S. Wang, E. De Sturler, and G. H. Paulino. Large-scale topology optimization using preconditioned krylov subspace methods with recycling. *International Journal for Numerical Methods in Engineering*, 69:2441–2468, 2007.

[114] P. Wesseling. *An introduction to multigrid methods*. Pure and Applied Mathematics (New York). John Wiley & Sons, Ltd., Chichester, 1992.

[115] P. Wesseling. Introduction to multigrid methods. Technical report, DTIC Document, 1995.

[116] G. Wittum. On the convergence of multi-grid methods with transforming smoothers. *Numerische Mathematik*, 57(1):15–38, 1990.

[117] M. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American mathematical society*, 42(1):39–56, 2005.

[118] S. J. Wright and J. Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[119] D. M. Young. *Iterative solution of large linear systems*. Elsevier, 2014.

[120] D. M. Young and K. C. Jea. Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra and its applications*, 34:159–194, 1980.

[121] J. Zowe, M. Koćvara, and M. P. Bendsøe. Free material optimization via mathematical programming. *Mathematical Programming*, 79(1-3):445–466, 1997.