MULTIGRID BARRIER AND PENALTY METHODS FOR LARGE SCALE TOPOLOGY OPTIMIZATION OF SOLID STRUCTURES

by

ALEXANDER BRUNE

A thesis submitted to
University of Birmingham
for the degree of
PHD OF APPLIED MATHEMATICS

School of Mathematics
College of Engineering and Physical Sciences
University of Birmingham
February 2022

Abstract

We propose two algorithms for solving minimum compliance topology optimization problems defined on finite element meshes with several million elements, where the design geometry is parameterized on the discretized problem domain by an element-wise constant density field and we use the variable thickness sheet formulation to map the density to the material stiffness. The first method is an interior point (IP) method and the second follows the penalty-barrier multiplier (PBM) or nonlinear rescaling framework. To solve the linear systems arising in each optimization iteration, we use a multigrid-preconditioned Krylov solver. We employ a reformulation of the linear system to obtain a symmetric positive definite matrix that is amenable to standard multigrid transfer operators. We test the performance of both our algorithms on a wide range of numerical examples, comparing their performance to each other and to that of the well-established optimality criteria (OC) method. Our PBM algorithm proves to be more robust and efficient than both the IP and OC method.

We then extend our approach to problems defined on unstructured meshes, which necessitates switching to an algebraic multigrid preconditioner. Using the (adaptive) smoothed aggregation method of Vaněk, Mandel, and Brezina, we propose and test different non-standard setup strategies for the multigrid transfer operators in order to identify the most efficient one for our type of problem.

The PBM method is applied to the dual of the compliance minimization problem, which permits an easy integration of unilateral contact constraints. We include examples featuring such constraints in our numerical experiments, both for problems on uniform structured meshes and on unstructured meshes.

ACKNOWLEDGEMENTS

Mathematicians, according to people who are not mathematicians, are obsessed with deriving detailed definitions and applying rigorous analysis. The common assumption is that this purported obsession is not limited to mathematics but extends to everything else. One might think, therefore, that I would relish the task of precisely quantifying the many ways in which people have directly or indirectly influenced my work or supported me. This is incorrect. Writing acknowledgements is nothing like mathematics. There are no referees to flag missing references to a particular colleague, friend or family member before publication; one cannot leave missed acknowledgements for future work or hope that other researchers will pick up the slack. Nevertheless, I shall attempt to treat this problem like a mathematical one. I will limit myself to a few essential individual acknowledgements, beyond which I will define comprehensive sets of people who have been relevant to my PhD at the University of Birmingham. Specific examples of elements of these sets go beyond the scope of this thesis and are left as an exercise for the reader.

First and foremost, I thank my supervisor Michal Kočvara for his support, guidance and reassurance. Whether it was fixing bugs in my code or fences in your garden, I truly enjoyed our collaboration.

I want to express my appreciation for the postgraduate community at the School of Mathematics: all the wonderful people I have had the pleasure to share an office (building) with, the sociable atmosphere, the inclusivity and the personal connections I have made. This community set my PhD experience at this university apart from what I might have found anywhere else.

Outside of mathematics, I thank all of my friends and family for enduring my never-ending laments about doing a PhD. Thank you to my mother for always showing confidence that I was doing my best, and to my father for kindly nudging me back on track when I really wasn’t. Thank you to my sister for keeping me grounded and reminding me that there are worse things than writing a thesis – namely, high school.

The research presented in this thesis was funded by the University of Birmingham – and to some extent, although unofficially, by Erna Gründl, whose constant support over the years has made the path that led me here much smoother.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

Scalars, vectors and matrices

$a \in \mathsf{R}$	Real scalars (italicized)
$\boldsymbol{a} \in \mathsf{R}^n$	n–dimensional real vectors (bold)
$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$	Compound vector, equivalent to $\left(\boldsymbol{a}^\mathsf{T}, \boldsymbol{b}^\mathsf{T}, \boldsymbol{c}^\mathsf{T}\right)^\mathsf{T}$
$\boldsymbol{a}_\mathsf{I} = (a_i)_{i\in\mathsf{I}} \in \mathsf{R}^{\lvert\mathsf{I}\rvert}$	Set-indexing notation for vectors $\boldsymbol{a} \in \mathsf{R}^n$ and sets $\mathbf{I} \subset \{1, \ldots, n\}$
$\boldsymbol{a} \leq \boldsymbol{b}$	$a_i \leq b_i$ for all i

$\mathrm{A} \in \mathsf{R}^{n\times n}$	Matrices (uppercase, upright)
$\mathrm{A}_{\mathbf{IJ}} = [a_{ij}]_{i\in\mathsf{I}, j\in\mathsf{J}} \in \mathsf{R}^{\lvert\mathsf{I}\rvert\times\lvert\mathsf{J}\rvert}$	Set-indexing notation for matrices
$\mathrm{A}_{\mathbf{I}:}, \mathrm{A}_{:\mathbf{J}}$	Like above, with $\mathbf{J}$, $\mathbf{I} = \{1, \ldots, n\}$, resp.

$\mathrm{A} \quad 0$	A is positive definite, i.e. $\boldsymbol{d}^\mathsf{T}\mathrm{A}\boldsymbol{d} > 0$ for all $\boldsymbol{d} \in \mathsf{R}^n \setminus \{0\}$
$\mathrm{A} \quad 0$	A is positive semi-definite, i.e. $\boldsymbol{d}^\mathsf{T}\mathrm{A}\boldsymbol{d} \geq 0$ for all $\boldsymbol{d} \in \mathsf{R}^n$

Canonical vectors and matrices

$\mathbf{1} = (1, \ldots, 1)^\mathsf{T}$	Vector of ones of appropriate dimension
$\mathbf{0}$	Zero vector
$\mathrm{V} = \mathrm{diag}\{\boldsymbol{v}\}$	Diagonal matrix produced by a vector
$\mathrm{I} = \mathrm{diag}\{\mathbf{1}\}$	Identity matrix of appropriate dimension
0	Zero matrix of appropriate dimension

CHAPTER 1

INTRODUCTION

Perhaps the earliest work that can be credited as a contribution to the field of *structural optimization* is the article by Michell [87]. In it, Michell identified the shape of truss structures to support a given set of external forces using a minimal amount of material, without exceeding prescribed stress limits in any of the bars. This is but one of many possible examples of a structural optimization problem: one aims to find the specifications of a physical structure that is optimal with respect to a certain objective, such as the amount of material used or the overall stiffness, while at the same time satisfying a set of constraints. The latter are given by the governing equations of the mathematical model of the underlying physical scenario and any other quantifiable restrictions on the structure. The problem also needs to define parameters that we can adjust in order to optimize the structure, which are called *design variables*. They typically parameterize the geometry of the structure, but can comprise other properties, such as material constants. The choice of parameterization leads to different types of structural optimization problems. If the design variables specify the dimensions of pre-defined structural features, such as the width, length or cross-sectional area of ribs or trusses, for example, we speak of *sizing optimization*. Allowing for a more general variation of the boundary of an initial design brings us into the territory of *shape optimization*. If the design variables determine the structure's geometry in an even more flexible manner, by prescribing material distribution, for instance, then we are dealing with *topology optimization*. The contributions of this thesis fall into this last category. The distinctions between the sub-disciplines of structural optimization are not always clear cut and the brief definitions given here may by many be seen, rightfully, as oversimplifications. For a comprehensive introduction to structural optimization, see for example [62], or the much-cited book by Bendsøe and Sigmund [22], a standard reference, in particular for topology optimization.

After the article by Michell, it was not until the 60s and 70s that academic interest in this subject picked up. These early contributions were predominantly analytical treatments of structural optimization problems, see for example [63]. Research efforts began to focus on numerical methods for topology optimization on discretized design domains in the 80s, e.g. [21], which also saw the introduction of the popular *solid isotropic material with penalization* (SIMP) method for the approximation of material behaviour in designs with voids [18].

Promising research in algorithms and increased availability of computing power prompted the development of commercial software for structural optimization in the 90s. Topology optimization, in particular, has gained importance for practical applications in light of recent advances in additive manufacturing, which facilitates the realization of complex optimal designs, see for example [28, 41]. The scope of topology optimization has greatly expanded in recent decades. Various types of structural responses have been successfully incorporated as objectives or constraints, and problems are taken from a wide range of areas beyond solid mechanics, such as fluid dynamics, thermodynamics, or multi-physics, among others [22]. While there is naturally still some discrepancy between academic research and widespread commercial applications, topology optimization is nowadays utilized in many areas of engineering, in particular in the automotive industry [37] and the aircraft industry [137].

The specific parameterization of the design topology has a profound influence on the resulting mathematical problem, its properties and the efficiency of the methods used to solve it. The most common approach is to model the material distribution by *density* values ρ_i assigned to each element in the mesh that is used for the numerical analysis of the structural response, usually via the finite element (FE) method [94]. Discretizing the design in this way gives us a finite-dimensional problem. We further relax it so that instead of enforcing a strict distinction between void ($\rho_i = 0$) and solid ($\rho_i = 1$) elements, we allow density values within the range $[0, 1]$, so that the problem becomes continuous. In order to still obtain an optimal solution that approximates a $0 - 1$ design, the SIMP method [18, 19] uses a nonlinear function to map element density to material constants so that intermediate density values are penalized. If we instead prescribe a linear relation between density and stiffness, we arrive at the *variable thickness sheet* (VTS) problem [110, 97, 22]. In two dimensions, the VTS formulation models a sheet under plane stress with varying thickness parameterized by the design variables. This physical interpretation does not extend to three dimensions, in which case the VTS problem can be seen as a further relaxation of the SIMP-based topology optimization problem. Although the SIMP method is much more prevalent in academic as well as industrial applications, the VTS formulation permits the use of more efficient optimization algorithms. We will consider the latter in this thesis and address limitations in regards to an extension to the SIMP formulation.

Other approaches than the above should be mentioned for the sake of completeness. One can adhere to the element-based paradigm of the SIMP or VTS approach, but assign more than a single variable per element, which are typically related to the material constants in some way. Examples are homogenization methods, see for example [21], or free material optimization, e.g. [138]. Going along a different route entirely, level-set methods have also become popular for topology optimization, see for example the review by Dijk et al. [42], or the much-cited papers by Allaire, Jouve, and Toader [7] and Wang, Wang, and Guo [127].

In spite of the technological advance in computer hardware seen since the early days of topology optimization, the large computational cost remains one of the main challenges in its application. Obtaining a detailed design requires a high resolution of the discretization of the problem domain, which corresponds directly to the number of variables in the structural analysis and optimization problem. The linear systems that arise in each iteration of the optimization algorithm are accordingly very large, so that constructing and, in particular, solving them constitutes a significant part of the overall computational work. A prominent example of the resource demands of large-scale topology optimization is the article by Aage et al. [3], in which the optimal design for an aeroplane wing, discretized by more than one billion finite elements, was computed over several days using 8000 cores on a supercomputer.

A considerable amount of research has been dedicated to reducing the computational cost of topology optimization. Tackling the problem on the implementation level, different authors have investigated the use of parallel computing [25, 125, 46, 2, 1] and graphics card programming [116, 134, 39]. Others have focussed on decreasing the size of the problem. For example, Nguyen et al. proposed a multi-resolution approach [91, 92], using a coarser mesh for the structural analysis than for the density. Wang, Kang, and He also used separate meshes in [129], refining the mesh for the nodal density values adaptively only where a higher resolution was required. In [83], Lazarov studied the applicability of multi-scale FE methods to topology optimization.

Choosing the right technique to solve the linear systems in each optimization iteration is another point that is critical to improving efficiency. In [10], Amir, Bendsøe, and Sigmund reused information from previous iterations to form a reduced basis for the solution, employing a direct solver for the resulting reduced system. For large-scale problems, iterative solvers are generally more efficient, especially if an exact solution is not required [114, 131]. Among these, the class of Krylov solvers [114] is of particular importance. These methods find an approximate solution to a linear system such that it minimizes a specific functional on a subspace whose dimension increases with each solver iteration. For symmetric systems, the conjugate gradient (CG) [64] and the minimal residual (MINRES) [95] method are two particularly successful Krylov solvers. While the former is one of the most widely used iterative methods for symmetric positive definite systems, the latter can also solve indefinite (even singular) symmetric systems. Wang, de Sturler, and Paulino proposed an approach to recycle Krylov subspaces in a MINRES solver for topology optimization in [128]. Amir, Stolpe, and Sigmund used approximate solutions to the linear systems obtained from the CG method in [11], where the accuracy of the solution was adapted to the progress of the optimization.

A crucial component of any performant Krylov solver is preconditioning, which improves the spectral properties of the linear system, drastically reducing the number of solver iterations required to achieve a certain precision. The system matrix arising from the FE analysis typically becomes more ill-conditioned as the mesh is refined and the system size increases. An optimal preconditioner would ensure that the number of solver iterations needed to reach a given accuracy are not only small but independent of the system size. Multigrid methods [29, 61, 33] have become widely used as preconditioners, for the reason that they have the aforementioned property [130], at least when used on systems derived from discretized partial differential equations. The multigrid paradigm is based on a hierarchy of meshes – or grids – with decreasing resolution. The system matrix and residual of the linear system are projected onto these coarser grids and the approximate solution for the finest grid – the one on which the problem is originally defined – is recursively improved on these lower levels. This approach leads to a mesh independent convergence behaviour. Although multigrid methods are themselves linear solvers, they are often found to be more effective when used in combination with Krylov solvers. A multigrid-preconditioned CG method with an adaptive solver tolerance similar to [11] was used in [9]. Combining this with the reduced basis approach from [10], Amir proposed a recycling of the coarse-grid system matrices for subsequent optimization iterations [8]. Note that the idea of reusing a preconditioner based on a previous design iterate had been employed earlier by Kirsch, Kočvara, and Zowe in [74]. Multigrid-preconditioned CG solvers were also used in [1, 3, 80], among others. Others have used multigrid methods as linear solvers for topology optimization problems, rather than just as preconditioners. Maar and Schulz used a multigrid solver in an algorithm for compliance minimization [85] and Stainko proposed a multigrid solver for systems arising in problems with stress constraints [119]. In both cases, the linear systems were indefinite saddle-point systems, which required a much more involved multigrid setup compared to the standard method for positive definite systems.

The basic multigrid methodology, also called geometric multigrid (GMG), assumes a grid hierarchy obtained through simple, uniform mesh refinement, so that the projection operators can be defined by simple interpolation. This requires the top-level mesh to be highly structured and regular. If such a grid hierarchy cannot be easily obtained, for example because the design domain is not defined by a uniform mesh, we need to use algebraic multigrid (AMG) methods [47], which determine coarse grids and projection operators based solely on the system matrix entries. For certain types of problems, it can even make sense to use them when GMG methods would also be applicable. In [3], a hybrid GMG-AMG method was used as the algebraic approach was more efficient on coarser levels. Similar results were presented in [96], which compared the performance of GMG and AMG methods for a range of topology optimization problems.

Another aspect of large-scale topology optimization that has not received as much attention as the points discussed above is the choice of the optimization method. While established general-purpose methods like the interior point (IP) have sometimes been used for topology optimization, see e.g. [85, 70, 119, 120], for problems based on the SIMP approach, the most common methods are the method of moving asymptotes (MMA) [121] and the optimality criteria (OC) method [111, 136]. In a benchmarking paper by Rojas-Labanda and Stolpe, these were compared to a range of methods implemented in general-purpose optimization libraries [109]. In terms of the number of required iterations, the latter very often performed better, however, they lost some or all of that advantage when it came to computational time. A plausible explanation is that the OC method and MMA are first-order methods, using only function and derivative values at each iteration, whereas the other methods are second-order methods and also make use of Hessian information (or BFGS [93] approximations thereof). While second-order methods are known to generally converge in fewer iterations than first-order methods, the second-order information can make them more computationally expensive.

The results from the benchmarking paper do not fully extend to the VTS formulation. To show this will be the first contribution of this thesis. We propose two second-order methods for the minimum compliance VTS problem. The first is a primal-dual IP method [52, 132]. It is based on previous contributions by Kočvara and Mohammed [80] and Jarre, Kočvara, and Zowe [71]. Maar and Schulz also proposed an IP method for (SIMP-based) compliance minimization in [85], which however differs strongly from ours in how the linear systems are solved. Starting from the asymmetric, indefinite system given by the primal-dual IP approach, we reduce it in order to obtain an equivalent symmetric positive-definite system. We solve this by a multigrid-preconditioned Krylov solver. In contrast, the IP algorithm in [85] involved solving an indefinite system by a multigrid method designed specifically for such systems.

Our second method is a nonlinear rescaling or penalty-barrier multiplier (PBM) method. Proposed by Polyak and Teboulle [103] and Ben-Tal and Zibulevsky [17], based on the modified barrier methods due to Polyak [100], this class of algorithms attempts to overcome the notorious ill-conditioning of the linear systems observed in IP methods as one approaches the optimal solution. It has been successfully used for large-scale optimization, for example in [77] and the optimization software package Pennon [76], and for topology optimization in [79]. We apply the PBM approach not to the VTS problem itself, but to its dual, thus avoiding certain complications inherent in the primal formulation. To solve the linear systems, we perform a reduction very similar to that used in our IP method and solve it by the same multigrid-preconditioned Krylov solver.

We compare the performance of our two methods to that of the OC method, and reach the clear verdict that the latter is not competitive for this problem. We further compare the IP and PBM method on large-scale problems with more than a million finite elements. In this case, the PBM method proves to be both more robust and efficient than the IP method. As a bonus, the dual VTS problem allows for an easy integration of unilateral contact constraints. We include large-scale examples of problems featuring such constraints, solved by the PBM method.

The second contribution of this thesis is an investigation of different multigrid setup strategies. Peetz and Elbanna recently showed that AMG methods can be preferable to GMG methods as preconditioners in topology optimization, even on structured meshes, due to the material anisotropy that comes with the high-contrast density distribution seen in SIMP-based optimal designs [96]. While the computational overhead of the AMG is considerably larger than that of the GMG method, this is in some cases outweighed by the reduction in solver iterations. As we will show, the same cannot be said for the VTS problem, presumably due to smaller local density variations. We therefore focus on finding the most efficient AMG setup technique for problems on unstructured meshes, where GMG is not an option. The AMG approach we use in this thesis is the smoothed aggregation method introduced by Vaněk, Mandel, and Brezina [124], which has gained popularity due to its effectiveness [47].

The thesis is structured as follows: Chapter 2 covers the mathematical prerequisites. Section 2.1 introduces basic concepts of nonlinear optimization, and the two general purpose methods we will later apply to our specific optimization problem.

Section 2.2 very briefly outlines the finite element method for linear elasticity and its extension to unilateral contact constraints. The compliance minimization problem for the variable thickness sheet is reviewed in some detail in Section 2.3, where we also include a uniqueness result for the case of a non-zero lower bound on the element densities. We finish Section 2.3 with a description of the optimality criteria method for the VTS problem. In Section 2.4, we summarize relevant concepts and results for Krylov solvers and multigrid methods, both geometric and algebraic. Chapter 3 constitutes the main part of this thesis. A detailed description of our IP and PBM algorithm for the VTS minimum compliance problem and the procedure for solving the linear systems is given in Section 3.1. In Section 3.2, we study and compare the performance of the IP, PBM and OC method on a wide range of problems. The IP and PBM algorithms are applied to large-scale examples. In order to solve problems defined on unstructured meshes as efficiently as possible, we propose and compare different setup strategies for the smoothed aggregation AMG preconditioner, which we then use to solve unstructured large-scale problems by the PBM method. Section 3.3 addresses limitations of our optimization algorithm when it comes to an extension to the SIMP formulation. In Chapter 4, we summarize our results and draw conclusions to guide future research.

CHAPTER 2

BACKGROUND

This chapter introduces the theory required to understand the problem we wish to solve and the methods we use to do so. Since it is first and foremost an optimization problem, we begin by laying out basic concepts and results in nonlinear optimization. We also describe two general solution algorithms, the interior point and penalty-barrier multiplier method, which we will later apply to our problem. To even formulate this problem, which originally comes from the field of solid mechanics, we need to cover a few fundamentals of linear elasticity as well as the finite element method, used to numerically approximate the behaviour of solid structures, which we do in Section 2.2. Building on this, we can derive the minimum compliance topology optimization problem in Section 2.3, which also includes a third optimization algorithm, the optimality criteria method, which is specific to this problem. Finally, Section 2.4 focuses on the task of solving linear systems which inherently arise in each iteration of an optimization algorithm. To do this efficiently is critical for large-scale applications, where the number of degrees of freedom easily surpasses a million. We employ Krylov solvers and multigrid methods, which are outlined in Section 2.4.

As we cover a lot of ground in this chapter, each section apart from Section 2.3 is limited to a brief introduction and overview. When discussing algorithms, we prioritize an accessible presentation over a thorough analysis. The main convergence properties are generally included, especially where they are relevant to the interpretation of the performance of our implementation in practice, but we refer to the literature listed in each section for details.

2.1 Optimization

The mathematical problem that lies at the centre of this thesis is a constrained nonlinear optimization problem. From it arise both questions of a theoretical nature, like those about the existence and uniqueness of its solutions, and more practical questions, such as how to efficiently solve the problem at a large scale. It therefore makes sense to begin with a section which covers the necessary basics of nonlinear optimization. First of all, we will motivate and define the general form of nonlinear optimization problems with constraints, after which we will go over some standard optimality conditions in Section 2.1.1. For proofs and further reading, one may consult, for example, the comprehensive reference by Nocedal and Wright [93]. We complement these first fundamentals by some results which are particularly relevant to the class of convex optimization problems in Section 2.1.2, where we draw mainly from the book by Boyd and Vandenberghe [27]. Sections 2.1.3 and 2.1.4 describe two specific methods for solving optimization problems, the interior point method and the penalty-barrier multiplier – or nonlinear rescaling – method, respectively. For the former, we largely follow the book by Wright [132] and the comprehensive review article by Forsgren, Gill, and Wright [52]. For the latter, a more detailed list of references is given in Section 2.1.4. There is a third optimization method that we employ in this thesis, however, it is designed specifically for the compliance minimization problem. We defer a description of this method to Section 2.3.3, after the discussion of minimum compliance optimization.

In order to somewhat simplify the following treatment of nonlinear optimization, all functions are assumed to be twice continuously differentiable. This is also generally true for all functions introduced in this thesis, unless otherwise specified.

Now, assume we want to find a point $\boldsymbol{x} \in \mathsf{R}^n$ at which a function $f : \mathsf{R}^n \to \mathsf{R}$ attains the smallest possible value. Furthermore, let us require that this point satisfy certain criteria which limit the possible choices for $\boldsymbol{x}$ to a subset of R^n. We assume that we can describe this subset by means of real-valued functions, consecutively labelled, which define a set of inequalities $g_1(\boldsymbol{x}) \leq 0, g_2(\boldsymbol{x}) \leq 0, \ldots,$ and equations $h_1(\boldsymbol{x}) = 0, h_2(\boldsymbol{x}) = 0, \ldots$, which we refer to as *inequality* and *equality constraints*, respectively. Let I and E denote the corresponding set of label indices. The set

$$\mathrm{X} := \{\boldsymbol{x} \in \mathsf{R}^n \mid g_i(\boldsymbol{x}) \leq 0 \text{ for all } i \in \mathrm{I},\ h_j(\boldsymbol{x}) = 0 \text{ for all } j \in \mathrm{E}\}$$

defined by the constraints is called *feasible set* and a point $\boldsymbol{x}$ is called *feasible* if $\boldsymbol{x} \in \mathrm{X}$. Furthermore, if that point satisfies $g_i(\boldsymbol{x}) < 0$ for all $i \in \mathrm{I}$, it is called *strictly feasible* (with respect to the inequality constraints). We will assume throughout this thesis that the inequality constraints are topologically consistent, meaning that the set of strictly feasible points coincides with the relative interior of X. A constrained nonlinear optimization problem is the task of finding the smallest value of the function $f(\boldsymbol{x})$, called the *objective function*, over the entire feasible set X and is commonly formulated as a *nonlinear program* (NLP) [93, (12.1)]:

$$\min_{\boldsymbol{x} \in \mathsf{R}^n} \quad f(\boldsymbol{x}) \tag{2.1a}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leq 0\,, \quad \forall i \in \mathrm{I} \tag{2.1b}$$

$$h_j(\boldsymbol{x}) = 0\,, \quad \forall j \in \mathrm{E}\,. \tag{2.1c}$$

For a slightly simplified formulation, let us introduce some notation. When we deal with a set of consecutively labelled functions, such as $g_i(\boldsymbol{x})$ for all $i \in \mathrm{I}$, we will often denote these by a vector-valued function $\boldsymbol{g}(\boldsymbol{x}) := (g_i(\boldsymbol{x}))_{i \in \mathrm{I}}$. We extend the inequality sign to sets of inequalities, interpreted component-wise so that we can write $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$, where $\mathbf{0}$ is a vector of all zeros, instead of $g_i(\boldsymbol{x}) \leq 0$ for all $i \in \mathrm{I}$.

We can therefore write (2.1) as

$$\min_{\boldsymbol{x}\in\mathrm{R}^n} \quad f(\boldsymbol{x}) \tag{2.2a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}\,, \tag{2.2b}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}\,. \tag{2.2c}$$

As is commonly done, and without loss of generality, we defined the nonlinear program as a minimization problem. By changing the sign of the objective function, we obtain an equivalent maximization problem. It is straightforward to adjust any standard definition or theorem for a maximization problem.

The very first question to ask in the discussion of the nonlinear program (2.1) is what exactly counts as a solution. The answer is given by the following list of definitions, which can be found in [93, (p. 6 and 306)], for example. In it, we use $\mathrm{N}(\boldsymbol{x})$ to denote a neighbourhood of a point $\boldsymbol{x}$, i.e. an open, bounded set which contains the point itself.

Definition 2.1.1. Let $\boldsymbol{x}^* \in \mathrm{X}$ be a feasible point for (2.1). This point is called

(i) a *global optimum*, *global minimum*, or *global solution* if

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathrm{X}\,. \tag{2.3}$$

(ii) a *local optimum*, *local minimum*, or *local solution* if there exists a neighbourhood $\mathrm{N}(\boldsymbol{x}^*)$ such that

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathrm{N}(\boldsymbol{x}^*) \cap \mathrm{X}\,. \tag{2.4}$$

(iii) a *strict local* or *strict global minimum*, respectively, if (2.3) or (2.4) holds with strict inequality when $\boldsymbol{x} \,/\!= \boldsymbol{x}^*$.

(iv) an *isolated local optimum*, *isolated local minimum*, or *isolated local solution*

if there is a neighbourhood $N(\boldsymbol{x}^*)$ such that $\boldsymbol{x}^*$ is the only local solution in $N(\boldsymbol{x}^*) \cap X$.

While a solution is defined by comparison to other feasible points, there are conditions which allow us to identify or rule out a point $\boldsymbol{x} \in X$ as a solution using only information about the point itself. More specifically, these conditions for optimality require the first and second order derivatives of the objective function and constraint functions.

For our purposes, it is convenient to treat the gradient of a function $f(\boldsymbol{x})$ as a column vector, i.e. $\nabla f(\boldsymbol{x}) := (\partial f/\partial x_1, \dots, \partial f/\partial x_n)^T$. The Jacobian of a vector-valued function $\boldsymbol{g}(\boldsymbol{x}) \in R^{|I|}$ is composed of the gradients of the components arranged in columns, i.e. $\nabla \boldsymbol{g}(\boldsymbol{x}) := \nabla g_1(\boldsymbol{x}), \dots, \nabla g_{|I|}(\boldsymbol{x})$. With the notation out of the way, we will now turn towards some necessary and sufficient conditions for optimality.

2.1.1 Optimality Conditions

For an unconstrained NLP, i.e. one where $X = R^n$, one criterion that a point $\boldsymbol{x}^* \in R^n$ must satisfy to be a local optimum is that the gradient of the objective function vanishes [93, Theorem 2.2], i.e.

$$\nabla f(\boldsymbol{x}^*) = \mathbf{0}. \tag{2.5}$$

By using the Taylor expansion of $f(\boldsymbol{x})$ at $\boldsymbol{x}^*$, one can show that a non-zero gradient means that there is a direction in which one can move from $\boldsymbol{x}^*$ along which the objective function immediately decreases. Therefore, $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ is a necessary optimality condition, however, it does not guarantee that $\boldsymbol{x}^*$ is optimal. By also requiring that the Hessian of f is positive definite at $\boldsymbol{x}^*$, i.e.

$$\nabla f(\boldsymbol{x}^*) = \mathbf{0} \quad \text{and} \quad \nabla^2 f(\boldsymbol{x}^*) \quad 0, \tag{2.6}$$

we obtain a sufficient condition for local optimality [93, (Theorem 2.4)]. Indeed, by Taylor's theorem, the positive curvature of f at $\boldsymbol{x}^*$ tells us that f increases as soon as one moves away from $\boldsymbol{x}^*$ (and stays within a certain neighbourhood), making this point a strict local optimum. Due to the order of the derivatives involved, condition (2.5) is called a *first order* and (2.6) a *second order* optimality condition, respectively. Similar necessary and sufficient conditions exist for the general constrained NLP (2.1). But while (2.5) and (2.6) tell us if $f(\boldsymbol{x}^*)$ is (potentially) the smallest objective function value within a neighbourhood of $\boldsymbol{x}^*$, we are now only interested in neighbouring points which are also feasible.

In this context, we should first of all introduce the concept of the *active set*, which indicates which part of the boundary of the feasible set a point lies on.

Definition 2.1.2. Let $\boldsymbol{x} \in X$ be a feasible point for (2.1). The *active set* at $\boldsymbol{x}$ is the set of indices of all inequality constraints which are satisfied with equality at $\boldsymbol{x}$ and is denoted by

$$A(\boldsymbol{x}) := \{i \in I \mid g_i(\boldsymbol{x}) = 0\} .$$

A constraint is called *active* at $\boldsymbol{x}$ if $g_i(\boldsymbol{x}) = 0$, and inactive if $g_i(\boldsymbol{x}) < 0$.

We deviate slightly from [93, (Definition 12.1)], in that we do not include the index set of the equality constraints in A.

The optimality conditions for (2.1) must account for the geometry of the boundary of the feasible set. In particular, we need a way to identify the directions in which we can move from the point $\boldsymbol{x}^*$ and still stay within X. The standard optimality conditions for the constrained NLP make use of the Jacobian of the constraints to characterize "first-order feasible directions" [93]. This is only valid as long as we know that the constraint functions describe the geometry of the feasible set well enough. This can be guaranteed if $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ satisfy a so-called *constraint qualification (CQ)*. One such CQ, which will be used in this thesis, is given in the following definition, see also [93, (Definition 12.4)]:

Definition 2.1.3 (Linear Independence Constraint Qualification (LICQ))**.** We say that a feasible point $\boldsymbol{x} \in X$ with active set $A(\boldsymbol{x})$ satisfies the LICQ if the gradients of all equality constraints and all active inequality constraints are linearly independent. In other words, if the matrix

$$\nabla \boldsymbol{h}, \nabla \boldsymbol{g}_{A(\boldsymbol{x})}]$$

has full rank, the LICQ holds at $\boldsymbol{x}$.

If the LICQ is satisfied at a feasible point, then we can obtain a first order necessary optimality condition [93, (Theorem 12.1)].

Theorem 2.1.4 (Karush-Kuhn-Tucker (KKT) Conditions)**.** *Let $\boldsymbol{x}^* \in X$ be a local solution for the NLP (2.1) at which the LICQ is satisfied. Then there exist vectors $\boldsymbol{\lambda}^* \in \mathrm{R}^{|I|}$ and $\boldsymbol{\mu}^* \in \mathrm{R}^{|E|}$ such that*

$$\nabla \boldsymbol{f}(\boldsymbol{x}^*) + \nabla \boldsymbol{g}(\boldsymbol{x}^*)\boldsymbol{\lambda}^* + \nabla \boldsymbol{h}(\boldsymbol{x}^*)\boldsymbol{\mu}^* = \mathbf{0}, \tag{2.7a}$$

$$\boldsymbol{h}(\boldsymbol{x}^*) = \mathbf{0}, \tag{2.7b}$$

$$\boldsymbol{g}(\boldsymbol{x}^*) \leq \mathbf{0}, \tag{2.7c}$$

$$\boldsymbol{\lambda}^* \geq \mathbf{0}, \tag{2.7d}$$

$$\boldsymbol{g}(\boldsymbol{x}^*)^{\mathsf{T}}\boldsymbol{\lambda}^* = 0. \tag{2.7e}$$

The vectors $\boldsymbol{\lambda}^* \in \mathrm{R}^{|I|}$ and $\boldsymbol{\mu}^* \in \mathrm{R}^{|E|}$ are called *Lagrange multipliers*. Lines (2.7c–2.7e) comprise the so-called *complementarity conditions*. They imply that $\lambda_i^* = 0$ for any $i \in \mathbf{I}$ with $g_i(\boldsymbol{x}^*) < 0$ and, conversely, if $\lambda_i^* > 0$, then $g_i(\boldsymbol{x}^*) = 0 \Leftrightarrow i \in A(\boldsymbol{x}^*)$. We say that *strict complementarity* holds if $\lambda_i^* > 0$ for all $i \in A(\boldsymbol{x}^*)$. Note that, due to (2.7c) and (2.7d), (2.7e) may also be written as $g_i(\boldsymbol{x}^*)\lambda_i^* = 0$ for all $i \in \mathbf{I}$.

Remark 2.1.5. Any point $\boldsymbol{x}$ that satisfies the KKT conditions, regardless of whether or not it is a local optimum, is called a *KKT point* and together with the associated Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ forms a *KKT triple* $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$.

At first glance, the KKT conditions look very different from the simple first order condition (2.5). In order to highlight some similarities, we can define a function, called the *Lagrangian* [93, (12.33)],

$$L(\boldsymbol{x}, \lambda, \boldsymbol{\mu}) := f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^\top \lambda + \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\mu} \,. \tag{2.8}$$

The first KKT condition, (2.7a), is now equivalent to $\nabla L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{0}$, cf. (2.5). The second and third conditions are simply the requirement that $\boldsymbol{x}^*$ be a feasible point.

We can also formulate a second order sufficient optimality condition. It is similar to (2.6) in that it involves a positive-definite Hessian. However, this definiteness is now limited to a set of feasible directions and is furthermore required of the Hessian of the Lagrange function, since this accounts for the influence of the constraint functions. We first define a set of directions which are related to the KKT conditions. We assume we are given a KKT point $\boldsymbol{x}^*$ with inequality constraint multipliers $\boldsymbol{\lambda}^*$. The *critical cone* at $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is defined as [93, (12.53)]

$$\mathcal{C}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) := \left\{ \boldsymbol{d} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \nabla h_j(\boldsymbol{x}^*)^\top \boldsymbol{d} = 0 \;\; \forall j \in \mathcal{E}\,, \\ \nabla g_i(\boldsymbol{x}^*)^\top \boldsymbol{d} \left\{ \begin{array}{ll} = 0 & \forall i \in \mathcal{A}(\boldsymbol{x}^*) \text{ with } \lambda_i > 0 \\ \leq 0 & \forall i \in \mathcal{A}(\boldsymbol{x}^*) \text{ with } \lambda_i = 0 \end{array} \right. \end{array} \right\} . \tag{2.9}$$

A succinct interpretation of $\mathcal{C}$ made in [93] is as the cone of feasible directions at $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ “for which it is not clear from first derivative information alone whether f will increase or decrease”. If we are given enough second derivative information relating to all directions in $\mathcal{C}$, this can help us verify, not only that $\boldsymbol{x}^*$ is a local optimum, but that it is a strict local optimum. This is detailed in the following theorem which describes sufficient optimality conditions based on the Hessian of the Lagrangian, see also [93, (Theorem 12.6)].

Theorem 2.1.6 (Second Order Sufficient Optimality Conditions). *Let* $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$

be a KKT triple for (2.1). Further assume that the Hessian of the Lagrangian is positive definite on the critical cone C*, i.e.*

$$\boldsymbol{d}^{\mathsf{T}}\, \nabla^2_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\, \boldsymbol{d} > 0 \quad \forall\ \boldsymbol{d} \in \mathrm{C}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*),\ \boldsymbol{d} \neq \mathbf{0}\,. \tag{2.10}$$

Then $\boldsymbol{x}^*$ *is a strict local optimum for (2.1).*

We have stated conditions that allow us to identify a solution for the general constrained NLP. Some of the above results are stronger for the class of convex optimization problems. The next subsection will introduce the basic definitions and some additional results for this class. We will also touch upon the concept of duality, which is of particular importance for convex problems.

2.1.2 Convexity and Duality

The following is a brief treatment of convex optimization and duality, mainly following [27]. We start with a quick revision of basic definitions and properties.

Definition 2.1.7 (Convex set [27, (p. 23)]). A set $X \subset \mathbb{R}^n$ is called *convex* if, for all $\boldsymbol{x}, \boldsymbol{y} \in X$ and $t \in [0, 1]$, the following holds:

$$(1 - t)\boldsymbol{x} + t\boldsymbol{y} \in X\,.$$

Definition 2.1.8 (Convex function [27, (3.1)]). A function $f : X \to \mathbb{R}$ defined on a convex set $X \subset \mathbb{R}^n$ is called *convex* if, for any $\boldsymbol{x}, \boldsymbol{y} \in X$ and for $t \in [0, 1]$,

$$(1 - t) f(\boldsymbol{x}) + t f(\boldsymbol{y}) \geq f((1 - t)\boldsymbol{x} + t\boldsymbol{y})\,.$$

If this holds with strict inequality for $0 < t < 1$ and $\boldsymbol{x} \neq \boldsymbol{y}$, then f is called *strictly convex*.

There are a number of criteria that imply convexity of a differentiable function, see for example [27, (p. 69ff.)], which are listed in the next lemma.

Lemma 2.1.9. *A function f is*

- *convex if and only if $\nabla f(\boldsymbol{x})^\top(\boldsymbol{y}-\boldsymbol{x}) \leq f(\boldsymbol{y}) - f(\boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in X$.*
- *strictly convex if and only if $\nabla f(\boldsymbol{x})^\top(\boldsymbol{y}-\boldsymbol{x}) < f(\boldsymbol{y}) - f(\boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in X$ with $\boldsymbol{x}$ /= $\boldsymbol{y}$.*
- *convex if and only if $\nabla^2 f(\boldsymbol{x})\ 0$ for all $\boldsymbol{x} \in X$.*
- *strictly convex if $\nabla^2 f(\boldsymbol{x})\ 0$ for all $\boldsymbol{x} \in X$.*

Typical examples of convex (but not strictly convex) functions are affine functions and quadratic functions $f(\boldsymbol{x}) = \boldsymbol{x}^\top A \boldsymbol{x}$ for some matrix $A\ 0$ but A ';/ 0, while the same function for $A\ 0$ is strictly convex.

We can apply the definitions of a convex function and a convex set to the NLP (2.1) to obtain a convex optimization problem, which has a number of useful properties.

Definition 2.1.10 (Convex optimization problem)**.** Regard the NLP (2.1). If the feasible set X is a convex set and the objective function f is convex on X, then (2.1) is called a *convex optimization problem* [67].

This leads to the question of when the feasible set is convex, which is answered by the following lemma, see also [27, (4.15)].

Lemma 2.1.11. *The feasible set X defined by (2.1b) and (2.1c) is convex if every inequality constraint function g_i, $i \in$ I is convex and every equality constraint function h_j, $j \in$ E is affine.*

The next result is one of the most elementary and important results for convex optimization, see also [27, (p. 138)].

Theorem 2.1.12. *If $\boldsymbol{x}^*$ is a local solution to a convex optimization problem, then* *it is also a global solution.*

With the above basics of convex optimization covered, we move on to the concept of duality. While this is not exclusive to convex problems, certain results which are relevant to this thesis are and we will therefore focus on the convex case for much of our discussion of duality.

Let us recall the definition (2.8) of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ for the NLP (2.1). We have already seen that it plays a role in the necessary optimality conditions, but its significance for the NLP goes beyond that. Let us define the extended real-valued function $p : \mathbb{R}^n \to \mathbb{R} \cap \{\infty\}$ as

$$\begin{aligned} p(\boldsymbol{x}) &:= \sup_{\boldsymbol{\lambda}\in\mathbb{R}^{|I|}, \boldsymbol{\lambda}\geq \mathbf{0}, \boldsymbol{\mu}\in\mathbb{R}^{|E|}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= \sup_{\boldsymbol{\lambda}\in\mathbb{R}^{|I|}, \boldsymbol{\lambda}\geq \mathbf{0}, \boldsymbol{\mu}\in\mathbb{R}^{|E|}} f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^\top\boldsymbol{\lambda} + \boldsymbol{h}(\boldsymbol{x})^\top\boldsymbol{\mu} \,. \end{aligned}$$

For a fixed $\boldsymbol{x}$ which is infeasible, so that $g_i(\boldsymbol{x}) > 0$ for at least one i or $h_j(\boldsymbol{x}) \neq 0$ for at least one j, we can make the Lagrangian arbitrarily large, which gives us $p(\boldsymbol{x}) = \infty$. For a feasible $\boldsymbol{x}$, on the other hand, the largest possible value of the weighted constraint terms is 0. We conclude that

$$p(\boldsymbol{x}) = \begin{cases} f(x) & \text{if } \boldsymbol{x} \in X\,, \\ \infty & \text{otherwise.} \end{cases}$$

Since we assume that the original NLP is feasible, the infimum of p is finite and the minimization problem (2.1) can also be written as an unconstrained minimization of $p(\boldsymbol{x})$:

$$\min(2.1) = \inf_{\boldsymbol{x}\in\mathbb{R}^n} p(\boldsymbol{x}) = \inf_{\boldsymbol{x}\in\mathbb{R}^n} \sup_{\boldsymbol{\lambda}\geq\mathbf{0},\, \boldsymbol{\mu}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})\,. \qquad (2.11)$$

We can obtain the *dual* of this problem [27, (5.16)] by swapping the inf and sup terms, giving us

$$\sup_{\boldsymbol{\lambda}\geq\mathbf{0},\, \boldsymbol{\mu}} \inf_{\boldsymbol{x}\in\mathbb{R}^n} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sup_{\boldsymbol{\lambda}\geq\mathbf{0},\, \boldsymbol{\mu}} d(\boldsymbol{\lambda}, \boldsymbol{\mu})\,. \qquad (2.12)$$

Here, we have introduced $d(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \inf_{\boldsymbol{x}\in\mathbb{R}^n} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, which is called the *dual function* [27, (p. 216)]. Accordingly, (2.12) is called the dual problem. We refer to (2.11) analogously as the *primal problem*, and to $p(\boldsymbol{x})$ as the *primal function*. In this context, we also call the optimization variables $\boldsymbol{x}$ the *primal variables* and the Lagrange multipliers $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ the *dual variables* [27, (p. 215)]. We say that $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is feasible for the dual problem if $\boldsymbol{\lambda} \geq \mathbf{0}$. An essential relationship between $p(\boldsymbol{x})$ and $d(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is that the latter gives a lower bound for the former and, conversely, the former is an upper bound for the latter. This is called *weak duality* [27, (p. 225)].

Theorem 2.1.13 (Weak duality). *For any $\boldsymbol{x}$ feasible for (2.11) and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ feasible for (2.12), we have that*

$$d(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq p(\boldsymbol{x}) \,.$$

In particular,

$$d^* := \sup_{\boldsymbol{\lambda}\geq\mathbf{0},\boldsymbol{\mu}} d(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq \inf_{\boldsymbol{x}} p(\boldsymbol{x}) =: p^* \,.$$

For primal and dual feasible $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, the difference between the primal and dual function values

$$\delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := p(\boldsymbol{x}) - d(\boldsymbol{\lambda}, \boldsymbol{\mu}) \geq 0$$

is called the *duality gap* [27, (p. 226)]. The case $\delta = 0$ is called *strong duality* [27, (ibid)]. A direct consequence of the weak duality theorem is that any feasible $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ at which strong duality holds is primal and dual optimal. More specifically, $\boldsymbol{x}$ is a global solution for the primal problem and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a global dual solution. Trivially, the optimal values of the primal and dual functions are equal, i.e. $p^* = d^*$, in this case. A point at which strong duality is obtained can also be characterized as a *saddle-point* of the Lagrangian, defined as follows:

Definition 2.1.14. Let $\bar{\boldsymbol{x}}$ be primal feasible and $\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}$ be dual feasible. The point $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ is called a *saddle-point* of the Lagrangian if it satisfies

$$L(\bar{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq L(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}) \leq L(\boldsymbol{x}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}) \,,$$

for all primal feasible $\boldsymbol{x}$ and dual feasible $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ [27, (p. 238)].

Strong duality does not generally hold at the optimal solution, not even for convex problems. However, by imposing the following condition on (2.1), we can guarantee that $p^* = d^*$, see [27, (Section 5.2.3)].

Definition 2.1.15. We say that the optimization problem (2.1) satisfies *Slater's condition* if there exists a $\boldsymbol{x} \in \mathrm{X}$ which is strictly feasible with respect to the inequality constraints, i.e.

$$\boldsymbol{g}(\boldsymbol{x}) < \mathbf{0} .$$

Theorem 2.1.16. *Let (2.1) be convex. If Slater's condition is satisfied, then strong duality holds at the solution of (2.1).*

For convex problems which have only inequality constraints, Slater's condition doubles as a constraint qualification. What is more, it is a "global" CQ, meaning that from the existence of a single strictly feasible point it follows that a constraint qualification is satisfied at all feasible points. This is very convenient since it lets us apply the KKT conditions at every point in the feasible set. But even without Slater's condition, the KKT conditions have special significance for convex problems: They are not only necessary – provided a CQ holds – but also sufficient optimality conditions, [27, (p. 244)].

Theorem 2.1.17. *Let the NLP (2.1) be convex. Any triple $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ that satisfies the KKT conditions (2.7) is primal and dual optimal with zero duality gap, i.e.*

$$\sup_{\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\mu}} d(\boldsymbol{\lambda}, \boldsymbol{\mu}) = d(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = p(\boldsymbol{x}^*) = \inf_{\boldsymbol{x}} p(\boldsymbol{x}) .$$

The takeaway from this section should be that, given a convex optimization problem which satisfies Slater's condition, solving it is equivalent to solving its dual. In some cases, the dual problem of a convex NLP can be formulated explicitly in the form of a standard NLP, in which the primal variables take the role of Lagrange multipliers. It can sometimes be easier to solve this dual problem than the original one. Some optimization algorithms provide the Lagrange multipliers alongside the optimal solution, which means we can retrieve the primal solution from the dual solution. Each of the methods described in the next two sections belongs to this class of algorithms.

2.1.3 Interior Point Methods

At the time of writing, *interior point* (IP) methods are no doubt among the most popular solution methods for nonlinear optimization problems. They were first developed as polynomial-in-time algorithms for linear programming in the 1980s, starting with the seminal paper by Karmarkar [72]. Soon after, it was shown in [55] that Karmarkar's approach is closely connected to classical barrier methods. The latter had already been studied quite extensively about twenty years earlier [51], but had not received a lot of attention since. This was because their potential for fast convergence was not unlocked until Karmarkar's contribution, which sparked renewed interest in the field. In the following years, the scope of IP methods was gradually extended beyond linear programming, first to quadratic and complementarity problems, then to general convex problems [90], and finally to nonlinear programming [13]. Among the various different approaches that have been developed and studied within the general IP framework, arguably the most popular are those grouped under the term *primal-dual* methods.

This section gives a very basic introduction to primal-dual interior point methods for nonlinear optimization, covering only what is necessary for the specific algorithm proposed in Chapter 3 for the compliance minimization problem. For a more comprehensive overview, we refer to [52] and the references therein. We also recommend the book by Wright [132], which is an excellent self-contained resource on primal-dual methods for linear programming and a good introduction to the IP concept in general, since many concepts originally derived for the linear case have found their way into nonlinear algorithms.

The basic idea behind IP methods is usually illustrated in one of two ways. The first is via logarithmic barrier functions, which provide a tool to approximate an inequality constrained problem by an unconstrained one. Given an NLP

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\{f(\boldsymbol{x}) \mid g_i(\boldsymbol{x}) \le 0 \;\forall i \in \mathrm{I}\}, \tag{2.13}$$

with only inequality constraints, we replace it by the unconstrained minimization problem

$$\min_{\boldsymbol{x}} \varphi_s(\boldsymbol{x}), \quad \text{where } \varphi_s(\boldsymbol{x}) := f(\boldsymbol{x}) - s \sum_{i\in\mathrm{I}} \ln(-g_i(\boldsymbol{x})). \tag{2.14}$$

The *logarithmic barrier function*[1] φ_s with *barrier parameter* $s > 0$ is only defined for strictly feasible points. When $\boldsymbol{x}$ approaches the boundary of the feasible set, i.e. as $g_i(\boldsymbol{x}) \to 0$ for one or more i, $\varphi_s(\boldsymbol{x})$ goes to infinity. We can therefore guarantee that the solution of the *barrier problem* (2.14) lies in the interior of X – hence the name "interior point method". Furthermore, by changing the value of s, we can specify how much φ_s deviates from f in the interior of X. As s goes to zero, $\varphi(\boldsymbol{x}) \to f(\boldsymbol{x})$ for all strictly feasible $\boldsymbol{x}$. If we denote by $\boldsymbol{x}(s)$ the solution of (2.14) for a given $s > 0$ and by $\boldsymbol{x}^*$ the minimum of the original constrained problem (2.13), one might intuit that $\boldsymbol{x}(s) \to \boldsymbol{x}^*$ as $s \to 0$. This supposition can be confirmed under relatively mild conditions, see for example [52, Theorem 3.10]. We have, in effect, replaced the inequality constrained problem (2.13) by a family of unconstrained problems, each member of which we can in theory solve by an unconstrained optimization algorithm, such as, for example, the Newton method [93].

Equality constraints can be incorporated into the approach by adding them to

[1] In some places in the literature, the term "barrier function" is used to refer not to the entire objective function φ_s but only to the penalization function $(-\infty, 0) \ni g \mapsto -\ln(-g)$. This seems to be a matter of preference. We chose our nomenclature for consistency with later sections.

(2.14). For the resulting system,

$$\min_{\boldsymbol{x}}\{\varphi_s(\boldsymbol{x}) \mid \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}\}, \tag{2.15}$$

the KKT conditions do not feature complementarity-related inequalities and are thus just a system of equations. As before, we can use the Newton method to obtain a solution, this time applied to the KKT system rather than just to $\nabla\varphi_s(\boldsymbol{x}) = \mathbf{0}$. Note that this approach, which is sometimes called the Lagrange-Newton method, is also an integral component of sequential quadratic programming (SQP) algorithms [93]. For this reason, the term "barrier-SQP methods" is sometimes used when applying it within the barrier method framework [52].

The IP methodology can alternatively be derived starting from the KKT conditions (2.7) for the NLP. Recall that the complementarity conditions (2.7c–2.7e) imply that $g_i(\boldsymbol{x})\lambda_i = 0$ for all $i \in \mathrm{I}$. We can relax these conditions by instead requiring $-g_i(\boldsymbol{x})\lambda_i = s$ for all $i \in \mathrm{I}$ for some $s > 0$. We choose the sign on the left-hand side so that the condition is viable for strictly feasible primal and dual points. Along with equations (2.7a) and (2.7b), we then get the *perturbed KKT conditions* [52, (p. 571)]

$$\begin{aligned} \nabla L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla f(\boldsymbol{x}) + \nabla \boldsymbol{g}(\boldsymbol{x})\boldsymbol{\lambda} + \nabla \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\mu} &= \mathbf{0}, \\ \Lambda\, \boldsymbol{g}(\boldsymbol{x}) &= -s\mathbf{1}, \\ \boldsymbol{h}(\boldsymbol{x}) &= \mathbf{0}, \end{aligned} \tag{2.16}$$

where $\Lambda = \mathrm{diag}\{\boldsymbol{\lambda}\}$ is a diagonal matrix of Lagrange multipliers and $\mathbf{1}$ is a vector of ones. We now have a nonlinear system of equations of size $n + |\mathrm{E}| + |\mathrm{I}|$ for the unknowns $\boldsymbol{x}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$, which we can solve by the Newton method. This is equivalent to the system we obtain from the barrier method approach described earlier on. Indeed, while the Lagrange multipliers $\boldsymbol{\lambda}$ do not appear in the KKT conditions for (2.15), by the substitution $\lambda_i := -s/g_i(\boldsymbol{x})$ for all $i \in \mathrm{I}$, we can obtain (2.16). In the Newton method, we solve a system obtained from a linearization of

(2.16) for the increments $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\mu})$. The linear system is given by

$$\begin{pmatrix} \nabla^2 L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) & \nabla\boldsymbol{g}(\boldsymbol{x}) & \nabla\boldsymbol{h}(\boldsymbol{x}) \\ \Lambda\nabla\boldsymbol{g}(\boldsymbol{x})^\top & \mathrm{G}(\boldsymbol{x}) & 0 \\ \nabla\boldsymbol{h}(\boldsymbol{x})^\top & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\mu} \end{pmatrix} = - \begin{pmatrix} \nabla L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \Lambda\boldsymbol{g}(\boldsymbol{x}) + s\mathbf{1} \\ \boldsymbol{h}(\boldsymbol{x}) \end{pmatrix}, \tag{2.17}$$

where $\mathrm{G}(\boldsymbol{x}) := \mathrm{diag}\{\boldsymbol{g}(\boldsymbol{x})\}$. Using a step size $\kappa \in (0, 1]$, we update the current solution by these increments:

$$(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leftarrow (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) + \kappa(\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\mu}) . \tag{2.18}$$

We then linearize (2.16) at this new iterate and repeat the procedure until some convergence criterion is satisfied.

Just as we did for the minimizer $\boldsymbol{x}(s)$ of (2.14), we can denote the solution of (2.16) as a function of s. The map $s \mapsto (\boldsymbol{x}(s), \boldsymbol{\lambda}(s), \boldsymbol{\mu}(s))$, with $s > 0$, is called the *central path*. Given a monotonically decreasing sequence of $s_k \to 0$, we could attempt to find the points $(\boldsymbol{x}(s_k), \boldsymbol{\lambda}(s_k), \boldsymbol{\mu}(s_k))$ on the central path, giving us a sequence that converges to a KKT point of (2.13). We could achieve this, for example, through solving (2.16) by the Newton method, using the previous solution $(\boldsymbol{x}(s_{k-1}), \boldsymbol{\lambda}(s_{k-1}), \boldsymbol{\mu}(s_{k-1}))$ as the initial guess. While this is indeed the general idea behind the most prevalent class of IP methods – aptly called *path-following* methods – a linchpin of their efficiency is the realization that it suffices to follow the path closely, rather than trace it exactly. As a consequence, we do not run the Newton method until convergence for each s_k, but only for a few – or even just a single – iteration. The Newton method is known for displaying quadratic convergence once the iterates get close enough to the exact solution, assuming that the norm of the Hessian is bounded [93]. As long as we choose the rate at which s_k decreases appropriately, each solution for the kth Newton system will be sufficiently close to the solution of the $(k+1)$th system that a few Newton iterations keep us close enough to the central path to ensure that, as $s_k \to 0$, the IP iterates $(\boldsymbol{x}(s_k), \boldsymbol{\lambda}(s_k), \boldsymbol{\mu}(s_k))$

converge to a KKT point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ of the original problem (2.1).

A more detailed discussion of how to solve the Newton system is reserved for later sections. Since the system matrix is generally indefinite, it can be advantageous to reduce the system by block elimination to obtain a matrix more amenable to efficient solution methods. Section 3.1.1 describes an implementation of the IP method for the compliance minimization problem and the treatment of the Newton system arising from it. To stay within a certain neighbourhood of the central path, it is not even necessary to compute the Newton step to machine precision accuracy. *Inexact* IP methods, see for example [14, 36], use iterative solvers – briefly reviewed in Section 2.4.1 – which can reduce computational cost and be better suited to large-scale problems. Convergence proofs typically require that the accuracy of the solver is coupled to the progress of the IP iterations.

One important aspect of primal-dual interior point methods still remains to be addressed, which is the choice of the step size κ in (2.18). A well-known result for the Newton method is that within a neighbourhood of a solution that satisfies certain regularity criteria, one can always choose the step size $\kappa = 1$ with guaranteed superlinear or even quadratic convergence to the solution. In our case, however, as s becomes smaller and we move closer to the boundary of X, taking the full Newton step will inevitably yield an iterate which is primal and/or dual infeasible, i.e. $g_i(\boldsymbol{x}) > 0$ or $\lambda_i < 0$ for some i. Not only is the objective function of the minimization problem (2.15), which motivated the IP approach, undefined outside the interior of X, but zero or negative values in the matrix blocks Λ or G in (2.17) can lead to complications when solving this system. For these reasons, κ is usually chosen such that the iterates always (just) remain strictly feasible. Finding such a value is not in general a trivial task, but in the special case of bound constraints of the form $\underline{x}_i \leq x_i \leq \overline{x}_i$, with $\underline{x}, \overline{x} \in \mathsf{R}^n$, $\underline{x} < \overline{x}$, this is straightforward. Note that we can also choose different step sizes $\kappa_{\boldsymbol{x}}$, $\kappa_{\boldsymbol{\lambda}}$, $\kappa_{\boldsymbol{\mu}}$ for the different components of the iterate in (2.18).

Typical convergence results for IP methods show that the iterates converge towards a KKT point of (2.1) which is also a solution under suitable regularity conditions and if κ is reduced appropriately, see [52, (Section 5.1)] and references therein. The use of inexact Newton methods is justified by convergence analyses like that in [57], which examined a primal-dual method for nonlinear inequality and linear equality constraints and showed that superlinear convergence is guaranteed as long as the Newton system is solved approximately to within a certain accuracy. An issue that may arise in IP methods is that of ill-conditioning of the system matrix in (2.17) owing to values in both Λ and G approaching zero. While it is often much less of a problem than might be expected in practice [52], it proved to be a major drawback of the IP method for the minimum compliance problem as discussed in Chapter 3. A different class of methods which, for our case, manages to avoid this pitfall is described in the following section.

2.1.4 Nonlinear Rescaling and Penalty-Barrier Multiplier Methods

As mentioned in the previous section, IP methods and, in particular, classical barrier methods suffer from ill-conditioning of the Newton system matrix as the iterates approach the boundary of the feasible set X. To bypass this well-known problem, Polyak proposed so-called modified barrier function methods [100], see also [88, 56]. These are the basis for penalty-barrier multiplier methods which are employed in this thesis and we therefore include a brief introduction here. Regard once more the inequality constrained minimization problem (2.13). If we shift the boundary of the feasible set by a value equal to the barrier parameter s, we obtain the problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\{f(\boldsymbol{x}) \mid g_i(\boldsymbol{x}) \le s \Leftrightarrow \frac{g_i(\boldsymbol{x})}{s} - 1 \le 0 \ \ \forall i \in \mathbf{I}\}. \tag{2.19}$$

We can see that, if we applied the logarithmic barrier function to (2.19), giving us $f(\boldsymbol{x}) - s \ _{i\in\mathrm{I}} \ln(1 - g_i(\boldsymbol{x})/s)$, it would prevent divergence at the boundary of the "un-shifted" feasible set X. Furthermore, because the constraints in (2.19) are themselves defined in terms of the barrier parameter, as s goes to zero, the shifted boundary approaches that of X. Next, consider another modified problem,

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\left\{f(\boldsymbol{x}) \mid -s\ln\left(1-\frac{g_i(\boldsymbol{x})}{s}\right)\leq 0 \;\; \forall i \in \mathbf{I}\right\}. \tag{2.20}$$

Observe that the feasible set of (2.20) coincides with X for all $s > 0$. The real significance of (2.20), however, becomes clear once we construct its Lagrangian

$$F(\boldsymbol{x}) := f(\boldsymbol{x}) - s\sum_{i\in\mathrm{I}}\lambda_i \ln\left(1-\frac{g_i(\boldsymbol{x})}{s}\right). \tag{2.21}$$

The above function is called the *modified barrier function* [100] for (2.13). It has an obvious similarity with the logarithmic barrier function for the shifted problem (2.19). Moreover, we can see from (2.21) that a KKT point for (2.20) also satisfies the KKT conditions for the original inequality constrained problem (2.13) and vice versa. For convex programs, this means that the former and the latter have the same solution for any s. In the non-convex case, there exists a $\underline{s} > 0$ such that for all $s > \underline{s}$, a KKT point of the NLP is also a KKT point of (2.20) and $F(\boldsymbol{x})$ is strictly convex [100]. It is therefore possible to obtain a KKT point for the original problem through an iterative process in which we fix the multipliers $\lambda_i > 0$, $i \in \mathrm{I}$, solve $\nabla F(\boldsymbol{x}) = \mathbf{0}$ and update the multipliers in an appropriate way. We can do this for a fixed s that is large enough, although it is preferable to define a sequence $s_k \to 0$, as this is necessary to achieve superlinear convergence. This basic approach is reminiscent of augmented Lagrangian methods [105, 106, 107] and, indeed, $F(\boldsymbol{x})$ can be interpreted as an augmented Lagrangian function for the original problem (2.13). While it would be negligent to not mention this established class of optimization methods, any sort of proper treatment of augmented Lagrangian methods is beyond the scope of this thesis and we refer the interested reader to the book by Bertsekas [23].

The above technique can be generalized by considering a wider range of barrier functions that share the salient features of (2.21). Such an approach has been investigated both under the name of *penalty-barrier multiplier* (PBM) methods [17, 31], proposed by Ben-Tal and Zibulevsky, and in the context of *nonlinear rescaling* (NR) methods by Polyak and Teboulle [103]. Both frameworks are very similar and differ only in small details in the definition of the class of generalized barrier functions. Since our implementation uses a specific function that falls within the class of PBM functions, we will adhere to that nomenclature in the following. The same function has also been successfully used in the optimization software Pennon [76]. More recent results in the context of NR methods should however not go unmentioned. In particular, Griva and Polyak proposed an adaptive updating scheme for s, improving convergence behaviour [60], and primal-dual NR methods were shown in [101, 102] to achieve local quadratic convergence under certain regularity assumptions.

We now give a brief description of the PBM method. For details on the theory, we refer to [15]. As before, we start by considering the inequality constrained minimization problem (2.13). We will additionally assume that the problem is convex, meaning that both f and g_i, $i \in \mathbf{I}$ are convex functions. We rescale the inequalities as $p\,\phi\,(g_i(\boldsymbol{x})/p) \leq 0$ with a *penalty function*[1] ϕ and a *penalty parameter* $p > 0$, where ϕ belongs to a class of functions with the following properties: They are strictly increasing, twice differentiable, real-valued, strictly convex functions with domain $(-\infty, b)$, where $0 < b \leq \infty$. Furthermore, they satisfy:

(ϕ1) $\phi(0) = 0$

(ϕ2) $\phi\,(0) = 1$

(ϕ3) $\lim_{s \to b} \phi\,(s) = \infty$

(ϕ4) $\lim_{s \to -\infty} \phi\,(s) = 0.$

[1]The switch from "barrier" to "penalty" is made deliberately to underline that we no longer restrict ourselves to functions that enforce strict feasibility.

For all such penalty functions, the *rescaled* problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\left\{f(\boldsymbol{x}) \mid p\,\phi\left(\frac{g_i(\boldsymbol{x})}{p}\right) \le 0,\ i = 1,\dots,m\right\} \tag{2.22}$$

retains convexity and has the same feasible set and the same solution as (2.13), since a KKT point for either problem is also a KKT point for the other due to properties (ϕ1) and (ϕ2). Formulating a standard Lagrangian function of the rescaled problem gives us the following *augmented Lagrangian function* for the original problem (2.13):

$$\mathrm{L}_p(\boldsymbol{x};\boldsymbol{\lambda}) := f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i p\,\phi\left(\frac{g_i(\boldsymbol{x})}{p}\right). \tag{2.23}$$

It can be shown that L_p is strictly convex in $\boldsymbol{x}$ for any $\boldsymbol{\lambda} > \mathbf{0}$ and $p > 0$.

The iterative scheme of the PBM method is as follows: For fixed $p > 0$ and $\boldsymbol{\lambda} > \mathbf{0}$, we want to determine a vector $\boldsymbol{x}$ that satisfies the KKT conditions for the rescaled problem. We do this by setting $\nabla_{\boldsymbol{x}}\mathrm{L}_p(\boldsymbol{x};\boldsymbol{\lambda}) = \mathbf{0}$. Since L is strictly convex in $\boldsymbol{x}$, this is equivalent to minimising it with respect to $\boldsymbol{x}$. Similarly to the IP method, we do not actually require the exact solution of this subproblem. It suffices to get "close" to it. Following this, we update the Lagrange multipliers and penalty parameter. This constitutes one iteration of the PBM method, the details of which are:

$$\text{Step 1.}\quad \boldsymbol{x}^{(k+1)} \approx \arg\min_{\boldsymbol{x}} \mathrm{L}_{p^{(k)}}(\boldsymbol{x};\boldsymbol{\lambda}^{(k)}) \tag{2.24a}$$

$$\text{Step 2.}\quad \lambda_i^{(k+1)} = \lambda_i^{(k)}\phi'\left(\frac{g_i(\boldsymbol{x}^{(k+1)})}{p^{(k)}}\right) \tag{2.24b}$$

$$\text{Step 3.}\quad p^{(k+1)} = \pi^{(k)}p^{(k)}\,. \tag{2.24c}$$

Here, $\pi^{(k)} < 1$ is a penalty updating factor which can generally be different for each iteration. In Step 1, we run the Newton method until $\|\nabla_{\boldsymbol{x}}\,\mathrm{L}_p(\boldsymbol{x};\lambda)\|_2 \le \varepsilon^{(k)}$, where $\varepsilon^{(k)}$ is some prescribed tolerance, typically determined by the progress of the algorithm. If we choose a penalty function with $b = \infty$, we no longer run into the problem of full Newton steps resulting in infeasible iterates, as observed for the IP method. The step size can therefore be chosen by a standard line search strategy, for example by imposing the Armijo condition [93]. If the PBM method converges, it finds a KKT point for the original convex NLP with inequality constraints – and thus a primal-dual optimal solution.

For the Newton method in Step 1 of the PBM algorithm, we require the gradient and Hessian of the augmented Lagrangian with respect to $\boldsymbol{x}$. These are given by

$$\nabla_{\boldsymbol{x}} \mathrm{L}_p(\boldsymbol{x};\boldsymbol{\lambda}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \phi'\left(\frac{g_i(\boldsymbol{x})}{p}\right) \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) \tag{2.25}$$

and

$$\begin{aligned} \nabla^2_{\boldsymbol{x}} \mathrm{L}_p(\boldsymbol{x};\boldsymbol{\lambda}) = \nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}) &+ \sum_{i=1}^{m} \frac{\lambda_i}{p} \phi''\left(\frac{g_i(\boldsymbol{x})}{p}\right) \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) (\nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}))^\top \\ &+ \sum_{i=1}^{m} \lambda_i \phi'\left(\frac{g_i(\boldsymbol{x})}{p}\right) \nabla^2_{\boldsymbol{x}} g_i(\boldsymbol{x}) , \end{aligned} \tag{2.26}$$

where we have used $\nabla^2_{\boldsymbol{x}} = \nabla_{\boldsymbol{xx}}$ to denote the Hessian with respect to $\boldsymbol{x}$. Note that, due to the convexity of the rescaled problem (2.22), the Hessian of L_p is positive semidefinite for any $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}^m_+$.

While many candidates meet the definition of a penalty function given by (ϕ1)–(ϕ4), one particular choice for ϕ, found for example in [17], proved to be particularly efficient in our numerical trials:

$$\phi(\tau) = \begin{cases} -\frac{1}{4}\log(-2\tau) - \frac{3}{8}, & \text{if } \tau < -\frac{1}{2} , \\ \tau + \frac{1}{2}\tau^2 , & \text{if } \tau \geq -\frac{1}{2} . \end{cases} \tag{2.27}$$

This function combines properties of a quadratic penalty function and the logarithmic barrier function, which motivates the nomenclature “penalty-barrier function”. The NR framework includes a similar type of function, see for example [59]. The main difference to (2.27) is that the quadratic branch of the NR penalty function needs to lie completely outside the feasible set. While this is a requirement for the convergence analysis in [59] and the related papers [101, 102], we observed slightly faster and more reliable convergence behaviour for (2.27).

2.2 Finite Element Method for Linear Elasticity

Consider a continuum of linearly elastic material, described by a domain Ω in $\mathbb{R}^d$ with Lipschitz boundary Γ, where d is either 2 or 3. Assume that a mechanical scenario is specified on this domain, defined by the material properties of the continuum and by boundary conditions; the latter comprise forces on some part $\Gamma_f \neq \emptyset$ of the boundary and prescribed zero deformation (due to bearings) on another part $\Gamma_u \neq \emptyset$, which we assume to have a non-empty relative interior. The boundary subsets Γ_f and Γ_u are disjoint and $\Gamma_f \cup \Gamma_u = \Gamma$. For simplicity's sake, we do not consider body forces over Ω or prescribed non-zero deformation on the boundary. We assume that deformations are small enough that we do not need to differentiate between a current and a reference configuration, which means that a material point's location is approximately the same before and after deformation and is defined by a single point $\boldsymbol{x} \in \bar{\Omega}$ somewhere in the closure of the domain – as opposed to two points, one in the reference domain, the other in the current domain. Now, let $\boldsymbol{f} : \Gamma_f \rightarrow \mathbb{R}^d$ denote the boundary forces, or *loads*, and let us use $\boldsymbol{u} : \bar{\Omega} \rightarrow \mathbb{R}^d$ to denote the displacement $\boldsymbol{u}(\boldsymbol{x})$ of a material point located at $\boldsymbol{x} \in \bar{\Omega}$. Finally, the state of stress at each point in the domain is quantified by the *stress tensor* $\mathrm{T} : \bar{\Omega} \rightarrow \mathbb{R}^{d\times d}_{sym}$ $\mathrm{T} = [\,\tau_{ij}\,]^3_{i,j=1}$. For the purpose of this brief introduction, we will ignore the distinction between a second order tensor and its matrix representation. The displacements and stresses in the domain have to satisfy the boundary value problem

$$\operatorname{div} \mathrm{T} = \mathbf{0} \quad \text{on } \Omega, \tag{2.28a}$$

$$\mathrm{T}\boldsymbol{n} = \boldsymbol{f} \quad \text{on } \Gamma_f, \tag{2.28b}$$

$$\boldsymbol{u} \equiv \mathbf{0} \quad \text{on } \Gamma_u, \tag{2.28c}$$

where $\boldsymbol{n}$ is the outer surface normal on the domain boundary. System (2.28a) enforces static equilibrium everywhere in Ω and (2.28b) and (2.28c) are the boundary conditions according to the prescribed mechanical model. In order to obtain a well-posed problem, we need to link the state of stress to the state of deformation. The latter is characterized by the *(small) strain tensor*

$$\mathrm{E}(\boldsymbol{u}) = [\varepsilon_{ij}]_{i,j=1}^{3} := \left(\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right)_{i,j=1}^{3}$$

which is evidently symmetric. Hooke's law relates E to the stresses at each point $\boldsymbol{x} \in \Omega$ by the linear constitutive equation

$$\mathrm{T}(\boldsymbol{x}) = \mathsf{C}[\mathrm{E}(\boldsymbol{u}(\boldsymbol{x}))]\,. \tag{2.29}$$

Here, C is the fourth order elasticity tensor which maps the second order strain tensor to the second order stress tensor. Based on energy considerations, it can be shown to have the following important property.

Lemma 2.2.1. *The fourth order elasticity tensor for linearly elastic materials* $\mathsf{C} \in \mathsf{R}^{d\times d\times d\times d}$ *is symmetric in that*

$$\mathrm{X} \bullet \mathsf{C}[\mathrm{Y}] = \mathrm{Y} \bullet \mathsf{C}[\mathrm{X}]$$

for any two second order tensors $\mathrm{X}, \mathrm{Y} \in \mathsf{R}^{d\times d}$.

Proof. See for example [89, (Chapter 3.2)]. □

We further assume that C is uniformly positive definite, i.e., there exists a $c_0 > 0$ such that

$$\mathrm{X} \bullet \mathsf{C}[\mathrm{X}] \geq c_0$$

for any $\mathrm{X} \neq 0$.

With (2.28) and (2.29), we now have a well-posed boundary value problem in the displacements. We call the solution $\boldsymbol{u}$ *classical solution*. The requirements on this solution in terms of differentiability are rather strict: the displacements need to be twice continuously differentiable everywhere in the domain, i.e., $\boldsymbol{u} \in [C^2(\Omega)]^d$. We can ease these requirements by employing variational formulations, which we will now derive. Let us assume for the moment that the displacements and boundary forces are "smooth enough". Furthermore, we assume that $\boldsymbol{u}$ is geometrically admissible, i.e. it satisfies the boundary conditions (2.28c). If we take the inner product of both sides of (2.28a) with the displacements $\boldsymbol{u}$, then integrate over Ω using Green's identity, we arrive at

$$\int_\Omega \mathrm{T} \bullet \mathrm{E}(\boldsymbol{u}) \, \mathrm{d}\Omega = \int_{\Gamma_f} \boldsymbol{f} \cdot \boldsymbol{u} \, \mathrm{d}\Gamma + \int_{\Gamma_u} (\mathrm{T}\boldsymbol{n}) \cdot \boldsymbol{u} \, \mathrm{d}\Gamma, \tag{2.30}$$

where "$\cdot$" is the inner vector product. The above equation illustrates the principle of virtual work, which states that the work done by the internal forces equals that done by the external forces. Let us compare $\boldsymbol{u}$ to a displacement field which deviates from $\boldsymbol{u}$ in Ω but is also admissible. Such a displacement field is given by $\boldsymbol{u} + \boldsymbol{v}$ for any $\boldsymbol{v}$ which is zero on Γ_u. Equation (2.30) still holds if we substitute $\boldsymbol{u}$ by $\boldsymbol{u} + \boldsymbol{v}$. If we subtract from this the original equation (2.30), we can see that

$$\int_\Omega \mathrm{T} \bullet \mathrm{E}(\boldsymbol{v}) \, \mathrm{d}\Omega - \int_{\Gamma_f} \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\Gamma = 0, \tag{2.31}$$

which holds for any $\boldsymbol{v}$ which vanishes on Γ_u (and for which the integrals are defined). Recall that a classical solution $\boldsymbol{u}^*$ satisfies the constitutive equation (2.29) everywhere in Ω. Since C is symmetric and positive definite, we can define a potential energy term

$$\Pi(\boldsymbol{u}) := \frac{1}{2}\int_\Omega \mathrm{E}(\boldsymbol{u}) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u})]\, \mathrm{d}\Omega - \int_{\Gamma_f} \boldsymbol{f}\cdot\boldsymbol{u}\, \mathrm{d}\Gamma, \tag{2.32}$$

called the *potential energy of the elastic body*, and view the left-hand side of (2.31) as a variation of Π at $\boldsymbol{u}^*$ in the direction $\boldsymbol{v}$. Then, (2.31) implies that the classical solution is a critical point of the potential energy Π. In fact, it can be shown that $\boldsymbol{u}^*$ minimizes Π over the set of all geometrically admissible displacements, a result commonly referred to as the *principle of minimum potential energy*.

The variational equation (2.31) with $\mathrm{T} = \mathsf{C}[\mathrm{E}(\boldsymbol{u})]$ offers an alternative characterization of a solution to the boundary value problem. Importantly, this solution does not need to satisfy the same differentiability requirements as for the set of partial differential equations (2.28), so that it can be chosen from a larger function space. In particular, we consider the Sobolev space $\mathrm{V} = \{v \mid \int_\Omega v(\boldsymbol{x})^2 \mathrm{d}\Omega + \int_\Omega \|\nabla v(\boldsymbol{x})\|_2^2 \mathrm{d}\Omega < \infty\} \cap \{v \mid v \equiv 0 \text{ on } \Gamma_u\}$ and define $V := [\mathrm{V}]^d$. We assume the boundary forces are square integrable, i.e. $\boldsymbol{f} \in [L^2]^d$. This leads to the *weak formulation* of the boundary value problem

$$\begin{gathered}\text{Find } \boldsymbol{u} \in V \text{ s.t.} \\ \int_\Omega \mathrm{E}(\boldsymbol{v}) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u})]\, \mathrm{d}\Omega - \int_{\Gamma_f} \boldsymbol{f}\cdot\boldsymbol{v}\, \mathrm{d}\Gamma = 0 \quad \forall \boldsymbol{v} \in V\,. \end{gathered} \tag{2.33}$$

If $\boldsymbol{u}^*$ solves the above problem, it is called a *weak solution*. Any classical solution is also a weak solution, but the reverse only holds if the weak solution is smooth enough. The functions $\boldsymbol{v} \in V$, which we interpreted as variations of $\boldsymbol{u}$ in the principle of minimum potential energy, are called *test functions* in the context of the weak formulation. An important question is whether (2.33) always has a solution and whether it is unique, which is answered by the following result.

Theorem 2.2.2. *The weak formulation (2.33) of the boundary value problem has a unique solution.*

Proof. We refer the reader to [89, (Chapter 7)] for a complete proof and only highlight a few important points. For this, we will need the following norm for functions $\boldsymbol{v} \in V$:

$$\|\boldsymbol{v}\|_V^2 := \int_\Omega \|\boldsymbol{v}\|_2^2 \, d\Omega + \int_\Omega \|\nabla \boldsymbol{v}\|_2^2 \, d\Omega .$$

The analysis of (2.33) is based on Korn's inequality, see for example [44, (Chapter 3, §3.3)] or [89, (Chapter 6.3)], from which it follows that the bilinear form

$$(\boldsymbol{v}, \boldsymbol{u}) \mapsto \int_\Omega \mathrm{E}(\boldsymbol{v}) \bullet \mathrm{E}(\boldsymbol{u}) \, d\Omega$$

satisfies

$$\alpha \|\boldsymbol{v}\|_V^2 \leq \int_\Omega \mathrm{E}(\boldsymbol{v}) \bullet \mathrm{E}(\boldsymbol{v}) \, d\Omega$$

for any $\boldsymbol{v} \in V$ and a constant $\alpha > 0$ which is independent of $\boldsymbol{v}$. A bilinear form which satisfies the above inequality is called V-elliptic. Because C is symmetric, see Lemma 2.2.1, and positive definite, the V-ellipticity extends to the bilinear form $\int_\Omega \mathrm{E}(\boldsymbol{v}) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u})]$, see [89, (Chapter 7)]. The uniqueness of the solution of (2.33) is then a consequence of the Lax-Milgram theorem, see for example [40, (Theorem 1.1.3)] or [81, (Theorem 5.1.1)]. □

The weak formulation is the basis for the finite element method, which provides a consistent and efficient approach to finding an approximate solution of (2.33). For this, we confine ourselves to a finite-dimensional subspace V_h which approximates V. To motivate the particular subspace we will use, we need to first introduce a discretization of the domain Ω.

We divide $\overline{\Omega}$ into a finite number of convex polyhedra, called *elements*, thus creating a *mesh* or *grid* $\overline{\Omega}_h$. We limit our discussion to quadrilateral ($d = 2$) and hexahedral ($d = 3$) elements, although other choices are possible and common in other applications. The vertices of the polyhedra are called *nodes*. We assume that any two adjacent elements share exactly one face and that no node lies in the interior of an element's face, (i.e. we do not consider higher-order or non-conforming elements). The nodes along the boundary Γ_h of the mesh Ω_h lie on Γ, although, depending on the geometry of Γ, the polyhedral boundary Γ_h might obviously not coincide with Γ everywhere, but only approximate it. We assume that our mesh conforms in the same way to the boundaries Γ_u and Γ_f and refer to the corresponding parts of the mesh boundary as $\Gamma_{u,h}$ and $\Gamma_{f,h}$, respectively. A quick note for clarification: the subscript h is commonly used because it denotes some measure of the mesh resolution, such as the maximum edge-length or element-area/volume in the mesh. For later reference, let m denote the number of elements in the mesh and let N be the number of nodes.

Turning back to our search for an appropriate V_h, we consider the space $\mathcal{V}$ of piecewise bilinear (in two dimensions) or piecewise trilinear (in three dimensions) functions which are continuously differentiable in the interior of each element. Let us denote the locations of the mesh nodes by $\boldsymbol{x}_i$ for $i = 1, \ldots, N$. As a basis of our function space, we choose $\varphi_i(\boldsymbol{x})$ for $i = 1, \ldots, N$, such that $\varphi_i(\boldsymbol{x}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta, and which are identically equal to zero outside of elements which are adjacent to the node $\boldsymbol{x}_i$. Every function $f \in \mathcal{V}$ can then be written in terms of its values $f_i := f(\boldsymbol{x}_i)$ at the nodes, namely by $f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i \varphi_i(\boldsymbol{x})$. We will use $\mathcal{V}$ to construct the function space V_h. Furthermore, to stay consistent with V, we want all functions in V_h to vanish on $\Gamma_{u,h}$. We can easily achieve this by setting the corresponding nodal coefficients to zero. In summary, we define

$$V_h := \left\{ \boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} : \begin{array}{l} v_i(\boldsymbol{x}) = \sum_{k=1}^{N} v_{i,k}\varphi_k(\boldsymbol{x}) \text{ with} \\ v_{i,j} = 0 \text{ if } \boldsymbol{x}_j \in \Gamma_{u,h}, \quad \forall i = 1, \ldots, d \end{array} \right\}.$$

Remark 2.2.3. We have only considered the case that *all* components of a vector-valued function in V vanish at Γ_u. This can be generalized to boundary conditions which require only some components to be zero on Γ_u and, further, to several disjoint subsets of Γ_u, each associated with boundary conditions on a different set of components. On such a subset of Γ_u on which not all components of $\boldsymbol{u}$ are fixed, we can then even prescribe a load that lies in the range of the remaining degrees of freedom. This leads to some technical but straightforward adjustments of the function spaces V and V_h, which we omit for ease of presentation.

Now that we have defined a function space that the approximate solution should be taken from, we can reformulate (2.33) as follows:

$$\begin{gathered}\text{Find } \boldsymbol{u}_h \in V_h \subset V \text{ s.t.} \\ \int_{\Omega_h} \mathrm{E}(\boldsymbol{v}_h) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u}_h)]\,\mathrm{d}\Omega - \int_{\Gamma_{f,h}} \boldsymbol{f} \cdot \boldsymbol{v}_h\,\mathrm{d}\Gamma = 0 \quad \forall \boldsymbol{v}_h \in V_h.\end{gathered} \tag{2.34}$$

Since we can express $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$ as linear combinations of basis functions, it is possible to write (2.34) using matrix–vector multiplications. Let us define a vector $\hat{\boldsymbol{u}}$ that contains all nodal coefficients of $\boldsymbol{u}_h$, ordered first by node then by component, i.e. $\hat{u} := ((u_h)_{1,1}, (u_h)_{2,1}, \dots, (u_h)_{d,N})^\mathsf{T}$, and let us define $\hat{\boldsymbol{v}}$ analogously. In these vectors, we do not include those coefficients which are identically equal zero due to boundary conditions. We denote by n the length of these vectors. In other words, n is the number of degrees of freedom in problem (2.34). Lastly, we define vectors $\boldsymbol{\varphi}_i(\boldsymbol{x}) \in \mathbb{R}^d$ for $i = 1, \dots, n$ in such a way that we can write $\boldsymbol{u}_h(\boldsymbol{x}) = [\boldsymbol{\varphi}_1(x), \dots, \boldsymbol{\varphi}_n(x)]\,\hat{\boldsymbol{u}}$ and $\boldsymbol{v}_h(\boldsymbol{x}) = [\boldsymbol{\varphi}_1(x), \dots, \boldsymbol{\varphi}_n(x)]\,\hat{\boldsymbol{v}}$. Equation (2.34) is then equivalent to

$$\hat{\boldsymbol{v}}^\mathsf{T}\mathrm{K}\hat{\boldsymbol{u}} - \hat{\boldsymbol{v}}^\mathsf{T}\hat{\boldsymbol{f}} = 0 \quad \forall \hat{\boldsymbol{v}} \in \mathbb{R}^n, \tag{2.35}$$

where $\mathrm{K} \in \mathbb{R}^{n\times n}$ is the *stiffness matrix*, given by

$$\mathrm{K}_{ij} := \int_{\Omega_h} \mathrm{E}(\boldsymbol{\varphi}_i) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{\varphi}_j)]\,\mathrm{d}\Omega \quad \text{for } i,j = 1, \dots, n, \tag{2.36}$$

and $\hat{\boldsymbol{f}}$ is the nodal load vector, defined as

$$\hat{f}_i := \int_{\Gamma_{f,h}} \boldsymbol{f} \cdot \boldsymbol{\varphi}_i\,\mathrm{d}\Gamma \quad \text{for } i = 1, \dots, n. \tag{2.37}$$

Because (2.35) needs to hold for any $\hat{\boldsymbol{v}} \in \mathsf{R}^n$, it is equivalent to the system of equations

$$\mathrm{K}\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}}. \tag{2.38}$$

Due to our specific choice of basis functions, this linear system can be efficiently set up and solved. To compute the stiffness matrix and the load vector, we need to compute the integrals (2.36) and (2.37), respectively. This is done element by element. For instance, we define an *elemental stiffness matrix* $\mathrm{K}_e \in \mathsf{R}^n$ for each element with index $e = 1, \ldots, m$ by

$$(\mathrm{K}_e)_{ij} := \int_{\Omega_h^e} \mathrm{E}(\boldsymbol{\varphi}_i) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{\varphi}_j)]\, \mathrm{d}\Omega \quad \text{for } i, j = 1, \ldots, n, \tag{2.39}$$

where Ω_h^e is the part of the domain taken up by the eth element. Since the only basis functions that are non-zero on element e are those which correspond to the element's nodes, K_e is extremely sparse. FEM implementations typically use an incidence matrix, which allows us to know a priori which basis functions to consider for each element. Computing the integral numerically, for example by Gaussian quadrature rule with 2 points per problem dimension, can therefore be done quite efficiently. The *global* stiffness matrix is then simply assembled by $\mathrm{K} = \sum_{e=1}^{m} \mathrm{K}_e$. Due to the localized support of each basis function, there is limited overlap between the elemental stiffness matrices and therefore K is sparse. Figure 2.1 shows an example of the sparsity structure for a regular three-dimensional mesh.

Since the bilinear form $\int_\Omega \mathrm{E}(\boldsymbol{v}) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u})]$ is V-elliptic, as discussed in the proof of Theorem 2.2.2, it is obviously also V_h-elliptic. As a consequence, K is symmetric positive definite and (2.38) has a unique solution, assuming the mechanical structure defined by the domain and boundary conditions is kinematically determinate. Each K_e is symmetric and positive semidefinite. Due to the positive definiteness and sparsity of K, system (2.38) can be solved efficiently, for example by iterative Krylov solvers, described in Section 2.4.1.

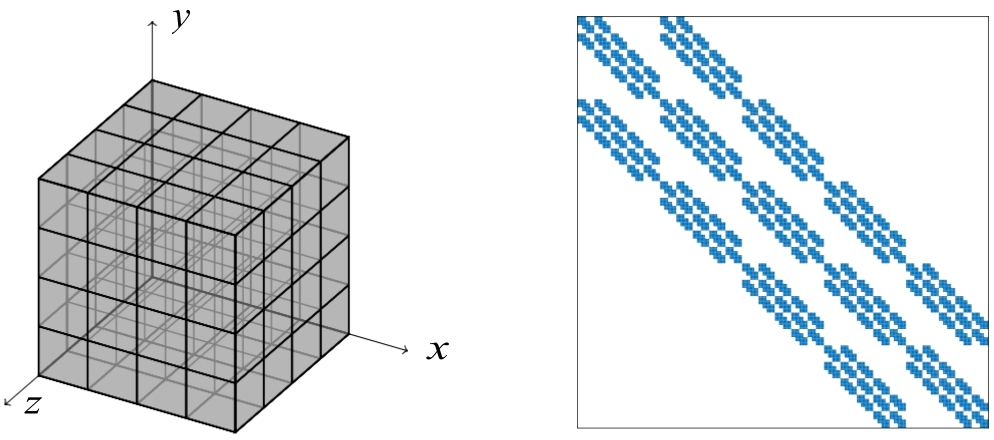


Figure 2.1: Regular, structured FE mesh with $4 \times 4 \times 4$ cube-shaped elements and the sparsity structure of the corresponding stiffness matrix.

An important theoretical result for the finite element method is that the principal of minimum potential energy extends to the FE discretization. Recall the definition (2.32) of the potential energy of the elastic body. For the discretized case, we get the following identity:

$$\begin{aligned}\Pi(\boldsymbol{u}_h) &= \frac{1}{2}\int_{\Omega_h} \mathrm{E}(\boldsymbol{u}_h) \bullet \mathsf{C}[\mathrm{E}(\boldsymbol{u}_h)]\,\mathrm{d}\Omega - \int_{\Gamma_{f,h}} \boldsymbol{f} \cdot \boldsymbol{u}_h\,\mathrm{d}\Gamma \\ &= \frac{1}{2}\hat{\boldsymbol{u}}^{\mathsf{T}}\mathrm{K}\hat{\boldsymbol{u}} - \hat{\boldsymbol{f}}^{\mathsf{T}}\hat{\boldsymbol{u}} \;=:\; \hat{\Pi}(\hat{\boldsymbol{u}})\,.\end{aligned}$$

It can be shown that the solution $\boldsymbol{u}_h^*$ of (2.34), with associated nodal vector $\hat{\boldsymbol{u}}^*$, minimizes Π over V_h, which can be written in a vectorized form as

$$\hat{\boldsymbol{u}}^* = \arg\min_{\hat{\boldsymbol{u}} \in \mathbb{R}^n} \hat{\Pi}(\hat{\boldsymbol{u}}) = \arg\min_{\hat{\boldsymbol{u}} \in \mathbb{R}^n} \frac{1}{2}\hat{\boldsymbol{u}}^{\mathsf{T}}\mathrm{K}\hat{\boldsymbol{u}} - \hat{\boldsymbol{f}}^{\mathsf{T}}\hat{\boldsymbol{u}}\,. \tag{2.40}$$

For the sake of completeness, we provide a few more details on the elasticity tensor C. While it has d^4 components overall, these can be reduced to only 2 independent variables for any isotropic material. In that case, which is the one we consider in this thesis, C can be expressed in terms of two material constants, for example the Young modulus $E > 0$ (not to be confused with the strain tensor E) and the Poisson ratio $0 \le \nu < \frac{1}{2}$. In vectorized form, the strains and stresses then satisfy the constitutive equations

$$\begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{12}\end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}\begin{bmatrix}\tau_{11}\\ \tau_{22}\\ \tau_{12}\end{bmatrix}$$

for $d = 2$, where we assume a state of plane stress as opposed to plane strain, and

$$\begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \varepsilon_{12}\\ \varepsilon_{23}\\ \varepsilon_{31}\end{bmatrix} = C\begin{bmatrix}1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)}\end{bmatrix}\begin{bmatrix}\tau_{11}\\ \tau_{22}\\ \tau_{33}\\ \tau_{12}\\ \tau_{23}\\ \tau_{31}\end{bmatrix},$$

where $C = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$, for $d = 3$. The lower triangular elements of E and T are defined by symmetry. The material constants need to satisfy certain conditions to ensure the invertibility of the elasticity tensor, see [89]. It can be assumed that all such criteria are met for the choices of E and ν used in our numerical experiments.

In the following sections and chapters, we will only be interested in the FE approximation of the boundary value problem. Hence, we will take the terms displacements and loads to be synonymous with the corresponding nodal vectors. Furthermore, we will simplify our notation by using $\boldsymbol{u}$ and $\boldsymbol{f}$ for those vectors and generally omit the hat wherever it was used to distinguish the vectorized approximate form. Likewise, we will drop the subscript h from the discretized domain and boundary.

Remark 2.2.4. While we did not cover the case of non-zero prescribed displacements at the boundary, we note that this can be incorporated into the presented approach as well. Similarly to boundary loads, such prescribed displacements lead to an integral over $\Gamma_{u,h}$ which contributes to the vector $\hat{\boldsymbol{f}}$.

Remark 2.2.5. We do not include any formal convergence analysis here and refer to the literature [40]. Subject to the regularity of the boundary value problem, one can usually expect the FE approximation to converge to the weak solution as $h \to 0$

with at least a linear rate. Under certain regularity assumptions, convergence of a higher order can be shown as well. For the problems we consider in this thesis, the mesh resolution is mainly dictated by the level of detail of the design we wish to model and we can generally assume that this resolution is fine enough to yield sufficiently accurate results for the displacements.

2.2.1 Unilateral contact

To conclude this section on the finite element method, we include a brief discussion of unilateral contact. Boundary conditions that account for obstacles in addition to merely prescribing fixed displacements along parts of the boundary Γ are an indispensable tool for more realistic modelling. As it will turn out, unilateral contact can be included in the FE model used in our optimization problem without too much additional effort, see Section 2.3.2. We provide here a pragmatic introduction starting from the discretized problem, which is geared towards our application in topology optimization. Contact is assumed to be frictionless and obstacles are treated as rigid bodies. For a rigorous analytic treatment of both the exact problem and the finite element discretization, as well as for more complicated contact modelling, such as for example contact with friction, bilateral contact, large deformations, and non-linearity, we refer to the literature – see for example [73, 69]. In particular, an analysis of the discretized contact constraints model which we employ can be found in [73, Section 6.4].

Consider once more the (discretized) domain Ω and its boundary $\Gamma = \Gamma_u \cup \Gamma_f$. In addition, let us now assume the presence of some obstacle close enough to the boundary that it might impede the deformation we would see due to the prescribed loads. We will refer to the part of Γ which we expect to potentially come into contact with the obstacle as the *contact surface* (both for $d = 2$ and $d = 3$) and will denote it by Γ_c. We assume that $\Gamma = \Gamma_u \cup \Gamma_f \cup \Gamma_c$ and that all three subsets are mutually disjoint[1]. To ensure that none of the nodes $\boldsymbol{x}_i$, $i \in \{1, \ldots, N\}$, which are on the

[1]Since obstacles typically only restrict degrees of freedom in a single direction, we can weaken this

contact surface penetrate the obstacle's boundary, we impose a *contact constraint* on each $\boldsymbol{x}_i \in \Gamma_c$. If we assume small deformations, we can approximate the boundary of the obstacle by its tangential plane at a point $\boldsymbol{y}_i$, which is the one closest to $\boldsymbol{x}_i$. This allows us to formulate the constraint by limiting the displacement of $\boldsymbol{x}_i$ along the obstacle's outer surface normal $\boldsymbol{n}_i$ at $\boldsymbol{y}_i$. Figure 2.2 illustrates the concept. If

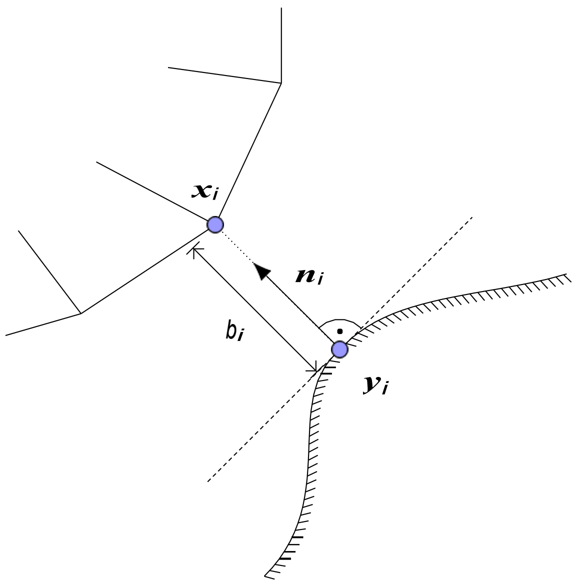


Figure 2.2: Node $\boldsymbol{x}_i$ on contact surface Γ_c, closest point $\boldsymbol{y}_i$ on the obstacle boundary, surface normal $\boldsymbol{n}_i$ and initial gap b_i.

we denote by $b_i := (\boldsymbol{x}_i - \boldsymbol{y}_i)^\top \boldsymbol{n}_i$ the initial distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, and by $\boldsymbol{u}_i$ the displacement vector for the node $\boldsymbol{x}_i$ we can write the contact constraint as

$$-\boldsymbol{n}_i^\top \boldsymbol{u}_i \leq b_i$$

Next, let us define an index set for all contact nodes $\boldsymbol{x}_i \in \Gamma_c$:

$$\chi := \{i : \boldsymbol{x}_i \in \Gamma_c\}\,. \tag{2.41}$$

For each $i \in \chi$, we have a corresponding point $\boldsymbol{y}_i$ on the obstacle boundary and the

assumption for component-wise boundary conditions, cf. Remark 2.2.3.

surface normal $\boldsymbol{n}_i$ at that point. We can arrange all $\boldsymbol{n}_i$ in a matrix

$$\mathrm{C} := [\boldsymbol{c}_i]_{i\in x} \in \mathbb{R}^{n\times|x|}, \quad \text{where } \boldsymbol{c}_i := \begin{bmatrix} 0 \\ \vdots \\ -\boldsymbol{n}_i \\ \vdots \\ 0 \end{bmatrix},$$

such that the components of $\boldsymbol{n}_i$ in $\boldsymbol{c}_i$ line up with those of $\boldsymbol{u}_i$ in the global displacement vector $\boldsymbol{u}$[1]. If we further define the vector of gap distances $\boldsymbol{b} := (b_i)_{i\in x}$, we can collate all contact constraints in the expression

$$\mathrm{C}^{\mathsf{T}}\boldsymbol{u} \leq \boldsymbol{b}\,. \tag{2.42}$$

Now that we have derived a set of contact constraints, we need to incorporate them into the finite element problem. Recall that the FE solution can be characterized by a minimization of the potential energy over the set of admissible displacements, see (2.40). In order to impose the constraints (2.42), we can include them in the definition of admissible displacements, which turns (2.40) into the constrained minimization problem

$$\min_{\mathrm{C}^{\mathsf{T}}\boldsymbol{u}\leq\boldsymbol{b}} \Pi(\boldsymbol{u}) = \min_{\mathrm{C}^{\mathsf{T}}\boldsymbol{u}\leq\boldsymbol{b}} \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}\boldsymbol{u} - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u}\,. \tag{2.43}$$

Where before, we could use the stationarity conditions for the unconstrained minimization to derive the FE equilibrium equations (2.38), we must now employ the

[1]Components corresponding to degrees of freedom fixed by the boundary conditions are excluded in $\boldsymbol{c}_i$ just as they are in $\boldsymbol{u}$.

KKT conditions, which gives us

$$\mathrm{K}\boldsymbol{u} = \boldsymbol{f} - \mathrm{C}\boldsymbol{\eta}\,, \tag{2.44a}$$

$$\mathrm{C}^{\mathsf{T}}\boldsymbol{u} \leq \boldsymbol{b}\,, \tag{2.44b}$$

$$\boldsymbol{\eta} \geq \mathbf{0}\,, \tag{2.44c}$$

$$\boldsymbol{\eta}^{\mathsf{T}}(\mathrm{C}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{b}) = 0\,. \tag{2.44d}$$

The above system provides necessary and sufficient conditions for a solution of (2.43), if we assume that the contact problem is feasible. Indeed, since Π is quadratic with Hessian $\mathrm{K}\ 0$ and the contact constraints are linear, (2.43) is convex. Therefore, (2.44) are sufficient conditions. If a CQ holds, they are also necessary. For linear inequality constraints, Slater's condition, see Definition 2.1.15, can be relaxed from requiring the existence of a strictly feasible point to a merely feasible point [27]. Since we assume feasibility, such a point exists.

The Lagrange multipliers $\boldsymbol{\eta}$ in (2.44) permit an instructive physical interpretation: η_i is the magnitude of the contact force at $\boldsymbol{y}_i$. First, the complementarity conditions (2.44c) and (2.44d) imply that a reaction force can only exist in the case of contact, i.e. when $\boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{u} - b_i = 0$. Second, the sign of $\mathrm{C}\boldsymbol{\eta}$ in the equilibrium equations (2.44a) indicates that the contact force is a compressive force which acts only in the normal direction, which is consistent with the assumption of frictionless contact.

2.3 Minimum Compliance Topology Optimization

This section introduces the optimization problem that is at the centre of this thesis. It is one of the numerous problems within the field of *structural optimization*. In the broadest sense, this term describes the search for a geometric design of a physical structure that is optimal with respect to some objective, while also being functional for its designated purpose and satisfying certain geometric or physical criteria. In our case, for example, the structure will be a mechanical part made of isotropic, linearly elastic material that is subjected to a static load. The objective will be to maximize its stiffness, while constraints are introduced by enforcing static equilibrium and limiting the amount of available material. We will give a condensed overview of the area of structural optimization, to provide context for the subsequent detailed treatment of the so-called variable thickness sheet problem. The solution algorithms we propose in Chapter 3 for this particular problem take advantage of its specific structure and certain properties which will be discussed in the main part of this section.

Structural optimization is commonly split into the three distinct categories of *sizing*, *shape*, and *topology optimization*[1] [22], which correspond to different methodologies of parameterising the geometric design of the structure. Sizing problems involve finding the optimal dimensioning of discrete elements at predefined locations within a structure, such as the thickness of bars in truss designs or the width and height of ridges. The geometric parameters through which the design is controlled are called *design variables*. In shape optimization, one modifies not just discrete features of the design, but rather the entire boundary, or parts thereof. Hence, the shape is defined by an infinite dimensional design variable. Finally, topology optimization also considers the placement of holes in the design and the connectivity of the boundary – unlike in shape optimization, where the topology is fixed. For this reason, topology optimization is also sometimes referred to as “Generalized Shape Optimization” [112].

For a thorough introduction to the subject, we refer to the much-cited book by Bendsøe and Sigmund [22], which is a comprehensive presentation of the field of topology optimization, as well as structural optimization in general, and includes an excellent survey of the literature. Furthermore, the lecture series edited by Rozvany and Lewiński [112] covers a lot of analytical aspects and includes a very instructive historical overview.

[1]Terms such as *design* or *optimal design* in place of *optimization* are also common in this context. We will use these terms interchangeably throughout this thesis.

An illustrative definition of topology optimization is "finding the optimal material distribution within a predefined domain". Let us denote this domain, called *design domain*, by $\Omega \subseteq \mathbb{R}^d$, where $d \in \{2, 3\}$. One of the first questions that arise in modelling our problem mathematically is how to parameterize the design topology. One possibility is to use an indicator function which distinguishes those parts of Ω which are "solid", that is, filled with material, from those that are "void". If we split up the design domain into the two disjoint subsets corresponding to solid and void regions, say, Ω_S and Ω_0, we can define an indicator function

$$\theta : \Omega \to \{0, 1\}, \quad \theta(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in \Omega_S, \\ 0, & \text{if } \boldsymbol{x} \in \Omega_0 . \end{cases}$$

It is this function θ which constitutes the (infinite dimensional) design variable. It describes the material distribution within the design domain. Numerical solution methods for topology optimization typically solve the problem on an FE discretization of Ω. We would therefore approximate θ by a piecewise constant function with a value assigned to each element in the mesh, thus turning the problem into a finite-dimensional one. Since the piecewise function can only take one of two distinct values, the resulting problem would be a discrete optimization problem, which is significantly more difficult to solve than a continuous optimization problem [93]. We avoid this issue by replacing θ by a smooth approximation $\rho(\boldsymbol{x}) : \Omega \to [0, 1]$. Again, the values 0 and 1 correspond to void and solid regions of the design domain, but $\rho(\boldsymbol{x})$ also permits intermediate values which correspond to "grey" areas. The point-wise values of $\rho(\boldsymbol{x})$ do not always allow a strict physical interpretation, but are usually referred to as *density*. Depending on the context, it can be related to a porous micro-structure [21], for example, or the thickness of a sheet in two dimensions [110]. Especially in the latter case, the range of the density function might be generalized to non-negative lower and upper bounds $0 \leq \underline{\rho} \leq \rho(\boldsymbol{x}) \leq \overline{\rho}$ for all $\boldsymbol{x} \in \Omega$, with an upper bound $\overline{\rho}$ for the thickness that is larger than 1. Note that we will nevertheless use the common term "$0 - 1$ design" for a ρ that is equal to either

$\underline{\rho}$ or $\overline{\rho}$ everywhere in Ω.

The next question arising in the formulation of the optimization problem is that of the relationship between the density function $\rho(\boldsymbol{x})$ and the material properties. Before we address this question, note that strategies for parameterising the material boundary which are not based on a material distribution function, such as level-set methods [42], have also been investigated, see for example the references in [22].

As mentioned above, the problem is discretized at this stage, even before any optimization methods come into play. This means we follow the "discretize-then-optimize" approach, as is usually the case in topology design applications, rather than the alternative "optimize-then-discretize", which is more common, for example, in optimal control [66]. The FE discretization of the domain and the displacement function follows Section 2.2. The density function is approximated by a piecewise constant function: to each element with index $i \in \{1, \ldots, m\}$, we assign a density value $\rho_i \in [\underline{\rho}, \overline{\rho}]$. Throughout this thesis, we will mainly deal with the finite dimensional problem obtained in this way. For a complete discussion of "exact", infinite-dimensional structural optimization, including questions of existence of solutions and convergence of the discretized solution to the exact solution, see [22] and references therein. Some results specific to the virtual thickness sheet are referenced later in this section where appropriate.

The first numerical implementation employing the concept of material distribution was by Bendsøe and Kikuchi [21] and was based on homogenization of microstructures. Arguably the most popular approach for approximating a $0-1$ design by material distribution is the *solid isotropic material with penalization* (SIMP) approach [18, 19], in which the local stiffness is interpolated between 0 and 1 by a non-decreasing polynomial. This leads to a penalization of "grey" areas and thus to nearly "black-and-white" designs. If instead one interpolates the stiffness linearly one obtains the so-called variable thickness sheet problem, which we consider in this thesis. Given an FE discretization of the design domain, we can generally express the global stiffness matrix as a function of the element-wise densities

$$\mathbb{R}^m \ni \boldsymbol{\rho} \mapsto K(\boldsymbol{\rho}) \in \mathbb{R}^{n\times n},$$

where $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_m]^\top$ is the density vector. The function $K(\boldsymbol{\rho})$ is chosen such that $K(\boldsymbol{\rho})\ 0$ if $\rho_i > 0$ for all $i \in \{1, \ldots, m\}$. This ensures that the equilibrium equations have a unique solution for any $\rho > \mathbf{0}$.

Having defined a parameterization of the topology and a mapping that connects it to material properties, we now turn towards formulating the optimization problem. Our objective is to minimize the work done by the external forces, which is given by $\frac{1}{2}\boldsymbol{f}^\top\boldsymbol{u}$. If we consider that $\boldsymbol{u}$ should satisfy the equilibrium equations (2.38), we can see that this is equivalent to minimising the internal strain energy $\frac{1}{2}\boldsymbol{u}^\top K(\boldsymbol{\rho})\boldsymbol{u}$. This can be interpreted as a measure for the overall compliance – the reciprocal of the stiffness – of the design. Hence, the terms compliance minimization or, equivalently, stiffness maximization are commonly applied to optimization problems involving this objective.

Remark 2.3.1. We assume that the load vector $\boldsymbol{f}$ is not a function of the densities $\boldsymbol{\rho}$. This is an important assumption for all theoretical results in this section, in particular those regarding uniqueness of solutions. As a product of the FE discretization, the components of $\boldsymbol{f}$ stem either from boundary integrals of prescribed loads over Γ_f or from integrals over elements that are connected to nodes with prescribed non-zero displacements, see Section 2.2. The local stiffness tensor, and thus the densities, factor only into the latter. Therefore, we can ensure that $\boldsymbol{f}$ is independent of $\boldsymbol{\rho}$ by only prescribing load boundary conditions and homogeneous displacement boundary conditions. For prescribed non-zero displacements on Γ_u, the vector $\boldsymbol{f}$ contains integrals over elements that are adjacent to Γ_u. By fixing the densities in these elements, we can therefore get a density-independent $\boldsymbol{f}$ even for inhomogeneous displacement boundary conditions.

Remark 2.3.2. We also exclude the trivial case $\boldsymbol{f} = \mathbf{0}$. As mentioned in [97], any density distribution that satisfies the volume constraint is a non-unique (and non-isolated) solution in those cases.

We want to achieve maximum stiffness while limiting the amount of material, which we do by imposing a constraint on the overall mass as a function of $\boldsymbol{\rho}$. Since in many applications, the densities are interpreted as an approximation of a 0–1 design, no strict distinction between mass and volume is drawn, so that the aforementioned constraint is commonly called *volume constraint*. For consistency with the majority of the literature, we stick to this terminology and denote the constraint by $\mathrm{vol}(\boldsymbol{\rho}) = V$, where $V > 0$ is the amount of volume (or mass) that we allow our design to take up and $\mathrm{vol}(\boldsymbol{\rho})$ is, in the most general terms, a weighted sum of the densities. Together with the bounds on each element density mentioned earlier, we obtain the *minimum compliance problem* in its most common form:

$$\min_{\boldsymbol{\rho}\in\mathbb{R}^m,\boldsymbol{u}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} \tag{2.45a}$$

$$\text{s.t.} \quad \mathrm{K}(\boldsymbol{\rho})\,\boldsymbol{u} = \boldsymbol{f}, \tag{2.45b}$$

$$\mathrm{vol}(\boldsymbol{\rho}) = V, \tag{2.45c}$$

$$\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}}. \tag{2.45d}$$

The vectors $\underline{\boldsymbol{\rho}}$ and $\overline{\boldsymbol{\rho}}$ contain the lower and upper density bounds, which can in general be different for each element. All constraints on the density are chosen so that (2.45) is strictly feasible and physically plausible. First, we assume the bound constraints satisfy $\mathbf{0} \leq \underline{\boldsymbol{\rho}} < \overline{\boldsymbol{\rho}}$. Moreover, the volume limit $V > 0$ should be a fraction of the total volume of the design domain, that is $\mathrm{vol}(\overline{\boldsymbol{\rho}}) > V$, to avoid the trivial "all solid" solution $\boldsymbol{\rho} = \overline{\boldsymbol{\rho}}$. It should obviously also satisfy $\mathrm{vol}(\underline{\boldsymbol{\rho}}) < V$.

One might argue that the volume constraint (2.45c) should actually be an inequality constraint, since we only require an *upper bound* on the material used in the design. It should however become obvious when considering the problem from a physical point of view that any given design can always be made stiffer by adding more material at any point that is not yet solid. Therefore, no design that does not satisfy $\mathrm{vol}(\boldsymbol{\rho}) \leq V$ with equality is an optimal solution. For the special case of the variable thickness sheet, see Section 2.3.1, it can be shown that the problem with $\mathrm{vol}(\boldsymbol{\rho}) = V$ is equivalent to that with $\mathrm{vol}(\boldsymbol{\rho}) \leq V$, as the corresponding Lagrange multiplier is always non-negative at the optimal solution [6, (Proposition 3.2)].

We have introduced the minimum compliance topology optimization problem in a general form. It has a very straightforward physical interpretation, while nevertheless presenting an objective that is relevant to practical applications [3]. Although it has been extensively studied, it remains prevalent in the literature not just for this reason, but also as it is a useful starting point for research on new or improved solution methods. See [85, 25, 128, 83, 1, 109, 80, 96] for just a few examples. For the rest of the section, we will focus our attention on the variable thickness sheet problem which results from a particular choice of the density-to-stiffness mapping $\boldsymbol{\rho} \mapsto \mathrm{K}(\boldsymbol{\rho})$.

2.3.1 The Variable Thickness Sheet Problem

First studied in the context of minimum compliance topology optimization in [110], the *variable thickness sheet* (VTS) problem traditionally consists of finding the optimal thickness at each point of a two-dimensional sheet. It corresponds to (2.45) with specific choices of $\mathrm{K}(\boldsymbol{\rho})$ and $\mathrm{vol}(\boldsymbol{\rho})$, namely

$$\mathbf{0} \leq \boldsymbol{\rho} \mapsto \mathrm{K}(\boldsymbol{\rho}) = \sum_{i=1}^{m} \rho_i \mathrm{K}_i \in \mathbb{R}^{n\times n}, \tag{2.46}$$

where K_i is the elemental stiffness matrix of the ith element for a thickness of 1, and

$$\mathrm{vol}(\boldsymbol{\rho}) = \sum_{i=1}^{m} \rho_i a_i, \tag{2.47}$$

where a_i is the area of the ith element – or its volume, in the three-dimensional case. As explained in Section 2.2, the elemental stiffness matrices K_i are all positive semidefinite and the global stiffness matrix $K(\mathbf{1})$ is positive definite[1]. Since a different weighting in the assembly of all K_i in (2.46) does not change the range of the overall matrix, we can guarantee that $K(\boldsymbol{\rho}) \succeq 0$ as long as $\boldsymbol{\rho} \geq \mathbf{0}$ and, in particular, that $K(\boldsymbol{\rho}) \succ 0$ if $\boldsymbol{\rho} > \mathbf{0}$. Substituting (2.46) and (2.47) into (2.45), we obtain the following problem:

$$\min_{\boldsymbol{\rho}\in\mathbb{R}^m,\boldsymbol{u}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{f}^\mathsf{T}\boldsymbol{u} \tag{2.48a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \rho_i K_i \, \boldsymbol{u} = \boldsymbol{f}, \tag{2.48b}$$

$$\sum_{i=1}^{m} \rho_i a_i = V, \tag{2.48c}$$

$$\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}}, \quad i = 1, \dots, m\,. \tag{2.48d}$$

This problem has several interesting and useful properties, some of which will be discussed in this section. Some hold for any choice of lower bounds $\underline{\boldsymbol{\rho}} \geq \mathbf{0}$ for which (2.48) is feasible, others require a non-zero lower bound on the density everywhere. The latter is true for a result concerning uniqueness of the solution of (2.48). The same result also relies on a further assumption which we highlight once more, namely that the load vector $\boldsymbol{f}$ is not a function of the design variables, see Remark 2.3.1. Indeed, practical examples can be given for non-unique density solutions for $\underline{\boldsymbol{\rho}} > \mathbf{0}$ and $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{\rho})$, see [98].

The relevance of the variable thicknees sheet problem extends beyond the application that gave it its name. In the discretized form (2.48), it has strong similarities to compliance minimization in truss topology optimization, since the design variables map linearly to the stiffness matrix [22, (Section 1.5.2)]. As a consequence, many important results for the latter can be directly applied to (2.48).

[1] as long as the boundary conditions guarantee that the mechanical scenario is kinematically determinate, which we assume unless otherwise specified

Furthermore, our problem corresponds to a special case of free material optimization, in which the stiffness tensor is the design variable rather than ρ. The result will generally represent an inhomogeneous, anisotropic material optimized for the principal stress directions due to the given load cases [22]. In order to impose some kind of limit on the amount of material used at each point of the design, one can introduce ρ as a globally limited resource and use it as a local upper bound on either the trace or the Frobenius norm of the stiffness tensor. For both choices, the optimal local stiffness tensor can be determined analytically in terms of ρ in the special case of a single load case. This simplifies the problem such that it corresponds to the VTS problem for a material with a zero Poisson ratio [138].

Finally, the VTS formulation can also be interpreted as a particularly simple modelling of a $0-1$ design problem. On the one hand, nonlinear mappings $\mathrm{K}(\boldsymbol{\rho})$ are more widespread for this purpose, in particular the SIMP approach [22, 18], more recently used in conjunction with a relaxed Heaviside projection [50, 126]. Such mappings effectively avoid large "grey" areas of intermediate density values. On the other hand, the much simpler structure of the VTS problem allows for the use of much more efficient solution algorithms. It can therefore at the very least be used to provide a lower bound on the minimum compliance, obtainable at comparatively low computational cost.

We now turn our attention to a number of theoretical results for the VTS problem. We first study the existence and uniqueness of solutions. We then devote a part of this section to equivalent formulations which can be interpreted as the dual of (2.48). These are of special importance for an efficient solution algorithm proposed in Chapter 3. Note that most results given in this section are not new contributions. We merely present and prove them in a way that is tailored to our specific problem, giving references to the original sources where appropriate.

Existence and uniqueness results for the infinite-dimensional, or "exact", VTS problem are summarized in [22, (p. 272–274)]. The existence of a solution for $\underline{\boldsymbol{\rho}} > \mathbf{0}$

is a standard result in optimal control [38]. It can be extended to $\underline{\boldsymbol{\rho}} \geq \mathbf{0}$ by using a min-max formulation of the problem [22, (ibid)], similar to that employed further below for the discretized case. Furthermore, Petersson [97] showed that, for $\underline{\boldsymbol{\rho}} > \mathbf{0}$, the displacements are unique, although the densities may not be. For the finite-dimensional VTS problem and under mild assumptions, we will show that a solution exists and both the optimal displacements and densities are unique.

The starting point for all major results in this section is a reformulation of (2.48) as a saddle point problem. It forms the basis of many theoretical existence and uniqueness results, both in finite and infinite dimensions, see, e.g. [97]. We lay it out in the following lemma, which covers the case $\underline{\boldsymbol{\rho}} \geq \mathbf{0}$. Note that $\mathrm{K}(\boldsymbol{\rho})$ can become singular if $\underline{\rho}_i = 0$ for some $i = 1, \dots, m$ and thus have a non-trivial null space. Our uniqueness theorem later in this section will require $\underline{\boldsymbol{\rho}} > \mathbf{0}$.

Lemma 2.3.3. *Assume that there exists at least one* $\boldsymbol{\rho}$ *feasible for (2.48) such that* $\boldsymbol{f}$ *is in the range of* $\mathrm{K}(\boldsymbol{\rho})$*. Then, (2.48) is equivalent to the saddle point problem*

$$\min_{\substack{\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}} \\ \sum_{i=1}^{m} \rho_i a_i = V}} \max_{\boldsymbol{u} \in \mathrm{R}^n} \boldsymbol{f}^{\mathsf{T}} \boldsymbol{u} - \frac{1}{2} \boldsymbol{u}^{\mathsf{T}} \mathrm{K}(\boldsymbol{\rho}) \boldsymbol{u}. \tag{2.49}$$

Proof. The proof we give here can also be found in [16] as part of the proof of Theorem 1. For any fixed, feasible $\boldsymbol{\rho}$, the matrix $\mathrm{K}(\boldsymbol{\rho})$ is by assumption positive semidefinite. The function $\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u}$ is therefore concave in $\boldsymbol{u}$. In order for the supremum

$$s(\boldsymbol{\rho}) := \sup_{\boldsymbol{u} \in \mathrm{R}^n} \boldsymbol{f}^{\mathsf{T}} \boldsymbol{u} - \frac{1}{2} \boldsymbol{u}^{\mathsf{T}} \mathrm{K}(\boldsymbol{\rho}) \boldsymbol{u}$$

to be attained at some $\boldsymbol{u}^*$, this vector needs to satisfy the necessary and sufficient optimality condition $\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u}^* = \boldsymbol{f}$. The maximal value in that case is $\frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u}^*$. If, on the other hand, the supremum cannot be attained for the chosen $\boldsymbol{\rho}$, then it is because $\boldsymbol{f}$ lies in the non-trivial null space of $\mathrm{K}(\boldsymbol{\rho})$ and we can choose $\boldsymbol{u}$ as any element of that null space to make $\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} \neq 0$ arbitrarily large. Since, by our assumption, there is at least one $\boldsymbol{\rho}$ such that $\boldsymbol{f}$ does not lie in the null space of $\mathrm{K}(\boldsymbol{\rho})$, the problem

$$\min_{\substack{\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}} \\ \sum_{i=1}^{m} \rho_i a_i = V}} s(\boldsymbol{\rho})$$

is well defined and we can write "max" rather than "sup" in $s(\boldsymbol{\rho})$, giving us (2.49). We can then use the optimality condition $\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$ to resolve the inner maximization and show that it is indeed equivalent to problem (2.48). □

Recall that the term $\Pi^*(\boldsymbol{\rho}) = \min_{\boldsymbol{u}} \frac{1}{2}\boldsymbol{u}^\top \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f}^\top \boldsymbol{u}$ is the potential energy of the elastic body at equilibrium, as shown in Section 2.2. If we switch the sign of the objective function in (2.49), and thereby turn the min-max into a max-min problem, we can see that compliance minimization can be viewed as a maximization of Π^* over all feasible designs. At the same time, compliance minimization is equivalent to stiffness maximization. For this reason, the potential energy is often, although somewhat vaguely, interpreted as a measure of stiffness.

An important property of (2.49) is that the inner maximum can be interpreted as a convex function of $\boldsymbol{\rho}$. This leads to the following lemma, which can also be found in [4].

Lemma 2.3.4. *The saddle point problem (2.49) is equivalent to a convex minimization problem in* $\boldsymbol{\rho}$*.*

Proof. Let us denote the objective function of the saddle point problem as

$$f_{\boldsymbol{u}}(\boldsymbol{\rho}) := \boldsymbol{f}^\top \boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^\top \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}^\top \boldsymbol{u} - \frac{1}{2}\sum_{i=1}^{m} \rho_i \left(\boldsymbol{u}^\top \mathrm{K}_i \boldsymbol{u}\right),$$

that is, we view it as a family of functions in the variable $\boldsymbol{\rho}$. For any $\bar{\boldsymbol{u}} \in \mathbb{R}^n$, the function $f_{\bar{\boldsymbol{u}}}(\boldsymbol{\rho})$ is linear and thus convex in $\boldsymbol{\rho}$. The point-wise maximum of a finite or infinite number of convex functions is also convex [68, (Proposition B.2.1.2)]. In other words, if we denote the inner maximum of (2.49) by

$$s(\boldsymbol{\rho}) := \max_{\boldsymbol{u}\in\mathbb{R}^n} \boldsymbol{f}^\mathsf{T}\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \max_{\boldsymbol{u}\in\mathbb{R}^n} f_{\boldsymbol{u}}(\boldsymbol{\rho})\,,$$

then the function $s(\boldsymbol{\rho})$ is convex. Furthermore, (2.49) can be written as

$$\begin{aligned} \min_{\boldsymbol{\rho}\in\mathbb{R}^m} \quad & s(\boldsymbol{\rho}) \\ \text{s.t.} \quad & \sum_{i=1}^{m} \rho_i a_i = V \\ & \underline{\boldsymbol{\rho}} \le \boldsymbol{\rho} \le \overline{\boldsymbol{\rho}}\,. \end{aligned}$$

The volume and bound constraints define a convex polyhedral feasible set which, together with the convex objective function, gives us a convex optimization problem.

□

For later reference, we apply the standard optimality conditions discussed in Section 2.1.1 to (2.48). Matching the general nonlinear optimization notation in (2.2) to that in (2.48), we have the optimization variables $\boldsymbol{x} = (\boldsymbol{\rho}, \boldsymbol{u})$, the objective function $f(\boldsymbol{x}) = f(\boldsymbol{\rho}, \boldsymbol{u}) = \frac{1}{2}\boldsymbol{f}^\mathsf{T}\boldsymbol{u}$, the inequality constraint functions $\boldsymbol{g}(\boldsymbol{\rho}, \boldsymbol{u})_{\{i=1,\ldots,m\}} = \boldsymbol{\rho} - \overline{\boldsymbol{\rho}}$ and $\boldsymbol{g}(\boldsymbol{\rho}, \boldsymbol{u})_{\{i=m+1,\ldots,2m\}} = \underline{\boldsymbol{\rho}} - \boldsymbol{\rho}$, and the equality constraint functions $\boldsymbol{h}(\boldsymbol{\rho}, \boldsymbol{u})_{\{1,\ldots,n\}} = \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f}$ and $h_{n+1}(\boldsymbol{\rho}, \boldsymbol{u}) = \sum \rho_i a_i - V$. We omit the displacements $\boldsymbol{u}$ from the notation of the active set $\mathcal{A}(\boldsymbol{\rho})$, since they are not featured in the inequality constraints. The gradients for the constraint functions are

$$\nabla\boldsymbol{g}(\boldsymbol{\rho}, \boldsymbol{u}) = \begin{bmatrix} I & -I \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \nabla\boldsymbol{h}(\boldsymbol{\rho}, \boldsymbol{u}) = \begin{bmatrix} \mathrm{B}(\boldsymbol{u})^\mathsf{T} & \boldsymbol{a} \\ \mathrm{K}(\boldsymbol{\rho}) & \boldsymbol{0} \end{bmatrix}, \tag{2.50}$$

where $\boldsymbol{a} = (a_1, \ldots, a_m)^\mathsf{T}$ is the vector of element areas/volumes and the short hand notation

$$\mathrm{B}(\boldsymbol{u}) := [\mathrm{K}_1\boldsymbol{u}, \ldots, \mathrm{K}_m\boldsymbol{u}]$$

has been introduced for readability. Also note that we have used the symmetry of

$\mathrm{K}(\boldsymbol{\rho})$. We can now set up the KKT conditions for the VTS problem. Let us denote the Lagrange multipliers for the different sets of constraints in the VTS problem by $\boldsymbol{\mu}, \lambda, \underline{\boldsymbol{r}}$ and $\overline{\boldsymbol{r}}$, so that the Lagrangian is given by

$$\begin{aligned}
L(\boldsymbol{\rho}, \boldsymbol{u}, \boldsymbol{\mu}, \lambda, \underline{\boldsymbol{r}}, \overline{\boldsymbol{r}}) &= f(\boldsymbol{\rho}, \boldsymbol{u}) + \boldsymbol{h}(\boldsymbol{\rho}, \boldsymbol{u})^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\mu} \\ \lambda \end{pmatrix} + \boldsymbol{g}(\boldsymbol{\rho}, \boldsymbol{u})^{\mathsf{T}} \begin{pmatrix} \underline{\boldsymbol{r}} \\ \overline{\boldsymbol{r}} \end{pmatrix} \\
&= \frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} + (\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f})^{\mathsf{T}} \boldsymbol{\mu} \\
&\quad + \Big(\sum_i \rho_i a_i - V\Big) \lambda + (\boldsymbol{\rho} - \overline{\boldsymbol{\rho}})^{\mathsf{T}} \overline{\boldsymbol{r}} + \big(\underline{\boldsymbol{\rho}} - \boldsymbol{\rho}\big)^{\mathsf{T}} \underline{\boldsymbol{r}}.
\end{aligned}$$

Accordingly, the KKT conditions comprise the constraints (2.48b) and (2.48c) and the following equations:

$$\nabla_{\boldsymbol{\rho}} L = \mathrm{B}(\boldsymbol{u})^{\mathsf{T}} \boldsymbol{\mu} + \lambda \boldsymbol{a} + \overline{\boldsymbol{r}} - \underline{\boldsymbol{r}} = \mathbf{0}, \tag{2.51a}$$

$$\nabla_{\boldsymbol{u}} L = \mathrm{K}(\boldsymbol{\rho})\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{f} = \mathbf{0}, \tag{2.51b}$$

$$(\boldsymbol{\rho} - \overline{\boldsymbol{\rho}})^{\mathsf{T}} \underline{\boldsymbol{r}} = 0, \tag{2.51c}$$

$$\big(\underline{\boldsymbol{\rho}} - \boldsymbol{\rho}\big)^{\mathsf{T}} \overline{\boldsymbol{r}} = 0, \tag{2.51d}$$

$$(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho}),\ \big(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}\big),\ \underline{\boldsymbol{r}}, \overline{\boldsymbol{r}} \geq \mathbf{0}. \tag{2.51e}$$

Comparing (2.51b) and (2.48b), we can see that

$$\boldsymbol{\mu} = -\frac{1}{2}\boldsymbol{u}. \tag{2.52}$$

This identity allows us to eliminate $\boldsymbol{\mu}$ and the set of equations (2.51b), which greatly simplifies the task of solving the VTS problem numerically, see Section 3.1. In fact, the identity also holds for the general minimum compliance problem (2.45). It is a *self-adjoint* [1] problem, meaning that the *adjoint variables* $\boldsymbol{\mu}$ are a scalar multiple of the displacements $\boldsymbol{u}$.

We now ask whether the constraint qualification from Definition 2.1.3 is satisfied

[1]In the context of shape optimization and optimal control, see for example [112, 84], (2.48b) are called the *state equations* and $\boldsymbol{u}$ the *state variables*, while (2.51b) are the *adjoint equations* and $\boldsymbol{\mu}$ the *adjoint variables*.

at a solution candidate for (2.48). Beyond the first order optimality conditions, the LICQ is of importance in convergence results for some optimization algorithms, in particular the nonlinear rescaling methods discussed in Section 2.1.4.

Lemma 2.3.5. *Assume $(\boldsymbol{\rho}, \boldsymbol{u})$ is feasible for (2.48) and $0 < \underline{\rho_i} < \rho_i < \overline{\rho}_i$ for at least one $i \in \{1, \dots, m\}$. Then, the LICQ is satisfied at $(\boldsymbol{\rho}, \boldsymbol{u})$.*

Proof. We start by showing that the gradients $\nabla \boldsymbol{g}_i(\boldsymbol{\rho}, \boldsymbol{u})$, $i \in \mathrm{A}(\boldsymbol{\rho})$, of active inequality constraints are linearly independent and that $\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}(\boldsymbol{\rho}, \boldsymbol{u})$ has full rank smaller than m. We omit the variables $(\boldsymbol{\rho}, \boldsymbol{u})$ in all functions for the rest of the proof. Because our problem is strictly feasible by assumption, i.e. $\underline{\boldsymbol{\rho}} < \overline{\boldsymbol{\rho}}$, no upper and lower bound constraint on the same ρ_i can be active at the same time. Therefore, we have $i \in \mathrm{A}(\boldsymbol{\rho}) \Rightarrow i + m \notin \mathrm{A}(\boldsymbol{\rho})$ and vice versa, for all $i \in \{1, \dots, m\}$. We can see from (2.50), that of any two linearly dependent columns in $\nabla \boldsymbol{g}$ – so of each pair of positive and negative unity vectors – at most one can be present in $\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}$. The remaining columns are all linearly independent. Furthermore, since at least one ρ_i is strictly feasible by our assumption, $\mathrm{rank}\{\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}\} = |\mathrm{A}(\boldsymbol{\rho})| < m$.

We continue to assemble $[\nabla \boldsymbol{h}, \nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}]$, starting by adding the right most column of $\nabla \boldsymbol{h}$ to $\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}$, which contains only the vector $\boldsymbol{a} > \mathbf{0}$ in the upper block, see (2.50). Since, by the assumption that at least one ρ_i is strictly feasible, there is at least one zero row of $\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}$, and therefore one $i \in \{1, \dots, m\}$ for which $(\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})} \boldsymbol{v})_i = \mathbf{0}^\mathsf{T} \boldsymbol{v} = 0 \neq a_i$ for any $\boldsymbol{v} \neq \mathbf{0}$. Hence, the vector $\boldsymbol{a}$ is linearly independent of $\nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}$. As for the remaining part of $\nabla \boldsymbol{h}$, we know that $\mathrm{rank}\{[\mathrm{B}, \mathrm{K}]^\mathsf{T}\} = n$ because K is positive definite. That it is linearly independent of the rest of $[\nabla \boldsymbol{h}, \nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}]$ is evident from the fact that its column vectors are not coplanar with the other columns. Hence, $[\nabla \boldsymbol{h}, \nabla \boldsymbol{g}_{\mathrm{A}(\boldsymbol{\rho})}]$ has full rank $|\mathrm{A}(\boldsymbol{\rho})| + 1 + n \leq m + n$ and the LICQ is satisfied. □

To show uniqueness of the solution of (2.48), we will make use of the second order sufficient conditions, see Theorem 2.1.6, which the following lemma tells us are satisfied at each solution of (2.48).

Lemma 2.3.6. *If $\underline{\boldsymbol{\rho}} > \mathbf{0}$, every KKT point $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ of (2.48) satisfies the second order sufficient optimality conditions (2.10) and is therefore an isolated local minimum.*

Proof. The first step is to construct the critical cone defined in (2.9). Let

$$(\boldsymbol{\rho}^*, \boldsymbol{u}^*, \boldsymbol{\mu}^*, \lambda^*, \underline{\boldsymbol{r}}^*, \overline{\boldsymbol{r}}^*)$$

be a KKT tuple of the VTS problem. The critical cone $\mathrm{C}(\boldsymbol{\rho}^*, \boldsymbol{u}^*, \underline{\boldsymbol{r}}^*, \overline{\boldsymbol{r}}^*)$ is given by

$$\mathrm{C} = \left\{ \boldsymbol{d} = \begin{pmatrix} \boldsymbol{d}_{\boldsymbol{\rho}} \\ \boldsymbol{d}_{\boldsymbol{u}} \end{pmatrix} \in \mathbb{R}^{m+n} : \begin{array}{l} \begin{bmatrix} \mathrm{B}(\boldsymbol{u}^*) & \mathrm{K}(\boldsymbol{\rho}^*) \\ \boldsymbol{a}^\top & \mathbf{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{d}_{\boldsymbol{\rho}} \\ \boldsymbol{d}_{\boldsymbol{u}} \end{pmatrix} = \mathbf{0}, \\ (\boldsymbol{d}_{\boldsymbol{\rho}})_i \begin{cases} = 0 & \forall i : \rho_i^* = \overline{\rho},\ \overline{r}_i^* > 0 \text{ or } \\ & \rho_i^* = \underline{\rho},\ \underline{r}_i^* > 0 \\ \leq 0 & \forall i : \rho_i^* = \overline{\rho},\ \overline{r}_i^* = 0 \\ \geq 0 & \forall i : \rho_i^* = \underline{\rho},\ \underline{r}_i^* = 0 \end{cases} \end{array} \right\}.$$

We will only need the first set of equations in the above set, which gives us

$$\mathrm{B}(\boldsymbol{u}^*)\, \boldsymbol{d}_{\boldsymbol{\rho}} = -\mathrm{K}(\boldsymbol{\rho}^*)\, \boldsymbol{d}_{\boldsymbol{u}} \quad \forall \begin{pmatrix} \boldsymbol{d}_{\boldsymbol{\rho}} \\ \boldsymbol{d}_{\boldsymbol{u}} \end{pmatrix} \in \mathrm{C}\,. \tag{2.53}$$

Next, we need to construct the Hessian of the Lagrangian. For the diagonal blocks, we have $\nabla^2_{\boldsymbol{\rho}} L = 0$ and $\nabla^2_{\boldsymbol{u}} L = 0$. The only non-zero parts are the off-diagonal blocks. The lower left block is $\nabla_{(\boldsymbol{u},\boldsymbol{\rho})} L = [\mathrm{K}_1\boldsymbol{\mu}, \ldots, \mathrm{K}_m\boldsymbol{\mu}]$, the upper right block its transpose, due to symmetry. Using (2.52), the Hessian resolves to

$$\nabla^2_{(\boldsymbol{\rho},\boldsymbol{u})} L = -\frac{1}{2} \begin{bmatrix} 0 & \mathrm{B}(\boldsymbol{u})^\top \\ \mathrm{B}(\boldsymbol{u}) & 0 \end{bmatrix}.$$

Now let $\boldsymbol{d} = (\boldsymbol{d}_{\boldsymbol{\rho}}, \boldsymbol{d}_{\boldsymbol{u}}) \in \mathrm{C}(\boldsymbol{\rho}^*, \boldsymbol{u}^*, \underline{\boldsymbol{r}}^*, \overline{\boldsymbol{r}}^*)$. Multiplying the Hessian of the Lagrangian by $\boldsymbol{d}$ from both sides and using (2.53), we get

$$\begin{aligned}\boldsymbol{d}^\top \nabla^2 L\, \boldsymbol{d} &= -\frac{1}{2}\Big(\boldsymbol{d}_{\boldsymbol{\rho}}^\top \mathrm{B}(\boldsymbol{u}^*)^\top \boldsymbol{d}_{\boldsymbol{u}} + \boldsymbol{d}_{\boldsymbol{u}}^\top \mathrm{B}(\boldsymbol{u}^*)\, \boldsymbol{d}_{\boldsymbol{\rho}} \Big)\\ &= -\frac{1}{2}\Big(-\boldsymbol{d}_{\boldsymbol{u}}^\top \mathrm{K}(\boldsymbol{\rho}^*)^\top \boldsymbol{d}_{\boldsymbol{u}} - \boldsymbol{d}_{\boldsymbol{u}}^\top \mathrm{K}(\boldsymbol{\rho}^*)\, \boldsymbol{d}_{\boldsymbol{u}} \Big)\\ &= \boldsymbol{d}_{\boldsymbol{u}}^\top \mathrm{K}(\boldsymbol{\rho}^*)\, \boldsymbol{d}_{\boldsymbol{u}}\,.\end{aligned}$$

Since $\mathbf{0} < \underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho}^*$, we know that $\mathrm{K}(\boldsymbol{\rho}^*)$ is positive definite. Thus, we have

$$\boldsymbol{d}^\top \nabla^2 L\, \boldsymbol{d} = \boldsymbol{d}_{\boldsymbol{u}}^\top \mathrm{K}(\boldsymbol{\rho}^*)\, \boldsymbol{d}_{\boldsymbol{u}} > 0 \quad \forall\, \boldsymbol{d} \in \mathrm{C}(\boldsymbol{\rho}^*, \boldsymbol{u}^*, \underline{\boldsymbol{r}}^*, \overline{\boldsymbol{r}}^*),\ \boldsymbol{d} \neq \mathbf{0}\,,$$

which concludes the proof. □

We can now combine all previous lemmas to show that the finite-dimensional VTS problem with non-zero lower density bounds has a unique solution under very reasonable assumptions.

Theorem 2.3.7. *Consider problem (2.48) for the case* $\underline{\boldsymbol{\rho}} > \mathbf{0}$. *Assume that a solution* $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ *of (2.48) satisfies* $\underline{\rho}_i < \rho_i^* < \overline{\rho}_i$ *for at least one* $i \in \{1, \dots, m\}$. *Then, the solution is unique.*

Proof. According to Lemma 2.3.5, the LICQ is satisfied because there is at least one strictly feasible density ρ_i^*. This, in turn, means that $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ satisfies the KKT conditions, so that, due to Lemma 2.3.6, it is an isolated solution. Lemma 2.3.3 together with Lemma 2.3.4 therefore tell us that $\boldsymbol{\rho}^*$ is an isolated local minimum of an equivalent convex problem. Since every local solution of a convex minimization problem is a global solution, the fact that it is isolated in this case means it is unique. Finally, because $\boldsymbol{u}^*$ is uniquely defined by $\boldsymbol{u}^* = \mathrm{K}(\boldsymbol{\rho}^*)^{-1}\boldsymbol{f}$, the solution $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ of (2.48) is unique. □

The requirement of an optimal solution to contain at least one "grey" element is easily satisfied. All of our numerical examples show large areas of intermediate density values, suggesting that these are a typical feature of optimal VTS designs. A strict $0-1$ solution is more likely to be seen for a zero Poisson ratio, $\nu = 0$. Even then, it is only possible if solid and (nearly) void elements in the design domain can sum up exactly to the volume constraint constant V, i.e. there exists a subset of elements $I \subset \{1,\dots,n\}$ such that $\sum_{i\in I}\bar{\rho}_i a_i + \sum_{i\notin I}\underline{\rho}_i a_i = V$. If this is indeed the case, a small perturbation of the volume constraint $V+\epsilon_V$ for a small enough ϵ_V will no longer allow a $0-1$ solution for the perturbed VTS problem, thereby implying uniqueness of the solution.

We have presented a uniqueness result for the variable thickness problem. Next, we will cover some duality results which will form the basis of a solution algorithm in Chapter 3.

The dual VTS problem for a non-zero lower density bound

In the following, we introduce an optimization problem which is equivalent to (2.48). We will show that the optimal solution of this problem provides an optimal solution for the VTS problem as well as the associated Lagrange multipliers. We will assume that the lower bounds $\underline{\boldsymbol{\rho}}$ are strictly greater than 0. We consider the case $\underline{\boldsymbol{\rho}} = \mathbf{0}$ at the end of the section.

All of the following results are due to Achtziger et al. [6] and Ben-Tal and Bendsøe [16], as well as Klarbring, Petersson, and Rönnqvist [75] for the case of unilateral contact, which is discussed in Section 2.3.2.

Following [16, 78] in the context of equivalent formulations for truss topology optimization, we introduce the following minimization problem:

$$\min_{\substack{u\in\mathbb{R}^n,\ \alpha\in\mathbb{R},\ \underline{\nu},\overline{\nu}\in\mathbb{R}^m}} \quad \alpha V - \boldsymbol{f}^\top \boldsymbol{u} - \underline{\boldsymbol{\rho}}^\top \underline{\boldsymbol{\nu}} + \overline{\boldsymbol{\rho}}^\top \overline{\boldsymbol{\nu}} \tag{2.54a}$$

$$\text{s.t.} \quad \frac{1}{2}\boldsymbol{u}^\top \mathrm{K}_i \boldsymbol{u} \le \alpha\, a_i - \underline{\nu}_i + \overline{\nu}_i, \quad i = 1, \dots, m, \tag{2.54b}$$

$$\underline{\nu}_i \ge 0, \quad i = 1, \dots, m, \tag{2.54c}$$

$$\overline{\nu}_i \ge 0, \quad i = 1, \dots, m. \tag{2.54d}$$

The next result can essentially be traced back to [16], although it is more of a by-product of other results of that paper.

Theorem 2.3.8. *Assume that the set of feasible $\boldsymbol{\rho}$ for (2.48) is strictly feasible, that is $\underline{\boldsymbol{\rho}} < \overline{\boldsymbol{\rho}}$. Then, problems (2.48) and (2.54) are equivalent in the following sense:*

(i) If one problem has a solution, so does the other and

$$\min(2.48) = -\min(2.54)\ .$$

(ii) Let $(\boldsymbol{u}^*, \alpha^*, \underline{\boldsymbol{\nu}}^*, \overline{\boldsymbol{\nu}}^*)$ be a solution of (2.54). Further, let $\boldsymbol{\tau}^*$ be the vector of associated Lagrange multipliers for the inequality constraints (2.54b). Then $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$ is a solution of (2.48). Moreover, $\alpha^*, \underline{\boldsymbol{\nu}}^*, \overline{\boldsymbol{\nu}}^*$ are the Lagrange multipliers associated with this solution for the volume and bound constraints, respectively.

(iii) Let $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ be a solution of (2.48). Further, let $\underline{\boldsymbol{r}}^*$ and $\overline{\boldsymbol{r}}^*$ be the Lagrange multipliers for the lower and upper bounds on $\boldsymbol{\rho}$, respectively, and let λ^* be the multiplier for the volume constraint. Then $(\boldsymbol{u}^*, \lambda^*, \underline{\boldsymbol{r}}^*, \overline{\boldsymbol{r}}^*)$ is a solution of (2.54). Moreover, $\boldsymbol{\rho}^*$ are the Lagrange multipliers associated with this solution for the inequality constraints (2.54b).

Remark 2.3.9. Before we prove Theorem 2.3.8, let us make the observation that it very much resembles a duality theorem. However, the two equivalent problems do not completely display a duality structure. Firstly, both problems are formulated as minimization problems. Secondly, $\boldsymbol{u}$ is an optimization variable in both problems. Lastly, while the solution to problem (2.54) provides a tuple of optimal solution and Lagrange multipliers to problem (2.48), the converse is not true. However, the main result in the context of this thesis lies in the fact that we can solve (2.54) instead of (2.48). Furthermore, as will be shown later in this section, we can use the solution to easily compute the difference between the objective function values of both problems for a given solution – the duality gap, as it were – which provides a measure of optimality due to Theorem 2.3.8(i). For this reason, we refer to (2.54) as the *dual* of the VTS problem.

Proof. We know from Lemma 2.3.3, that (2.48) is equivalent to the saddle point problem (2.49). This problem is convex (actually linear) and bounded in $\boldsymbol{\rho}$, and concave in $\boldsymbol{u}$, so we can swap "max" and "min", see for example [108, (Cor. 37.3.2)], to get

$$\max_{\boldsymbol{u}\in\mathbb{R}^n} \min_{\substack{\underline{\boldsymbol{\rho}}\le\boldsymbol{\rho}\le\overline{\boldsymbol{\rho}} \\ \sum_{i=1}^m \rho_i a_i = V}} \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u}. \tag{2.55}$$

Due to our assumption of strict feasibility, there exists a Slater point for the feasible set of the inner (convex) optimization problem, so we may replace it by its Lagrangian dual. The Lagrange multipliers for the inequality constraints will be denoted by $\underline{\boldsymbol{r}} \ge \mathbf{0}$ and $\overline{\boldsymbol{r}} \ge \mathbf{0}$, that for the volume (equality) constraint by $\lambda \in \mathbb{R}$:

$$\begin{aligned}\max_{\boldsymbol{u}\in\mathbb{R}^n} \max_{\substack{\lambda\in\mathbb{R} \\ \underline{\boldsymbol{r}},\overline{\boldsymbol{r}}\ge\mathbf{0}}} \min_{\boldsymbol{\rho}\ge\mathbf{0}} \; & \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} \\ & + \lambda\left(\sum_{i=1}^m \rho_i a_i - V\right) - \underline{\boldsymbol{r}}^{\mathsf{T}}(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}) + \overline{\boldsymbol{r}}^{\mathsf{T}}(\boldsymbol{\rho} - \overline{\boldsymbol{\rho}}).\end{aligned} \tag{2.56}$$

We can include the non-negativity constraint on $\boldsymbol{\rho}$ in the innermost optimization problem since we know that its solution, being the solution of (2.55), satisfies $\boldsymbol{\rho} \ge \underline{\boldsymbol{\rho}} \ge \mathbf{0}$.

Now regard the dual problem (2.54). It can equivalently be formulated as the following min-max problem, using a partial Lagrangian function with multipliers $\boldsymbol{\tau} \geq \mathbf{0}$:

$$\min_{\substack{\boldsymbol{u}\in\mathbb{R}^n,\ \alpha\in\mathbb{R}\\ \underline{\boldsymbol{\nu}},\overline{\boldsymbol{\nu}}\geq\mathbf{0}}} \max_{\boldsymbol{\tau}\geq\mathbf{0}} \ \alpha V - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \underline{\boldsymbol{\nu}}^{\mathsf{T}}\underline{\boldsymbol{\rho}} + \overline{\boldsymbol{\nu}}^{\mathsf{T}}\overline{\boldsymbol{\rho}} + \sum_{i=1}^{m} \tau_i\left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i + \underline{\nu}_i - \overline{\nu}_i\right)$$

which can be rearranged further to give

$$\begin{aligned}\min_{\substack{\boldsymbol{u}\in\mathbb{R}^n,\ \alpha\in\mathbb{R}\\ \underline{\boldsymbol{\nu}},\overline{\boldsymbol{\nu}}\geq\mathbf{0}}} \max_{\boldsymbol{\tau}\geq\mathbf{0}} \ & \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\tau})\boldsymbol{u} - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} \\ & + \alpha\left(V - \sum_{i=1}^{m}\tau_i a_i\right) + \underline{\boldsymbol{\nu}}^{\mathsf{T}}(\boldsymbol{\tau} - \underline{\boldsymbol{\rho}}) - \overline{\boldsymbol{\nu}}^{\mathsf{T}}(\boldsymbol{\tau} - \overline{\boldsymbol{\rho}}).\end{aligned} \tag{2.57}$$

Identifying $\boldsymbol{\tau}$, α, $\underline{\boldsymbol{\nu}}$, and $\overline{\boldsymbol{\nu}}$ with $\boldsymbol{\rho}$, λ, $\underline{\boldsymbol{r}}$, and $\overline{\boldsymbol{r}}$, respectively, and changing the sign of the objective function (and thus changing "max" to "min" and "min" to "max"), we can see that (2.56) and (2.57) are equivalent. All three claims follow from this. □

Remark 2.3.10. The dual problem (2.54) is a convex optimization problem. Indeed, most functions are linear. The quadratic inequality constraint functions (2.54b) feature the elemental stiffness matrices K_i, which are positive semidefinite, making these constraint functions convex as well.

Remark 2.3.11. In later sections, we will often use the dual variables α, $\underline{\boldsymbol{\nu}}$, $\overline{\boldsymbol{\nu}}$ rather than the Lagrange multipliers λ, $\underline{\boldsymbol{r}}$, $\overline{\boldsymbol{r}}$, since they are equal at the solution. Apart from where the distinction is important, we can safely ignore it for the sake of decluttering the notation.

We end this section with another formulation of the dual VTS problem, which only features the variables $\boldsymbol{u}$ and α. It allows us to obtain an expression for the duality gap without the Lagrange multipliers $\underline{\boldsymbol{\nu}}$, $\overline{\boldsymbol{\nu}}$, which can be used as an optimality measure in methods that solve the primal VTS problem (2.48) and return only $\boldsymbol{\rho}$, $\boldsymbol{u}$ and α as the solution. In particular, we will use this optimality measure for the stopping criterion of the standard method outlined in Section 2.3.3 and for comparing its results to those of the primal-dual methods in Section 3.1. The following result was first derived in [16].

Theorem 2.3.12. *Problem (2.54) is equivalent to the unconstrained nonsmooth problem*

$$-\max_{\boldsymbol{u}\in\mathbb{R}^n,\alpha\in\mathbb{R}} -\alpha V + \boldsymbol{f}^{\top}\boldsymbol{u} - \sum_{i=1}^{m} \max\left(\left(\frac{1}{2}\boldsymbol{u}^{\top}K_i\boldsymbol{u} - \alpha a_i\right)\underline{\rho}_i, \left(\frac{1}{2}\boldsymbol{u}^{\top}K_i\boldsymbol{u} - \alpha a_i\right)\overline{\rho}_i\right) \tag{2.58}$$

in the following sense:

(i) $\min(2.54) = -\max(2.58)$,

(ii) Let $(\boldsymbol{u}^*, \alpha^*, \underline{\boldsymbol{v}}^*, \overline{\boldsymbol{v}}^*)$ *be a solution of (2.54). Then* $(\boldsymbol{u}^*, \alpha^*)$ *is a solution of (2.58). Conversely, every solution* $(\boldsymbol{u}^*, \alpha^*)$ *of (2.58) is also (part of) a solution of (2.54).*

Proof. We will show that (2.54) and (2.58) are equivalent reformulations of each other. Introducing an auxiliary variable $\boldsymbol{s} \in \mathbb{R}^m$, problem (2.58) can be directly rewritten as

$$\begin{aligned}
\max_{\boldsymbol{u}\in\mathbb{R}^n,\alpha\in\mathbb{R},\boldsymbol{s}\in\mathbb{R}^m} \quad & -\alpha V + \boldsymbol{f}^{\top}\boldsymbol{u} - \sum_{i=1}^{m} s_i \\
\text{s.t.} \quad & \left(\frac{1}{2}\boldsymbol{u}^{\top}K_i\boldsymbol{u} - \alpha a_i\right)\overline{\rho}_i \le s_i, \quad i = 1,\dots,m, \\
& \left(\frac{1}{2}\boldsymbol{u}^{\top}K_i\boldsymbol{u} - \alpha a_i\right)\underline{\rho}_i \le s_i, \quad i = 1,\dots,m.
\end{aligned}$$

The constraints in the above problem can be written as

$$\frac{1}{2}\boldsymbol{u}^{\top}K_i\boldsymbol{u} - \alpha a_i \le \min\left(\frac{s_i}{\underline{\rho}_i}, \frac{s_i}{\overline{\rho}_i}\right), \quad i = 1,\dots,m.$$

Noting that $\overline{\rho}_i > \underline{\rho}_i > 0$, we define

$$\begin{aligned}
\underline{\nu}_i &= 0 && \text{and} \quad \overline{\nu}_i = \frac{s_i}{\overline{\rho}_i}, && \text{if} \quad \frac{s_i}{\underline{\rho}_i} > \frac{s_i}{\overline{\rho}_i} > 0, \\
\underline{\nu}_i &= -\frac{s_i}{\underline{\rho}_i} && \text{and} \quad \overline{\nu}_i = 0, && \text{if} \quad \frac{s_i}{\underline{\rho}_i} \le \frac{s_i}{\overline{\rho}_i} \le 0.
\end{aligned}$$

With this, the above set of constraints can also be written as

$$\frac{1}{2}\boldsymbol{u}^T K_i \boldsymbol{u} - \alpha\, a_i \le \overline{\nu}_i - \underline{\nu}_i \quad i = 1, \dots, m.$$

Obviously, all $\underline{\nu}_i$ and $\overline{\nu}_i$ satisfy the non-negativity constraints. Lastly, we can reformulate the objective function to match (2.54), since

$$\sum_{i=1}^{m} s_i = \sum_{i:\, \frac{s_i}{\underline{\rho}_i} > \frac{s_i}{\overline{\rho}_i}} \overline{\rho}_i \frac{s_i}{\overline{\rho}_i} + \sum_{i:\, \frac{s_i}{\underline{\rho}_i} \le \frac{s_i}{\overline{\rho}_i}} \underline{\rho}_i \frac{s_i}{\underline{\rho}_i} = \sum_{i:\, \frac{s_i}{\underline{\rho}_i} > \frac{s_i}{\overline{\rho}_i}} \overline{\rho}_i \overline{\nu}_i - \sum_{i:\, \frac{s_i}{\underline{\rho}_i} \le \frac{s_i}{\overline{\rho}_i}} \underline{\rho}_i \underline{\nu}_i = \sum_{i=1}^{m} \overline{\rho}_i \overline{\nu}_i - \sum_{i=1}^{m} \underline{\rho}_i \underline{\nu}_i .$$

We switch the sign of the objective function and claims (i) and (ii) follow. □

Assume that $(\boldsymbol{u}, \alpha)$ is feasible for (2.58) and $(\boldsymbol{\rho}, \boldsymbol{u})$ is feasible for the primal problem (2.48). Combining Theorem 2.3.12 (i) and Theorem 2.3.8 (i), we get the identity $\min\,(2.48) = \max\,(2.58)$ and thus the following formula for the duality gap:

$$\begin{aligned}
\delta(\boldsymbol{u}, \alpha) &:= \min\,(2.48) - \max\,(2.58) \\
&= -\frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} + \alpha V \\
&\quad + \sum_{i=1}^{m} \max\left\{ \underline{\rho}_i \left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}} K_i \boldsymbol{u} - \alpha a_i\right), \overline{\rho}_i \left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}} K_i \boldsymbol{u} - \alpha a_i\right)\right\}.
\end{aligned} \tag{2.59}$$

The dual VTS problem for a zero lower density bound

Strictly speaking, the VTS problem with non-zero lower density bounds belongs to the category of sizing, rather than topology optimization, since the elements are never completely removed in areas where the optimal design might be completely void of material [98, 5]. That it is a valid approximation of the proper topology optimization problem where $\underline{\boldsymbol{\rho}} = \mathbf{0}$ was shown by Achtziger [5]. Still, we would ideally like to solve the problem for $\underline{\boldsymbol{\rho}} = \mathbf{0}$, since this would yield a more accurate result. Numerically, the main issue with this is that the density-stiffness map (2.46) for the VTS only gives $\mathrm{K}(\boldsymbol{\rho}) \quad 0$ for $\boldsymbol{\rho} > 0$. Typically, one therefore either chooses a non-zero lower bound or a minimal stiffness coefficient for void elements in order to avoid ill-conditioning [22]. However, none of the tested algorithms struggled with this issue even for zero lower bounds, see Chapter 3. For this reason, the rest of the section contains modifications – indeed, simplifications – of the duality results presented previously, now for the case $\underline{\boldsymbol{\rho}} = \mathbf{0}$. The proofs require only minor adjustments which are fairly straightforward. Therefore, brief notes are given rather than comprehensive proofs for most modified results.

For a zero lower bound on the densities, $\underline{\boldsymbol{\rho}} = \mathbf{0}$, the dual of the VTS problem has the following form:

$$\min_{\boldsymbol{u}\in\mathrm{R}^n,\, \alpha\in\mathrm{R},\, \boldsymbol{v}\in\mathrm{R}^m} \quad \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} + \overline{\boldsymbol{\rho}}^\mathsf{T}\boldsymbol{v} \tag{2.60a}$$

$$\text{s.t.} \quad \frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} \leq \alpha\, a_i + v_i \qquad i = 1, \ldots, m, \tag{2.60b}$$

$$v_i \geq 0, \qquad i = 1, \ldots, m. \tag{2.60c}$$

Theorem 2.3.13. *Problems (2.48) with $\underline{\boldsymbol{\rho}} = \mathbf{0}$ and (2.60) are equivalent in the following sense:*

(i) If one problem has a solution, so does the other and

$$\min(2.48) = -\min(2.60)\ .$$

(ii) Let $(\boldsymbol{u}^*, \alpha^*, \boldsymbol{v}^*)$ be a solution of (2.60). Further, let $\boldsymbol{\tau}^*$ be the vector of associated Lagrange multipliers for the inequality constraints (2.60b). Then $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$

is a solution of (2.48). Moreover, α^*, $\boldsymbol{v}^*$ are the Lagrange multipliers associated with this solution for the volume and upper bound constraint, respectively.

(iii) Let $(\boldsymbol{\rho}^*, \boldsymbol{u}^*)$ be a solution of (2.48). Further, let $\boldsymbol{r}^*$ be the Lagrange multipliers for the upper bound constraint and let λ^* be the multiplier for the volume constraint. Then $(\boldsymbol{u}^*, \lambda^*, \boldsymbol{r}^*)$ is a solution of (2.60). Moreover, $\boldsymbol{\rho}^*$ are the Langrange multipliers associated with this solution for the inequality constraint (2.60b).

Proof. We can amend the proof of Theorem 2.3.8 by considering only a partial Lagrangian of the primal problem, so that no multiplier for the lower bound appears in (2.56). The non-negativity constraint in the minimization term is now equivalent to $\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho}$. In the rest of the proof, we simply drop all terms involving the lower bound or associated multipliers. □

Theorem 2.3.14. *Problem (2.60) is equivalent to the unconstrained nonsmooth problem*

$$-\max_{\boldsymbol{u}\in\mathbb{R}^n,\alpha\in\mathbb{R}} -\alpha\bar{V} + \boldsymbol{f}^\top\boldsymbol{u} - \sum_{i=1}^{m} \max\left(0, \frac{1}{2}\boldsymbol{u}^\top \mathrm{K}_i\boldsymbol{u} - \alpha a_i\right)\bar{\rho}_i \tag{2.61}$$

in the following sense:

(i) $\min(2.54) = -\max(2.61)$,

(ii) Let $(\boldsymbol{u}^*, \alpha^*, \boldsymbol{v}^*)$ be a solution of (2.60). Then $(\boldsymbol{u}^*, \alpha^*)$ is a solution of (2.61). Conversely, every solution $(\boldsymbol{u}^*, \alpha^*)$ of (2.61) is also (part of) a solution of (2.60).

Proof. Completely analogous to the proof of Theorem 2.3.12, we can rewrite (2.61) as

$$\begin{aligned} \max_{\boldsymbol{u}\in\mathbb{R}^n, \alpha\in\mathbb{R}, \boldsymbol{s}\in\mathbb{R}^m} \quad & -\alpha V + \boldsymbol{f}^\top \boldsymbol{u} - \sum_{i=1}^{m} s_i \\ \text{s.t.} \quad & \left(\frac{1}{2}\boldsymbol{u}^\top \mathrm{K}_i \boldsymbol{u} - \alpha a_i\right) \overline{\rho}_i \le s_i, \quad i = 1, \dots, m, \\ & 0 \le s_i, \quad i = 1, \dots, m. \end{aligned}$$

Now, we simply define $v_i = s_i/\overline{\rho}_i$ and switch the sign of the objective function to show the equivalence to problem (2.60). □

Assume that $(\boldsymbol{u}, \alpha)$ is feasible for (2.61) and $(\boldsymbol{\rho}, \boldsymbol{u})$ is feasible for the primal problem (2.48) with $\underline{\boldsymbol{\rho}} = \mathbf{0}$. We get the following formula for the duality gap:

$$\begin{aligned} \delta(\boldsymbol{u}, \alpha) &:= \min\,(2.48) - \max\,(2.61) \\ &= -\frac{1}{2}\boldsymbol{f}^\top \boldsymbol{u} + \alpha V \\ &\quad + \sum_{i=1}^{m} \max\left\{0, \overline{\rho}_i \left(\frac{1}{2}\boldsymbol{u}^\top \mathrm{K}_i \boldsymbol{u} - \alpha a_i\right)\right\}. \end{aligned} \tag{2.62}$$

It should be noted that all results regarding uniqueness of a solution rely on the positive definiteness of the stiffness matrix and thus on $\underline{\boldsymbol{\rho}} > \mathbf{0}$. This is true for the discrete case discussed here as well as for the continuous case [97]. Not even the LICQ can be shown to hold, at least as it was done in Lemma 2.3.5, if $\underline{\boldsymbol{\rho}} = \mathbf{0}$. One can construct a simple example of a non-unique solution for this case by noticing that displacements in areas of zero density can be arbitrary, within certain regularity bounds [97].

We still choose to solve the zero lower bound version of the VTS with the algorithms presented in Chapter 3. This is justified, firstly, by successful convergence in practice. Secondly, Achtziger showed in [5] that a sequence of finite-dimensional VTS problems for monotonically decreasing lower bounds $\underline{\boldsymbol{\rho}}^{(j)} > 0$ with $\lim_{j\to\infty} \underline{\boldsymbol{\rho}}^{(j)} = \mathbf{0}$ converges in terms of the optimal objective function value. While the sequence of optimal densities does not necessarily converge, it seemed to do so in most our numerical examples. Where it did not, this manifested in small areas of quickly oscillating density values, reminiscent of the well-known “checkerboarding” microstructure seen for material penalization [22] or the fibre-like structure in free-material optimization [138]. The analysis in [98] suggests that this phenomenon is due to the wrong choice of FE basis functions for the displacements and could be avoided by using piecewise quadratic instead of bilinear basis functions. However, the observed artefacts were usually small in range and amplitude and did not impede the physical interpretation of the design, so that we deem bilinear basis functions completely satisfactory for all practical purposes – at least in all examples we have considered.

2.3.2 The Variable Thickness Sheet Problem with Unilateral Contact

Considering that unilateral contact is a relatively simple but important step towards more realistic problem modelling, it has received comparatively little attention in the topology optimization literature, especially in recent years. Seminal research on unilateral contact in truss and sheet design was done in the 1990s, see for example [75, 99, 79]. Klarbring, Petersson, and Rönnqvist [75] proposed an LP formulation for truss optimization under unilateral contact, including a proof of existence of a solution. In [97], Petersson provided existence results for the exact VTS problem, and in [99], Petersson and Patriksson proposed a subgradient method along with convergence results. A method based on a dual reformulation of the problem was described in [79], which is the basis for the algorithm derived later in this section. For a review of important contributions at the time, see the article by Hilding, Klarbring, and Petersson [65].

More work has been done on more general problems. We give a short, non-exhaustive overview here as a starting point for further reading. Mankame and Ananthasuresh [86] looked at compliant mechanisms in truss design with unilateral contact. Several papers deal with the SIMP formulation of topology optimization involving contact. A solution method for the SIMP minimium compliance problem with unilateral contact, based on a nested approach combining sequential linear programming and an interior point method for the inner linear program, can be found in [120]. Fancello [48] used numerical solvers for contact problems in a nested approach for mass minimization under contact and local failure constraints. Recently, topology optimization problems involving large deformations and bilateral contact have also been solved [49, 82].

We now extend the VTS formulation of the minimum compliance problem for the case that rigid body obstacles are present. We include these in the form of contact constraints as described in Section 2.2.1. The derivation of both the primal and dual optimization problem very closely follow the previous section. We start with the saddle point formulation of the minimum compliance problem, see Lemma 2.3.3, only now, we include the contact constraints

$$\mathrm{C}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{b} \leq \mathbf{0}$$

and start from the max-min, rather than the min-max formulation:

$$\max_{\substack{\underline{\boldsymbol{\rho}} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}} \\ \sum_i \rho_i a_i = V}} \; \min_{\mathrm{C}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{b} \leq \mathbf{0}} \; \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u}\,. \tag{2.63}$$

To derive a primal optimization problem analogous to (2.48), we note that the inner minimization is now no longer unconstrained. Where before, we could resolve it simply through the stationarity condition, we now have to introduce the Lagrange multipliers $\boldsymbol{\eta}$ and utilize the KKT conditions (2.44). These are necessary and sufficient for the inner minimization problem, since the potential energy is quadratic in $\boldsymbol{u}$ for a fixed $\boldsymbol{\rho}$ and thus convex. Using (2.44a) and (2.44d), we get

$$\min_{\mathrm{C}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{b} \leq \mathbf{0}} \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} = \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\left(\boldsymbol{f} - \mathrm{C}\boldsymbol{\eta}\right) - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} = -\frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} - \frac{1}{2}\boldsymbol{b}^{\mathsf{T}}\boldsymbol{\eta}\,.$$

If we now switch the sign of the objective function in (2.63), we see that it is equivalent to [79]:

$$\min_{\boldsymbol{\rho}\in\mathbb{R}^m,\boldsymbol{u}\in\mathbb{R}^n,\boldsymbol{\eta}\in\mathbb{R}^l} \quad \frac{1}{2}\boldsymbol{f}^\mathsf{T}\boldsymbol{u} + \frac{1}{2}\boldsymbol{b}^\mathsf{T}\boldsymbol{\eta} \tag{2.64a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \rho_i \mathrm{K}_i \, \boldsymbol{u} = \boldsymbol{f} - \mathrm{C}\boldsymbol{\eta}\,, \tag{2.64b}$$

$$\mathrm{C}^\mathsf{T}\boldsymbol{u} \le \boldsymbol{b}\,, \tag{2.64c}$$

$$\boldsymbol{\eta} \ge 0\,, \tag{2.64d}$$

$$\boldsymbol{\eta}^\mathsf{T}(\mathrm{C}^\mathsf{T}\boldsymbol{u} - \boldsymbol{b}) = 0\,, \tag{2.64e}$$

$$\sum_{i=1}^{m} \rho_i a_i = V\,, \tag{2.64f}$$

$$0 < \underline{\rho}_i \le \rho_i \le \overline{\rho}_i, \quad i = 1,\dots,m\,. \tag{2.64g}$$

The equivalence between minimization of compliance (or maximization of stiffness) and the maximization of the equilibrium potential energy in (2.63), from which our problem has been derived, is not necessarily intuitive. This is especially true once contact is included, as pointed out in [97]. Nevertheless, we attempt to give a plausible physical interpretation of (2.64). Added to the compliance in the objective function are the nodal contact stresses weighted by the initial gaps $\boldsymbol{b}$. If we view $\boldsymbol{b}$ as prescribed displacements and the contact stress values $\boldsymbol{\eta}$ as the resulting forces on the contact surface, then $\frac{1}{2}\boldsymbol{b}^\mathsf{T}\boldsymbol{\eta}$ is the negative work done by the contact stresses. The objective function of problem (2.64) means we minimize the external work for prescribed loads while maximizing the external work for prescribed displacements [97, 20]. Alternatively, one could say that at all points at which the loaded design deforms enough to close a non-zero gap, we try to keep the reaction (pressure) forces as low as possible, so as not to lean on the contact surface too much. Where initial gaps are zero (or comparatively small), large contact forces are not penalized in the objective (as much). This means the optimal design might in fact rely on those parts of the contact surface for load-bearing.

We turn our attention once again to the saddle point formulation (2.63), in order to derive a dual problem analogous to (2.54). Since the inner minimization problem is still concave, we can proceed just like in the proof of Theorem 2.3.8 and obtain the following problem [79]:

$$\begin{aligned}
\min_{\boldsymbol{u}\in\mathrm{R}^n,\, a\in\mathrm{R},\, \underline{\boldsymbol{v}},\overline{\boldsymbol{v}}\in\mathrm{R}^m} \quad & \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} - \underline{\boldsymbol{\rho}}^\mathsf{T}\underline{\boldsymbol{v}} + \overline{\boldsymbol{\rho}}^\mathsf{T}\overline{\boldsymbol{v}} \\
s.t. \quad & \frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} \le \alpha a_i - \underline{v}_i + \overline{v}_i, \quad i = 1,\dots,m, \\
& \mathrm{C}^\mathsf{T}\boldsymbol{u} \le \boldsymbol{b}, \\
& \underline{v}_i \ge 0, \quad i = 1,\dots,m, \\
& \overline{v}_i \ge 0, \quad i = 1,\dots,m.
\end{aligned}$$

In contrast to the primal problem (2.64), incorporating unilateral contact in the dual only adds one extra set of inequality constraints and no extra variables.

We focus on the case of zero lower density bounds, as the method we will use in Chapter 3 to solve the problem performed well for $\underline{\boldsymbol{\rho}} = \mathbf{0}$ despite open questions of uniqueness or theoretical convergence. The dual VTS problem with unilateral contact for zero lower density bounds is

$$\min_{\boldsymbol{u}\in\mathrm{R}^n,\, a\in\mathrm{R},\, \boldsymbol{v}\in\mathrm{R}^m} \quad \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} + \overline{\boldsymbol{\rho}}^\mathsf{T}\boldsymbol{v} \tag{2.65a}$$

$$\text{s.t.} \quad \frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} \le \alpha a_i + v_i \quad i = 1,\dots,m, \tag{2.65b}$$

$$\mathrm{C}^\mathsf{T}\boldsymbol{u} \le \boldsymbol{b}, \tag{2.65c}$$

$$v_i \ge 0, \quad i = 1,\dots,m. \tag{2.65d}$$

Naturally, we can obtain a result similar to Theorem 2.3.13.

Theorem 2.3.15. *Problems (2.64) with* $\underline{\boldsymbol{\rho}} = \mathbf{0}$ *and (2.65) are equivalent in the following sense:*

(i) If one problem has a solution, so does the other and

$$\min(2.64) = -\min(2.65)\ .$$

(ii) Let $(\boldsymbol{u}^*, \alpha^*, \boldsymbol{v}^*)$ be a solution of (2.65). Further, let $\boldsymbol{\tau}^*$, $\boldsymbol{\eta}^*$ be the associated vectors of Lagrange multipliers for the inequality constraints (2.65b) and (2.65c). Then $(\boldsymbol{\tau}^*, \boldsymbol{u}^*, \boldsymbol{\eta}^*)$ is a solution of (2.64). Moreover, α^*, $\boldsymbol{v}^*$ are the Lagrange multipliers associated with this solution for the volume and bound constraint, respectively.

(iii) Let $(\boldsymbol{\rho}^*, \boldsymbol{u}^*, \boldsymbol{\eta}^*)$ be a solution of (2.64). Further, let $\boldsymbol{r}^*$ be the Lagrangian multipliers associated with the upper bounds on $\boldsymbol{\rho}$, and let λ^* be the multiplier for the volume constraint. Then $(\boldsymbol{u}^*, \lambda^*, \boldsymbol{r}^*)$ is a solution of (2.65). Moreover, $\boldsymbol{\rho}^*$, $\boldsymbol{\eta}^*$ are the Lagrange multipliers associated with this solution for the inequality constraints (2.65b) and (2.65c), respectively.

Proof. The proof is a straightforward adjustment of the proofs of Theorems 2.3.8 and 2.3.13, taking into consideration the added contact constraints and corresponding Lagrange multipliers $\boldsymbol{\eta}$. □

For the case without contact constraints, we introduced another reformulation of the problem as the basis for a duality gap expression which did not require any Lagrange multipliers apart from α. This served the purpose of being able to compare the results of primal and dual algorithms. However, since contact constraints cannot be easily incorporated into the primal method used for (2.48), no such comparison will be required. We can instead simply use the following expression for the duality gap, based on Theorem 2.3.15:

$$\begin{aligned}\delta(\boldsymbol{u},\boldsymbol{\eta},\alpha,\boldsymbol{\nu},\overline{\boldsymbol{\nu}}) &:= \min(2.64) - (-\min(2.65))\\ &= \frac{1}{2}\boldsymbol{f}^\top\boldsymbol{u} + \frac{1}{2}\boldsymbol{b}^\top\boldsymbol{\eta} + \alpha V - \boldsymbol{f}^\top\boldsymbol{u} + \overline{\boldsymbol{\rho}}^\top\boldsymbol{\nu}\\ &= -\frac{1}{2}\boldsymbol{f}^\top\boldsymbol{u} + \frac{1}{2}\boldsymbol{b}^\top\boldsymbol{\eta} + \alpha V + \overline{\boldsymbol{\rho}}^\top\boldsymbol{\nu}\,.\end{aligned}$$

This concludes our treatment of the variable thickness sheet problem. We end the section with a description of an optimization method designed specifically for the VTS problem.

2.3.3 The Optimality Criteria Method

The *optimality criteria* (OC) method [22, 111, 136] is a popular algorithm for solving the minimum compliance problem. It is easy to implement and can handle the VTS as well as the SIMP formulation, exemplified by the much-cited Matlab implementation [12] and more recently [50]. In a benchmarking paper by Rojas-Labanda and Stolpe comparing different optimization algorithms, the OC method outperformed other methods in terms of computational time, including the method of moving asymptotes [121] which is perhaps the most widely used method in topology optimization. We will compare the performance of our IP and PBM implementations against that of the OC method in Section 3.2 and therefore outline this method here. One thing to note straight away is that it does not allow for contact constraints. On the other hand, while we derive it here only for the VTS problem, it is straightforward to extend to the SIMP formulation, for which it is most often used.

In [16], Ben-Tal and Bendsøe derived a particular set of necessary and sufficient optimality conditions for the minimum compliance problem, which offer a very instructive characterization of the optimal solution. In its original form, they are stated as conditions for (2.58). We include it in a slightly rephrased form further below, connecting it more closely to the formulations (2.48) and (2.54). To motivate the next theorem, let us regard the first set of inequality constraints of the dual VTS problem (2.54):

$$\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i + \underline{\nu}_i - \overline{\nu}_i \leq 0, \quad i = 1, \dots, m.$$

From Theorem 2.3.12, we know that, at the solution, the densities $\boldsymbol{\rho}$ double as Lagrange multipliers for the above constraints. Furthermore, $\underline{\boldsymbol{\nu}}$ and $\overline{\boldsymbol{\nu}}$ are the Lagrange multipliers of the primal lower and upper bound constraints, respectively. For any *intermediate* element density, that is any ρ_i with $\underline{\rho}_i < \rho_i < \overline{\rho}_i$, complementarity therefore demands

$$\rho_i\left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i\right) = 0. \tag{2.66}$$

As $\rho_i \neq 0$, this further implies $\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i = 0$. The following theorem shows that the converse is also true (although in a less strict sense).

Theorem 2.3.16. *A pair $\boldsymbol{\rho}^*$, $\boldsymbol{u}^*$ is an optimal solution of (2.48) and α^* is the associated multiplier for the volume constraint if and only if*

$$\rho_i^* = \underline{\rho}_i \quad \text{if} \quad \frac{1}{2}(\boldsymbol{u}^*)^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u}^* < \alpha^* a_i, \tag{2.67a}$$

$$\rho_i^* = \overline{\rho}_i \quad \text{if} \quad \frac{1}{2}(\boldsymbol{u}^*)^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u}^* > \alpha^* a_i, \tag{2.67b}$$

$$\underline{\rho}_i \leq \rho_i^* \leq \overline{\rho}_i \quad \text{if} \quad \frac{1}{2}(\boldsymbol{u}^*)^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u}^* = \alpha^* a_i, \tag{2.67c}$$

$$\sum_{i=1}^{m}\rho_i\mathrm{K}_i\boldsymbol{u} = \boldsymbol{f}, \tag{2.67d}$$

$$\sum_{i=1}^{m}\rho_i a_i = V. \tag{2.67e}$$

Proof. See [16]. □

The last two conditions are simply the equilibrium equations and volume constraint. Equations (2.67a–2.67c) allow an interesting interpretation of α^*. Since $\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}(\rho_i\mathrm{K}_i)\boldsymbol{u}$ is the internal strain energy of element i, we can view $\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u}/a_i$ as the strain energy normalized with respect to the element's mass. The multiplier $\alpha*$ gives a particular threshold value for this term: Each element whose normalized strain energy exceeds $\alpha*$ requires the maximum amount of material, according to (2.67b); on the other hand, if this energy is below $\alpha*$, the minimum amount of material is sufficient, see (2.67a). For any element with intermediate density, the normalized strain energy is exactly $\alpha*$.

The OC method is an iterative updating scheme that focuses on the densities $\rho_1, \dots, \rho_m$, which it essentially splits into three groups according to (2.67a–2.67c). Those that satisfy (2.67c) are fixed points for (2.66), which motivates the update formula

$$\rho_i^{(k+1)} := \rho_i^{(k)} \frac{\frac{1}{2}(\boldsymbol{u}^{(k)})^{\mathsf{T}} \mathrm{K}_i \boldsymbol{u}^{(k)}}{\alpha a_i} \quad , \tag{2.68}$$

where α is determined through bisection such that $\boldsymbol{\rho}^{(k+1)}$ satisfies the volume constraint. To avoid overshooting and improve convergence, we impose constant move limits. Additionally, we apply a "damping" to the term in brackets on the right-hand side by raising it to a power $0 < q < 1$. Furthermore, if the update would push ρ_i outside of the feasible bounds, we assume that ρ_i belongs to one of the groups defined by (2.67a) and (2.67b) and assign it the value $\underline{\rho_i}$ or $\overline{\rho_i}$ accordingly. Algorithm 1 summarizes the method. As a measure of optimality, we use the duality gap defined in (2.59). Unless otherwise specified, we follow [12] for the choice of the move limit, bisection bounds and damping parameter: $\overline{\alpha}_0 = 10^9$, $\underline{\alpha}_0 = 0$, $\Delta\rho = 0.2$ and $q = \frac{1}{2}$.

Note that the displacements and densities are not updated simultaneously. The OC method technically does not solve (2.48), but the so-called *nested* formulation of the minimum compliance problem, where $\boldsymbol{u}$ is featured not as a variable, but as a function of $\boldsymbol{\rho}$. The equilibrium equations are not included explicitly in the nested formulation, but implicitly through $\boldsymbol{u}(\rho)$.

Algorithm 1 Optimality Criteria Method

Let $\boldsymbol{\rho} \in \mathbb{R}^m$ be given such that $\sum_{i=1}^m \rho_i a_i = V$, $\underline{\rho}_i \leq \rho_i \leq \overline{\rho}_i$, $i = 1, \dots, m$. Further, let $\varepsilon_{\text{OC}} > 0$, $\overline{\alpha}_0 > \underline{\alpha}_0 > 0$, $0 < \Delta\rho < 1$ and $q < 1$ be given.

```
 1: repeat
 2:     Solve $K(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$
 3:     $\underline{\rho}^i = \max\{\underline{\rho}_i, \rho^i - \Delta\rho\}$, $\overline{\rho}^i = \min\{\overline{\rho}_i, \rho^i + \Delta\rho\}$ for $i = 1, \dots, m$
 4:     $\overline{\alpha} = \overline{\alpha}_0$, $\underline{\alpha} = \underline{\alpha}_0$
 5:     while $\frac{\overline{\alpha} - \underline{\alpha}}{\overline{\alpha} + \underline{\alpha}} > \varepsilon_{\text{OC}}$ do
 6:         $\alpha = (\underline{\alpha} + \overline{\alpha})/2$
 7:         for $i = 1, \dots, m$ do
 8:             $\rho_i^+ = \min\left\{\max\left\{\rho_i\left(\frac{\frac{1}{2}\boldsymbol{u}_i^T K_i \boldsymbol{u}_i}{\alpha a_i}\right)^q, \underline{\rho}_i\right\}, \overline{\rho}_i\right\}$
 9:         end for
10:         if $\sum_{i=1}^m \rho_i^+ a_i > V$ then
11:             $\underline{\alpha} = \alpha$
12:         else
13:             $\overline{\alpha} = \alpha$
14:         end if
15:     end while
16:     $\boldsymbol{\rho} = \boldsymbol{\rho}^+$
17: until $\delta(\boldsymbol{u}, \alpha) < \varepsilon_{\text{OC}}$
```

2.4 Iterative Methods for Linear Systems

Each of the optimization algorithms discussed in the previous sections entails several linear systems that need to be solved numerically throughout the course of the optimization. In the IP and PBM method, we need to obtain one or more Newton increments at each iteration, and the OC method recomputes the displacements from the equilibrium equations after each update of the densities. Solving the corresponding linear systems accounts for a large part of the overall computational cost of the optimization. Choosing the right solver is therefore a crucial part of any efficient strategy for solving large-scale problems, where even small improvements can save hours, if not days, of computation time. This section is concerned with numerical methods for finding a solution to the type of linear system that we will encounter in the algorithms proposed in Chapter 3 and in the OC method. It is of the general form

$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.69}$$

where we are given a *right-hand side* vector $\boldsymbol{b} \in \mathbb{R}^n$ and a system matrix $A \in \mathbb{R}^{n\times n}$ which is symmetric and positive definite. It is also sparse, i.e. the number of non-zero entries in A is a small fraction of n^2. In particular, we assume that this number is in the order of n. The goal is to solve large-scale systems where $n > 10^6$.

In general, methods for solving linear systems are grouped into direct solvers and iterative solvers. The former yield an exact[1] solution of (2.69) after a fixed number of steps, while the latter produce a sequence $\boldsymbol{x}_k$ which converges to the solution as $k \to \infty$. To compare the two in terms of computational efficiency, let us briefly recall the *Landau-notation* [131]:

Definition 2.4.1. A function $c : \mathbb{N} \longrightarrow (0, \infty)$ is said to be (in) $O(d(n))$ for another function $d : \mathbb{N} \longrightarrow (0, \infty)$, if there exists a constant $\beta > 0$ such that

$$c(n) \leq \beta\, d(n) \quad \forall\, n \in \mathbb{N}.$$

We write $c(n) = O(d(n))$. Further, we say that a computational procedure has $O(d(n))$ complexity if it requires $O(d(n))$ floating point operations.

In general, direct methods have $O(n^3)$ complexity [131, 104]. This can be reduced if one takes advantage of the sparsity of the system matrix and any special structure it might exhibit. For a banded matrix with upper and lower bandwidth $p \ll n$, for example, direct solvers can be adjusted to run in $O(np^2)$ [104]. However, they still suffer from so-called fill-in. Direct methods typically involve some factorization of the matrix A, for example $A = LL^T$ for the Cholesky-factorization [131]. The matrix L does not generally have the same sparsity structure, in fact, it is usually less sparse than A. As a result, memory becomes an issue for large n.

The main computational cost in iterative solvers comes from matrix-vector multiplications performed in each iteration. Since this operation is in $O(n^2)$, the overall complexity of such a solver is $O(kn^2)$, where k is the number of iterations needed to

[1]assuming exact arithmetic

obtain an acceptable solution. As before, this can be improved by taking into account the sparsity of A. If one does not require an exact solution, iterative methods that can reach a satisfactory approximation in only a few, that is $N \ll n$, iterations have a clear advantage over direct methods. This makes them a better choice for our purposes, since in our optimization algorithms, we only need to solve the Newton systems with relatively low accuracy[1]. For a thorough introduction to and a good general overview of direct and iterative solvers, we refer to the books [131] by Wendland and [104] by Quarteroni, Sacco, and Saleri.

Two classes of iterative solvers are of particular interest for the kind of system we will need to solve: Krylov subspace methods, more specifically the conjugate gradient and minimal residual method, and multigrid methods. The former are guaranteed to return an exact solution after at most n steps and could thus be classified as direct methods; however, their strength lies in obtaining a good approximation in relatively few steps if the matrix A is well-conditioned. The basic concepts of Krylov methods and the general convergence behaviour of the conjugate gradient and minimal residual method are the subject of Section 2.4.1. We also touch upon preconditioning, motivating the use of multigrid methods as spectrally equivalent preconditioners. Section 2.4.2 then gives a brief introduction to multigrid methods, which are very effective for systems arising from the discretization of elliptic partial differential equations, such as the equilibrium equations (2.38). They make use of specific properties of the resulting system matrix and constitute an optimal iterative method in that the number of iterations required until a certain error measure is below a fixed tolerance value does not depend on the size of the system. When the mesh is no longer regular or structured, certain components of the standard multigrid method need to be generalized. This leads to algebraic multigrid methods, which are also covered in Section 2.4.2.

[1]compared with machine precision

2.4.1 Krylov Subspace Methods

To start off, we introduce some basic notation. Let $\boldsymbol{x}_0$ be the initial guess for a solution to system (2.69) and let us denote by $\boldsymbol{x}_k$, $k = 1, 2, \ldots$, the sequence of approximations returned by an iterative solver. Further, the kth *residual* is defined as $\boldsymbol{r}_k := \boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k$. We now turn to a class of iterative solvers which construct iterates that satisfy

$$\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathrm{K}_k(\mathrm{A}, \boldsymbol{r}_0)\,,$$

where $\mathrm{K}_k(\mathrm{A}, \boldsymbol{r}_0)$ is the *Krylov space* defined as

$$\mathrm{K}_k(\mathrm{A}, \boldsymbol{r}_0) := \operatorname{span}\left\{\boldsymbol{r}_0, \mathrm{A}\boldsymbol{r}_0, \mathrm{A}^2\boldsymbol{r}_0, \ldots, \mathrm{A}^{k-1}\boldsymbol{r}_0\right\}.$$

Accordingly, such solvers are called *Krylov (subspace) methods*. Two methods of this class find application in our algorithms, the *conjugate gradient* (CG) method and the *minimum residual* (MINRES) method. Because we use the generic versions of these solvers, what follows is a very condensed discussion, covering the main convergence results and highlighting the differences between the two methods. Many books have been written that can be consulted for more details. We mention here the book by Saad [114], an extremely comprehensive treatment on iterative solvers, and that by Greenbaum [58], as a reference for the MINRES method in particular. The corresponding chapters in [131] also provide a good introduction to Krylov solvers. Detailed pseudo-code for all methods discussed in this section can be found in [58, 131]. Last but not least, no literature list for the CG method is complete without the excellent introductory article by Shewchuk [118].

Both the CG and MINRES method can be viewed as optimization algorithms for a function, say $r(\boldsymbol{x})$, that is in some way a measure of how much $\boldsymbol{x}$ deviates from the true solution of (2.69). In each iteration, we update the approximate solution by

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \sigma_k \boldsymbol{d}_k\,,$$

where the step size σ_k is defined such that it minimizes $r(\boldsymbol{x}_k + \sigma \boldsymbol{d}_k)$ over $\sigma \in \mathbb{R}$. The search directions $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{k-1}$ are chosen so that they form a basis of $\mathcal{K}_k(A, \boldsymbol{r}_0)$. The methods differ in their choice of $r(\boldsymbol{x})$ and in how they construct the basis vectors.

For later reference, we denote the error of some $\boldsymbol{x}$ with respect to (2.69), that is the difference between $\boldsymbol{x}$ and the exact solution $\boldsymbol{x}^*$ of (2.69), by $\boldsymbol{e} := \boldsymbol{x}^* - \boldsymbol{x}$. We further observe the following identity involving the kth residual and the error $\boldsymbol{e}_k := \boldsymbol{x}^* - \boldsymbol{x}_k$ of the kth iterate:

$$A\boldsymbol{e}_k = \boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_k . \tag{2.70}$$

Conjugate Gradient Method

The CG method, originally proposed by Hestenes and Stiefel [64], is no doubt one of the most popular iterative methods for large, sparse, symmetric positive definite matrices. It uses a set of vectors $\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{k-1}\}$ as a basis of $\mathcal{K}_k(A, \boldsymbol{r}_0)$ which are A-conjugate, that is

$$\boldsymbol{d}_i^T A \boldsymbol{d}_j = 0 \quad \text{for } i \neq j .$$

Because the matrix A is positive definite, we can define an inner product $(\boldsymbol{v}, \boldsymbol{w})_A := \boldsymbol{v}^T A \boldsymbol{w}$ and a corresponding norm

$$\|\boldsymbol{v}\|_A = (\boldsymbol{v}, \boldsymbol{v})_A^{1/2}, \tag{2.71}$$

which we call *energy norm*. We can motivate the CG method as a minimization algorithm for the functional $r(\boldsymbol{x}) := \frac{1}{2} \boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}$. Since this $r(\boldsymbol{x})$ is strictly convex, (2.69) represents the necessary and sufficient optimality condition and its solution $\boldsymbol{x}^*$ is also the unique minimum of $r(\boldsymbol{x})$. The CG method produces iterates $\boldsymbol{x}_k$, $k = 1, 2, \ldots$, each of which minimizes $r(\boldsymbol{x})$ over $\boldsymbol{x}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$. Note that adding a constant term to $r(\boldsymbol{x})$ does not change its minimizer. Hence, we can see from

$$\begin{aligned}
r(\boldsymbol{x}) + (\boldsymbol{x}^*)^\mathsf{T} \mathrm{A}\boldsymbol{x}^* &= \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathrm{A}\boldsymbol{x} - \boldsymbol{b}^\mathsf{T}\boldsymbol{x} + \frac{1}{2}(\boldsymbol{x}^*)^\mathsf{T}\mathrm{A}\boldsymbol{x}^* \\
&= \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathrm{A}\boldsymbol{x} - (\mathrm{A}\boldsymbol{x}^*)^\mathsf{T}\boldsymbol{x} + \frac{1}{2}(\boldsymbol{x}^*)^\mathsf{T}\mathrm{A}\boldsymbol{x}^* \\
&= \frac{1}{2}(\boldsymbol{x}^* - \boldsymbol{x})^\mathsf{T}\mathrm{A}(\boldsymbol{x}^* - \boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{e}\|_\mathrm{A}^2
\end{aligned}$$

that minimizing $r(\boldsymbol{x})$ is equivalent to minimizing the energy norm of the error $\boldsymbol{e}$.

Due to the A-conjugacy of the search directions and the above optimality property, the kth residual vector is orthogonal to $\mathrm{span}\{\boldsymbol{d}_0, \dots, \boldsymbol{d}_{k-1}\}$. Therefore, $\boldsymbol{d}_k$ can be obtained by a Gram-Schmidt orthonormalization of $\boldsymbol{r}_k$ with respect to the inner product $(\ ,\ )_\mathrm{A}$. At first glance, this would seem to require all previous search directions in order to construct the next one. In this case, however, the procedure can be reduced to a three-term recurrence scheme. This leads to an algorithm in which, at each iteration, we can efficiently compute $\boldsymbol{d}_{k+1}$, $\boldsymbol{r}_{k+1}$ and $\boldsymbol{x}_{k+1}$ by a few vector inner products, vector additions and one matrix-vector multiplication. Furthermore, we need only store a single instance of each of these vector iterates.

Minimum Residual Method

The MINRES method was introduced by Paige and Saunders [95] as a solver for indefinite symmetric systems. It is nowadays a popular method for systems arising, for example, in incompressible fluid flow dynamics [45]. It can be seen as a generalization of the CG method: The way in which the search directions $\boldsymbol{d}_0, \boldsymbol{d}_1, \boldsymbol{d}_2, \dots,$ are computed in the CG method can be derived from the Lanczos algorithm for building an orthogonal basis of $\mathrm{K}_k(\mathrm{A}, \boldsymbol{r}_0)$, see for example [114]. Paige and Saunders extended this approach to indefinite matrices, for which one cannot generally define an inner product, so that the concepts of A-conjugacy and the energy norm are no longer applicable. The MINRES method iteratively constructs an orthogonal basis $\{\boldsymbol{d}_0, \dots, \boldsymbol{d}_{k-1}\}$ of $\mathcal{K}_k(A, \boldsymbol{r}_0)$ and updates $\boldsymbol{x}_k$ such that the residual norm $\|\boldsymbol{r}_k\|_2$ is minimized over $\boldsymbol{x}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$.

A single iteration of the MINRES method has roughly the same computational complexity as a single iteration of the CG method. In both cases, the most costly operation is a matrix-vector multiplication. Therefore, the overall costs of the two methods can be expected to scale the same with the size of the problem. In general, there are two main reasons for a difference in performance. First, the formulae and computational steps in a single iteration are more complicated for the MINRES than for the CG method. In particular, the updating scheme for the basis vectors is a three-term recurrence, so that more vectors need to be stored simultaneously and more vector additions need to be performed. Second, both methods usually formulate a stopping criterion in terms of the residual norm, since the error is not readily available. Since this is exactly what the MINRES method minimizes over the Krylov subspace, it generally requires fewer iterations than the CG method. One therefore has to consider this trade-off between cost per iteration and total number of iterations when choosing between the two solvers. While the CG method seems to be far more popular in practice, we compare both methods for our problem in Section 3.2, showing that the MINRES method performs better in some cases.

Convergence

The convergence behaviour of the CG and MINRES method is determined by the spectrum of the matrix A, as we will now show. Although the MINRES method can also solve indefinite problems, the optimization algorithms in which we will use it will only involve positive definite systems. We will therefore assume $A \succ 0$ for ease of presentation. We will denote iterates of the CG and MINRES method by a superscript *CG* and *MR*, respectively. Recall the optimality properties for the two methods:

$$\|\boldsymbol{e}_k^{CG}\|_A = \min_{\boldsymbol{x}_0+\mathcal{K}_k} \|\boldsymbol{e}\|_A\,, \qquad \|\boldsymbol{r}_k^{MR}\|_2 = \min_{\boldsymbol{x}_0+\mathcal{K}_k} \|\boldsymbol{r}\|_2\,. \tag{2.72}$$

Being elements of $\boldsymbol{x}_0 + \mathcal{K}_k(\mathrm{A}, \boldsymbol{r}_0)$, we can express $\boldsymbol{x}_k^{CG}$ and $\boldsymbol{x}_k^{MR}$ in terms of $\boldsymbol{r}_0$ and powers of A:

$$\boldsymbol{x}_k^{\Box} = \boldsymbol{x}_0 + \sum_{i=0}^{k-1} \gamma_i^{\Box} \mathrm{A}^i \boldsymbol{r}_0 ,$$

where we used $\Box$ as a placeholder for CG and MR. Using (2.70), we can then derive a similar expression for the error,

$$\begin{aligned} \boldsymbol{e}_k^{\Box} = \boldsymbol{x}^* - \boldsymbol{x}_k^{\Box} = \boldsymbol{e}_0 - \sum_{i=0}^{k-1} \gamma_i^{\Box} \mathrm{A}^i \boldsymbol{r}_0 = \boldsymbol{e}_0 - \sum_{i=0}^{k-1} \gamma_i^{\Box} \mathrm{A}^i \mathrm{A} \boldsymbol{e}_0 \\ = \left(\mathrm{I} - \sum_{i=1}^{k} \gamma_i^{\Box} \mathrm{A}^i \right) \boldsymbol{e}_0 , \end{aligned}$$

and for the residual,

$$\begin{aligned} \boldsymbol{r}_k^{\Box} = \boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k^{\Box} &= \boldsymbol{b} - \mathrm{A} \left(\boldsymbol{x}_0 + \sum_{i=0}^{k-1} \gamma_i^{\Box} \mathrm{A}^i \boldsymbol{r}_0 \right) \\ &= \boldsymbol{b} - \mathrm{A}\boldsymbol{x}_0 - \sum_{i=1}^{k} \gamma_i^{\Box} \mathrm{A}^i \boldsymbol{r}_0 \\ &= \left(\mathrm{I} - \sum_{i=1}^{k} \gamma_i^{\Box} \mathrm{A}^i \right) \boldsymbol{r}_0 . \end{aligned}$$

To make the notation even more concise, we can write these identities as

$$\boldsymbol{e}_k^{\Box} = p^{\Box}(A)\boldsymbol{e}_0 , \qquad \boldsymbol{r}_k^{\Box} = p^{\Box}(A)\boldsymbol{r}_0 ,$$

where $p^{\Box}(t)$ is a polynomial of degree k with $p(0) = 1$. Denoting the space of kth degree polynomials by P^k, we can now see that (2.72) is equivalent to

$$\begin{aligned} \|\boldsymbol{e}_k^{CG}\|_{\mathrm{A}} &= \min_{\substack{p \in \mathrm{P}^k \\ p(0)=1}} \|p(\mathrm{A})\boldsymbol{e}_0\|_{\mathrm{A}} , \\ \|\boldsymbol{r}_k^{MR}\|_2 &= \min_{\substack{p \in \mathrm{P}^k \\ p(0)=1}} \|p(\mathrm{A})\boldsymbol{r}_0\|_2 . \end{aligned}$$

Since any vector can be expressed as a linear combination of orthonormal eigenvectors of A, the above can be written in terms of the eigenvalues $\lambda_1, \ldots, \lambda_n$ of A.

(For details, see any of the references at the beginning of this section). This further allows us to derive the following upper bounds:

$$\|\boldsymbol{e}_k^{CG}\|_A \leq \min_{\substack{p \in \mathrm{P}^k \\ p(0)=1}} \max_{1 \leq i \leq n} |p(\lambda_i)| \|\boldsymbol{e}_0\|_A ,$$

$$\|\boldsymbol{r}_k^{MR}\|_2 \leq \min_{\substack{p \in \mathrm{P}^k \\ p(0)=1}} \max_{1 \leq i \leq n} |p(\lambda_i)| \|\boldsymbol{r}_0\|_2 .$$

This result provides several important insights. The upper bound on the energy norm of the error and on the residual norm, respectively, after *k* iterations of the CG and MINRES method depends on how close to zero the *k*th degree polynomial *p* can get at every eigenvalue. This guarantees that both methods obtain the exact solution after at most *n* iterations, since we can then find a polynomial with a root at each λ_i, or after $l < n$ iterations if A only has *l* distinct eigenvalues. What is more, even if the eigenvalues are merely clustered around *l* distinct values, an appropriate polynomial of degree *l* can yield very small values at all λ_i. Disregarding possible clustering of eigenvalues, one can use Chebyshev polynomials to obtain less tight but informative upper bounds

$$\|\boldsymbol{e}_k^{CG}\|_A \leq 2 \left(\frac{\sqrt{\mathrm{cond}(A)} - 1}{\sqrt{\mathrm{cond}(A)} + 1} \right)^k \|\boldsymbol{e}_0\|_A ,$$

$$\|\boldsymbol{r}_k^{MR}\|_2 \leq 2 \left(\frac{\sqrt{\mathrm{cond}(A)} - 1}{\sqrt{\mathrm{cond}(A)} + 1} \right)^k \|\boldsymbol{r}_0\|_2 ,$$

where $\mathrm{cond}(A)$ is the *condition number* of A, defined as

$$\mathrm{cond}(A) = \frac{\max_i \lambda_i}{\min_i \lambda_i} .$$

These upper bounds can be made smaller by narrowing the spectrum of A.

Preconditioning

The convergence results presented above make it clear that the performance of the CG and MINRES method depend on the distribution of the eigenvalues – the *conditioning* – of the system matrix A. In fact, these algorithms are not competitive in practice for systems that are not particularly well-conditioned. Therefore, CG and MINRES implementations typically employ some form of *preconditioning* technique.

Let $M \in \mathbb{R}^{n \times n}$ be a symmetric matrix, which we will call the *preconditioning matrix* or simply the *preconditioner*. We can find a factorization $M = LL^{\mathsf{T}}$, (for example a Cholesky factorization or the square root of M), such that L is invertible. The original linear system (2.69) is equivalent to the "preconditioned" system

$$\underbrace{L^{-1}AL^{-\mathsf{T}}}_{=:B}\ \underbrace{L^{\mathsf{T}}\boldsymbol{x}}_{=:\boldsymbol{y}} = \underbrace{L^{-1}\boldsymbol{b}}_{=:\boldsymbol{c}} . \tag{2.73}$$

The matrix B is symmetric positive definite, so that we can solve $B\boldsymbol{y} = \boldsymbol{c}$ by either the CG or MINRES method and obtain the solution of (2.69) by solving $L^{\mathsf{T}}\boldsymbol{x} = \boldsymbol{y}$. We then speak of the *preconditioned* CG or MINRES method. If $\mathrm{cond}(B) < \mathrm{cond}(A)$ or the eigenvalues of B are more tightly clustered than those of A, preconditioning significantly improves the convergence rate. Practical implementations of preconditioned Krylov solvers do not actually involve the system (2.73), which should be seen as a mere formalism. A reformulation of the standard CG or MINRES method applied to (2.73) yields an algorithm that essentially only differs from the original one by its use of a preconditioned residual $\boldsymbol{z_k}$, which is obtained by solving $M\boldsymbol{z_k} = \boldsymbol{r}_k$, which leads to an additional computational step in each iteration. We therefore do not need to know the factorization $M = LL^{\mathsf{T}}$, nor, in fact, do we require M to be explicitly given as a matrix. More generally, we only need a linear operator that performs the function of M^{-1}. Convergence analysis of a preconditioned Krylov solver is typically done in terms of the spectrum of $M^{-1}A$, since this matrix has the same eigenvalues as B, see for example [131, (Lemma 8.1)].

A good preconditioner should meet two criteria in order to provide an overall speed-up of the solver. First, we want $\operatorname{cond}(M^{-1}A)$ to be small, or alternatively, we want the eigenvalues of $M^{-1}A$ to be tightly clustered; second, we need to be able to solve the system $M\boldsymbol{z}_k = \boldsymbol{r}_k$ efficiently. These two requirements are somewhat at odds, as exemplified by the extreme examples of $M = I$ and $M = A$. While $M = I$ clearly satisfies the second requirement, it does not improve the conditioning at all, whereas $M = A$ would guarantee convergence in a single iteration, but if we knew how to solve $A\boldsymbol{z}_k = \boldsymbol{r}_k$ efficiently, we would not be resorting to an iterative solver to begin with. A good preconditioner must therefore strike a balance between the two requirements.

In the context of large-scale problems, there is another important criterion that a preconditioner should meet. Not only should the condition number of $M^{-1}A$ be small compared to $\operatorname{cond}(A)$, but it should ideally be independent of the problem size. This would guarantee that the number of iterations required to solve, for example, a discretized PDE, stays the same as we increase the discretization resolution. A preconditioner of this kind, for which $\operatorname{cond}(M^{-1}A) = O(1)$, is called *spectrally equivalent* to A [130]. The multigrid method discussed in the next section is an excellent candidate for such a preconditioner. It is actually an iterative solver in its own right, which, when used to solve symmetric positive definite systems arising from elliptic partial differential equations, gives a reduction of the error

$$\|\boldsymbol{e}_{k+1}\|_A \leq \eta \|\boldsymbol{e}_k\|_A$$

in each iteration. Here, the constant $\eta \in (0, 1)$ is independent of the problem size. This leads to $\operatorname{cond}(M^{-1}A) = (1+\eta)/(1-\eta)$ [130, 45], making the multigrid method an optimal preconditioner.

For an overview of different preconditioners for Krylov methods, also covering indefinite and asymmetric systems, see the review article by Wathen [130]. For a more in-depth treatment of preconditioning techniques in general, see for example

[115].

2.4.2 Multigrid Methods

Whereas the previous iterative methods are applicable to any symmetric (positive definite) linear system, *multigrid* methods were developed specifically for systems resulting from the discretization of PDEs. The system matrix A, more specifically its spectrum and eigenspaces, in those cases displays particular properties that the multigrid approach takes advantage of. It does this in a way that decouples its convergence behaviour from the resolution of the discretization and thus the size of the matrix A. Multigrid methods were originally proposed by Brandt [29]. We refer to the book by Briggs, Henson, and McCormick [33] for a very instructive introduction, or to the corresponding chapters in [131, 114] for a shorter but equally accessible treatment. A comprehensive and thorough discussion can be found, for example, in [61].

Multigrid methods have two main components: a hierarchy of coarse meshes of decreasing resolution, on which low-resolution parts of a solution can be computed efficiently; and a specific type of iterative method which is particularly effective at fine-tuning the solution parts that require a high resolution. We will first consider this second component.

For the rest of the section, we assume that A is a symmetric positive definite matrix arising from the discretization of a system of elliptic partial differential equations. A large class of iterative solvers for (2.69) can be written in the form

$$\boldsymbol{x}_{k+1} = \mathrm{C}\boldsymbol{x}_k + \boldsymbol{c}, \tag{2.74}$$

where C is typically a matrix derived from a splitting of A and $\boldsymbol{c}$ is a constant vector involving the right-hand side $\boldsymbol{b}$. Examples of such solvers are the Jacobi method, successive over relaxation (SOR), or the Gauss-Seidel method [131, 114]. We assume the method is consistent, which means that $\boldsymbol{x}^* = C\boldsymbol{x}^* + c$, where $\boldsymbol{x}^*$ is the solution of (2.69). It is then easy to show that the following equation holds for the errors of successive iterates:

$$\boldsymbol{e}_{k+1} = C\boldsymbol{e}_k \, .$$

We naturally want to choose the matrix C so that the error is reduced in each iteration and $\boldsymbol{x}_k \to \boldsymbol{x}^*$ as $k \to \infty$. To establish a criterion for convergence, we define the *spectral radius* of C by

$$Q(C) := \max_i |\lambda_i(C)| \, ,$$

where $\lambda_1(C), \ldots, \lambda_n(C)$ are the eigenvalues of C. It can be shown that (2.74) converges for all $\boldsymbol{x}_0 \in \mathsf{R}^n$ if and only if $Q(C) < 1$. For certain iterative solvers, we can identify not just their overall convergence behaviour but how they affect different components of the error $\boldsymbol{e}_k$. Since A is symmetric, its eigenvectors can be chosen to be an orthonormal basis of R^n and we can express the error as a linear combination of these eigenvectors. For matrices arising from, for example, the discretization of elliptic PDEs, the eigenvectors can usually be interpreted as the point-wise values of oscillatory functions on the discretized domain Ω_h, which vary in their frequencies. Some iterative methods of the type (2.74) eliminate particularly fast those components of $\boldsymbol{e}_k$ which lie in the range of high-frequency eigenvectors. On the other hand, those same solvers are typically very slow at reducing the "smooth" error components – those that correspond to low-frequency eigenvectors. Such iterative solvers are therefore called *smoothers*. The damped Jacobi method, as well as the SOR and various forms of the Gauss-Seidel method fall into this category [131, 114, 61]. In multigrid methods, they are used specifically because of their ability to effectively reduce high-frequency errors and are complemented by another essential component of multigrid, the coarse-grid correction, which takes care of the low-frequency errors and which we will look at next.

The concept that gives multigrid methods their name is the use of several levels of

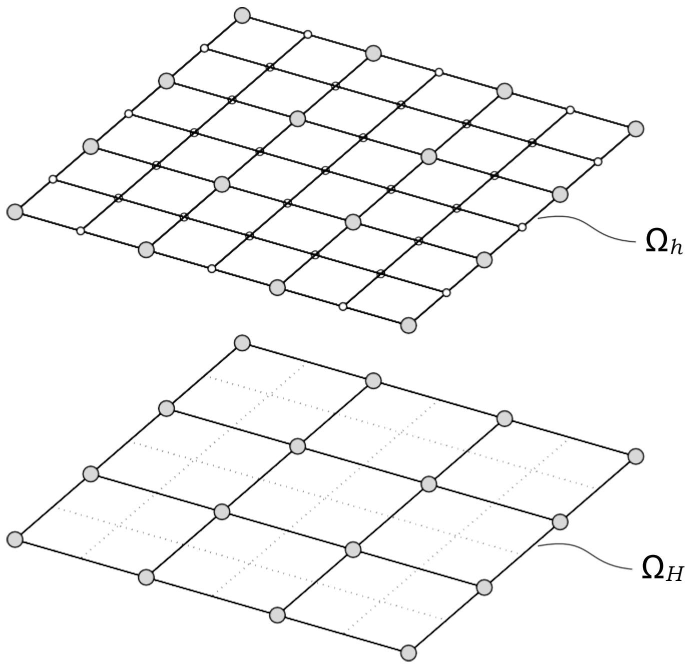


Figure 2.3: Basic example of a two-grid hierarchy. Fine-grid nodes are coloured white; coarse-grid nodes (and coinciding fine-grid nodes) are coloured grey.

meshes, or *grids*, each a discretization of the problem domain. Grids at lower levels have a coarser discretization resolution, with the mesh of the original problem at the "top" of the grid hierarchy. We will use the terms "grid" and "mesh" interchangeably. The top-level grid is commonly called the *fine* grid, whereas the other grids are called *coarse*. The essential characteristics of this multilevel scheme are best illustrated on the two-grid problem. Assume that we have two grids for the same domain, but of different resolution: the fine grid, which is basically our "original" grid on which (2.69) is defined and which we denote by Ω_h, and the coarse grid Ω_H . We denote by n_h and n_H , with $n_h > n_H$, the number of degrees of freedom for each grid. We use the same subscript notation for matrices defined on either grid, e.g. $\mathrm{A}_h := \mathrm{A}$, but superscripts for vectors, e.g. $\boldsymbol{b}^h := \boldsymbol{b}$. Figure 2.3 shows a simple example of a two-grid hierarchy, where Ω_H is created from Ω_h by halving the number of elements in each grid direction.

If we perform a certain number of smoother iterations on the fine grid, we assume that the resulting error $\boldsymbol{e}$ has mostly low-frequency, or smooth, components. The resolution of the coarser grid might therefore suffice to accurately represent this error. Recalling the identity (2.70), we could therefore solve $\mathrm{A}\boldsymbol{e} = \boldsymbol{r}$ on the coarser grid, which can be done more efficiently, since the size of the system is reduced. We could then obtain the solution by updating $\boldsymbol{x} \leftarrow \boldsymbol{x}+\boldsymbol{e}$. It becomes clear that we need to differentiate between the system we are solving on the fine grid, $\mathrm{A}_h\boldsymbol{x}^h = \boldsymbol{b}^h$, and the one on the coarse-grid, which is $\mathrm{A}_H\, \boldsymbol{e}^H = \boldsymbol{r}^H$. The matrix $\mathrm{A}_h = \mathrm{A} \in \mathsf{R}^{n_h\times n_h}$ and right-hand side vector $\boldsymbol{b}^h = \boldsymbol{b} \in \mathsf{R}^{n_h}$ are given as part of the problem specification, for example by an FE discretization on Ω_h. The question is how we should define $\mathrm{A}_H \in \mathsf{R}^{n_H\times n_H}$ and $\boldsymbol{b}^H \in \mathsf{R}^{n_H}$ on the coarse grid. Furthermore, how do we obtain a coarse-grid version $\boldsymbol{r}^H$ of the residual $\boldsymbol{r}^h = \boldsymbol{b}^h - \mathrm{A}_h\boldsymbol{x}^h$ and how do we map the solution $\boldsymbol{e}^H$ of $\mathrm{A}_H\boldsymbol{e}^H = \boldsymbol{r}^H$ back to the fine grid? For this purpose, we define transfer operators

$$\mathrm{I}_h^H : \mathsf{R}^{n_h} \longrightarrow \mathsf{R}^{n_H}, \quad \mathrm{I}_H^h : \mathsf{R}^{n_H} \longrightarrow \mathsf{R}^{n_h} .$$

We call I_h^H the *restriction operator* and I_H^h the *prolongation operator*. Let us consider the two-dimensional example shown in Figure 2.3. The grids Ω_h and Ω_H are both regular with uniform grid spacing in each direction. Ω_h is obtained by dividing each element in Ω_H into 4 squares. In this case, we can define the prolongation operator I_H^h through simple piecewise linear interpolation. If we assume for simplicity's sake that we have one degree of freedom for each node and consider a node that does not lie on the boundary of the mesh, we can express the contribution of a coarse-grid node to the 9 fine-grid nodes in its vicinity by a *stencil*, which takes the form

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

In three dimensions, a coarse-grid node contributes to 27 fine grid nodes and the stencil is

$$\frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \frac{1}{8}\begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix} \quad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} ,$$

where the left and right matrices correspond to the fine-grid layers above and below the coarse-grid node, which itself lies on the layer corresponding to the central matrix. In order to restrict a vector defined on the fine grid down to the coarse grid, we could choose a trivial injection and simply assign to each coarse-grid node the value of the fine-grid node that coincides with it. However, we achieve better results if we define the restriction operator by

$$I_h^H = \gamma (I_H^h)^\top \tag{2.75}$$

for some $\gamma > 0$. This *full weighting* approach effectively defines a coarse-grid value as a weighted sum of its surrounding fine-grid values. Extending the presented approach for the transfer operators to multiple degrees of freedom per node and boundary conditions that eliminate some or all DOFs for certain boundary nodes is fairly straightforward, albeit a bit cumbersome, and we omit the details. Note that a prolongation operator defined in the way just described has full rank.

We still need to define the coarse-grid system matrix A_H. This is done via a Galerkin projection [114]:

$$A_H := I_h^H A_h I_H^h . \tag{2.76}$$

If the system originates from an FE discretization, one can in fact show that $A_H\, \boldsymbol{x} = \boldsymbol{b}^H$ with A_H defined as above and $\boldsymbol{b}^H := I_h^H \boldsymbol{b}^h$ is exactly the system one would obtain from an FE discretization on the coarse mesh using piecewise bilinear basis functions.

Now all the components are in place and we can outline the two-grid iterative scheme. We begin by performing $\omega_1 \in \mathbb{N}$ iterations of a smoother with matrix C. Let $\boldsymbol{r}^h$ be the residual after this *pre-smoothing* step. The next step is called *coarse-grid correction*: We define the restricted residual by $\boldsymbol{r}^H := I_h^H \boldsymbol{r}^h$ and solve $A_H\, \boldsymbol{e}^H = \boldsymbol{r}^H$. For now, let us assume that we solve the restricted system exactly, so that we can write $\boldsymbol{e}^H = A_H^{-1} \boldsymbol{r}^H$. We interpolate the solution $\boldsymbol{e}^H$, defining $\boldsymbol{e}^h := I_H^h \boldsymbol{e}^H$ and use this to update $\boldsymbol{x}^h$. As a final (optional) step, we perform $\omega_2 \in \mathbb{N}$ *post-smoothing* iterations. Since what we have described is a sequence of linear operations, we can summarize one iteration of the two-grid scheme as an iterative solver of the form (2.74). We get

$$\boldsymbol{e}_{k+1} = T\boldsymbol{e}_k ,$$

where the iteration matrix of the two-grid scheme is given by

$$T = C^{\omega_2} \Big(\underbrace{I - I_H^h A_H^{-1} I_h^H A_h}_{=:T_{h,H}} \Big) C^{\omega_1} . \tag{2.77}$$

The matrix $T_{h,H}$ is called *coarse-grid correction operator* and it constitutes one of the two core elements of the two-grid scheme, the other being (pre- and post-)smoothing. A single iteration of the smoother within a smoothing step is sometimes called a *sweep*.

For our choice of transfer operators and any of the smoothers mentioned earlier on, it can be shown that the two-grid scheme converges. A general set of necessary conditions for convergence of the two-grid scheme is given by Theorem 2.4.4 further below, which can also be found in [114, 131]. We will first need the two definitions below in order to identify an appropriate smoother and transfer operators. They employ the norms $\|\cdot\|_{D_h}$ and $\|\cdot\|_{D_h^{-1}}$, where D_h is the (positive definite) main diagonal matrix of A_h. These norms are defined analogously to the energy norm (2.71).

Definition 2.4.2. We say that a smoother, or the corresponding matrix C, has the *smoothing property* if, for some $\alpha > 0$ which is independent of the mesh size,

$$\|C\boldsymbol{e}^h\|^2_{A_h} \leq \|\boldsymbol{e}^h\|^2_{A_h} - \alpha \|A_h \boldsymbol{e}^h\|^2_{D_h^{-1}} . \tag{2.78}$$

Definition 2.4.3. We say that a prolongation operator for the two-grid scheme has the *approximation property* if, for some $\beta > 0$ which is independent of the mesh size,

$$\min_{\boldsymbol{e}^H \in \mathbb{R}^{n_H}} \| \boldsymbol{e}^h - \mathrm{I}_H^h \boldsymbol{e}^H \|_{\mathrm{D}_h} \leq \beta \| \mathrm{A}_h \boldsymbol{e}^h \|_{\mathrm{D}_h^{-1}} . \tag{2.79}$$

Theorem 2.4.4. *Let A_h be symmetric positive definite. Let the prolongation operator I_H^h have full rank and the restriction operator I_h^H satisfy (2.75). Let the coarse grid system matrix be given by the Galerkin projection (2.76). Further, assume that the matrix C has the smoothing property and I_H^h has the approximation property with constants α and β, respectively. Then, we have $\alpha \leq \beta$ and*

$$\| \mathrm{T} \|_{\mathrm{A}_h} \leq \sqrt{1 - \frac{\alpha}{\beta}} =: \eta < 1 .$$

Consequently, the energy norm of the error is reduced in each iteration of the two-grid scheme by at least a factor of η, as

$$\| \boldsymbol{e}_{k+1} \|_{\mathrm{A}_h} \leq \| \mathrm{T} \|_{\mathrm{A}_h} \| \boldsymbol{e}_k \|_{\mathrm{A}_h} \leq \eta \| \boldsymbol{e}_k \|_{\mathrm{A}_h} .$$

Proof. See [114, (Theorem 13.3)] or [131, (Theorem 6.52)]. □

The key to the two-grid scheme's convergence is the combination of the smoother, which effectively reduces the high-frequency parts of the error, and the coarse-grid correction operator $\mathrm{T}_{h,H}$, see (2.77), which eliminates the smooth error components. As mentioned earlier, it is plausible to assume that the range of the prolongation operator I_H^h closely overlaps with the space of low-frequency eigenvectors of A_h. If it has full rank and (2.75) and (2.76) are satisfied, I_H^h can be shown to be A_h-orthogonal to the range of the coarse-grid correction operator, i.e. $\mathrm{range}(\mathrm{T}_{h,H}) \perp_{\mathrm{A}_h} \mathrm{range}(\mathrm{I}_H^h)$. Moreover, it can be shown that $\mathrm{T}_{h,H}$ is an A_h-orthogonal projection and that

$$\ker(\mathrm{T}_{h,H}) = \mathrm{range}(\mathrm{I}_H^h) , \tag{2.80}$$

where $\ker(\mathrm{T}_{h,H})$ is the kernel of $\mathrm{T}_{h,H}$. So indeed, we see that any error components that lie in the range of the prolongation operator are eliminated entirely by the coarse-grid correction.

In practice, the two-grid scheme is not viable, as n_H is usually still too large to yield any significant savings in computational work if we solve $\mathrm{A}_H \boldsymbol{e}^H = \boldsymbol{r}^H$ directly. Instead, we replace A_H^{-1} by a pre- and/or post-smoothing step and a coarse-grid correction on an even coarser grid. We repeat this recursively until we reach a grid that is coarse enough that the size of the system is sufficiently small that a direct solution is comparatively cheap. This recursive scheme constitutes the actual multigrid method. It requires a hierarchy of grids of decreasing size. Consider a two-dimensional regular mesh of m_x by m_y square elements. Refine it by dividing each element into 4 smaller elements of equal size and repeat several times to get a total of L grids, each with $m_l = 4^{(l-1)} m_x m_y$ elements and $N_l := \left(2^{(l-1)} m_x + 1\right)\left(2^{(l-1)} m_y + 1\right)$ nodes, where $l = 1, \ldots, L$. To go back and forth between each mesh level, we define prolongation and restriction operators I_{l-1}^{l} and I_{l}^{l-1}, respectively, where $l = L, \ldots, 2$. Algorithm 2 shows the so-called V-cycle multigrid method. We can generalize it by changing Line 7 to perform any fixed number of multigrid calls on the next lower level. For example, two successive calls would lead to what is called the W-cycle multigrid method.

A similar convergence result to Theorem 2.4.4 can be obtained for the multigrid method, see for example [61]. Crucially, the contraction factor η by which the energy norm is reduced in each iteration is independent of n_h. Therefore, the multigrid method is an optimal solver, as the number of iterations required to get the error below a certain threshold does not depend on the size of the problem. Of course, the overall computational cost still does. For sparse systems, the complexity of the multigrid method is thus in $O(n)$ [131]. In contrast, other iterative solvers such as (un-preconditioned) Krylov solvers require more iterations as the size of the system, and with it the condition number of the matrix, increases, giving them

Algorithm 2 V-Cycle Multigrid Method

Let I_{l-1}^{l} and I_{l}^{l-1}, $l = L, \dots, 2$, be transfer operators. Let $\mathrm{smooth}(A, \boldsymbol{x}_0, \boldsymbol{b}, \omega)$ denote ω iterations of a smoother for the system $A\boldsymbol{x} = \boldsymbol{b}$ with initial guess $\boldsymbol{x}_0$. Assume that system matrices are defined for each coarse level by $A_{l-1} = I_{l}^{l-1} A_l I_{l-1}^{l}$ for $l = L, \dots, 2$.

```
1: function $MG(l, A_l, \boldsymbol{x}^l, \boldsymbol{b}^l, \omega_1, \omega_2)$
2:     if $l = 1$ then
3:         $\boldsymbol{x}^1 \leftarrow A_1^{-1}\boldsymbol{b}^1$
4:     else
5:         $\boldsymbol{x}^l \leftarrow \mathrm{smooth}(A_l, \boldsymbol{x}^l, \boldsymbol{b}^l, \omega_1)$    ▷ pre-smoothing
6:         $\boldsymbol{r}^{l-1} = I_l^{l-1}(\boldsymbol{b}^l - A_l\boldsymbol{x}^l)$
7:         $\boldsymbol{e}^{l-1} = MG(l-1, A_{l-1}, \boldsymbol{0}, \boldsymbol{r}^{l-1}, \omega_1, \omega_2)$    ▷ coarse grid correction
8:         $\boldsymbol{x}^l \leftarrow \boldsymbol{x}^l + I_{l-1}^{l}\boldsymbol{e}^{l-1}$
9:         $\boldsymbol{x}^l \leftarrow \mathrm{smooth}(A_l, \boldsymbol{x}^l, \boldsymbol{b}, \omega_2)$    ▷ post-smoothing
10:    end if
11:    return $\boldsymbol{x}^l$
12: end function
```

$O(n^2)$ complexity.

However, although η in Theorem 2.4.4 is constant, it might still be very close to 1. So while the cost of the multigrid method scales linearly with n, it might only outperform other iterative solvers for extremely large n. The algorithm can be optimized by tailoring its parameters – such as γ in (2.75), the number of pre- and post-smoothing sweeps, or even parameters of the smoother itself – to the specific problem. This requirement makes it less robust and not suitable as a black box method on its own. Therefore, the multigrid method is commonly used as a preconditioner in Krylov solvers as discussed earlier in this section. The V-cycle can in theory be written as (the inverse of) a preconditioning matrix M^{-1}, even though the resulting expression would be rather unwieldy. The important thing to note is that, in order for it to represent a symmetric operator, which is important for both the CG and MINRES method, the number of pre- and post-sweeps should be equal, that is $\omega_1 = \omega_2$.

Smoothed Aggregation AMG

In the form discussed so far, the multigrid method assumes that a hierarchy of grids is given for which the prolongation operator can be defined via interpolation.

For structured meshes, especially with elements of equal size, such a hierarchy can be obtained easily enough through bisection of elements. However, things change once the domain geometry no longer allows for such a structured mesh. Naturally, therefore, the question arises of how to generalize the concept of fine and coarse grids to this case, or in fact, to cases where the problem does not correspond to any kind of grid at all. Methods which address this problem by defining coarse spaces and transfer operators based only on the structure of the system matrix are called *algebraic multigrid* (AMG) methods – as opposed to the *geometric* multigrid (GMG) variant discussed above. AMG methods can even have an advantage over the geometric approach when both are applicable, for example in the case of strong material anisotropy [47].

Note that the prerequisites of Theorem 2.4.4 are formulated in terms of operators and do not rely on the existence of an underlying "physical" mesh. Therefore, the theorem is applicable to any appropriate choice of prolongation operator that has the approximation property (2.79) and any smoother that satisfies (2.78). Since for elliptic problems, the typical smoothers used on structured discretizations also work on unstructured meshes, the main challenge is to construct several levels of coarse spaces and corresponding transfer operators. Each step from a higher (finer) to a lower (coarser) level should give a notable reduction in the number of DOFs. Furthermore, a more abstract notion of "smoothness" needs to be defined if we want to find prolongation operators whose range covers "smooth" error components.

A range of different AMG techniques have been developed, starting with what is now often called classic AMG methods [113, 30]. For a brief introduction, see for example [47], which also points to some other approaches. In particular, it mentions the *smoothed aggregation* (SA) method proposed by Vaněk, Mandel, and Brezina [124, 123] as a method that is "robust and efficient over a wide variety of problems". It is this latter method which we adopt in our algorithm. Since a detailed description would go beyond the scope of this thesis, we will only give an outline of the basic concepts. We refer to [124] for details of the basic algorithm, to [123] for the convergence analysis and to [32] for the adaptive smoothed aggregation method. Peetz and Elbanna also recently employed the SA method in the context of topology optimization [96]. For a unified analysis of different AMG techniques, see [133].

The first task we will address is that of defining the coarse space. We can again limit ourselves to the two-grid case, denoting fine and coarse grid by subscript h and H, respectively. In geometric multigrid methods, fine-grid nodes are locally condensed into coarse-grid nodes on the basis that neighbouring nodes are interdependent to some degree. This physical coupling is reflected by the off-diagonal elements of the matrix $A = [a_{ij}]_{i,j=1,\dots,n}$. Assuming for now that there is just one DOF per node, two nodes with indices i and j are neighbours in the algebraic sense if the matrix entry a_{ij} is non-zero. We say they are *strongly coupled* neighbours if $|a_{ij}|$ is large enough compared to the diagonal entries a_{ii} and a_{jj}. The SA method defines the *strongly-coupled neighbourhood of node i* as

$$N_i(\epsilon_N) := \left\{ j : |a_{ij}| \geq \epsilon_N \sqrt{a_{ii}a_{jj}} \right\},$$

where $\epsilon_N \in [0, 1)$ is a threshold parameter. For the case of multiple DOFs per node, we generalize this definition as follows. Let us denote the DOFs of nodes i and j by the vectors $\boldsymbol{i}$ and $\boldsymbol{j}$, respectively. Instead of single elements of A we now need to consider submatrices corresponding to index-vectors, denoted by, for example, $A_{\boldsymbol{ij}}$. Then, we define the strongly-coupled neighbourhood of node i with DOFs $\boldsymbol{i}$ by[1]

$$N_i(\epsilon_N) := \left\{ j : \|A_{\boldsymbol{ij}}\|_F^2 \geq \epsilon_N \|A_{\boldsymbol{ii}}\|_F \|A_{\boldsymbol{jj}}\|_F \right\}, \tag{2.81}$$

where $\|\cdot\|_F$ is the Frobenius norm. With this neighbourhood relationship in place, we can define a coarse grid, as it were, by partitioning the set of all nodes into disjoint subsets $C_1, \dots, C_{N_H}$ of strongly-coupled neighbours, called *aggregates*. Each

[1] We deviate from the definition given in [124], which involves the spectral radius of the matrix $A_{\boldsymbol{ii}}^{-1/2} A_{\boldsymbol{ij}} A_{\boldsymbol{jj}}^{-1/2}$. The additional implementation effort and computational work for this formula compared to ours does not seem warranted, judging by our numerical experiments.

aggregate represents a coarse-grid node. Next, we need to construct the transfer operators. The approach dictated by SA is motivated by the consideration that the range of the prolongation operator should include the smooth error components. Therefore, we need to consider the – so far geometric – notion of smoothness in a more abstract way.

The crucial feature, in terms of the convergence analysis, of smooth error components is that they are not effectively reduced by the smoother. We can adhere to this definition, noting that errors which are smooth in this sense do not generally have to be smooth in the geometric sense. The term *algebraically smooth* is therefore commonly used in this context. For the remainder of this section, "smoothness" will generally mean algebraic smoothness unless otherwise specified. In many cases, the smooth error components lie in the space of eigenvectors corresponding to the smallest eigenvalues of the positive definite matrix A. Consequently, if an error $\boldsymbol{e}$ is smooth, we have $A\boldsymbol{e} \approx 0$, which is why this (somewhat vaguely defined) subspace is often called the *near kernel* of A.

Let $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d_k}]$ be a matrix of near kernel basis vectors. In the standard version of SA, V has to be supplied by the user. For isotropic second order elasticity problems, (such as the one in Section 2.2), the natural – and best [123, 32] – choice would be the rigid body modes of the elastic structure. The rotational modes would require additional preprocessing steps for non-trivial mesh geometry. However, a basis for the translational modes is always readily available in the form of d linearly independent vectors of constant displacements. In three dimensions, this would give us $d_k = d = 3$ and the near kernel

$$V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ & \vdots & \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.82}$$

where we assume that all rows corresponding to displacements fixed by the boundary conditions have been eliminated. In our implementation, a near-kernel defined as above gives the best results, but the following derivations are valid for any choice of V.

The SA prolongation operator is constructed in two steps. First, we determine a *tentative prolongation operator* $P^h_H : \mathbb{R}^{n_H} \longrightarrow \mathbb{R}^{n_h}$, based on the principle that V should lie in its range and thus have a coarse-grid representation. This means we want to find P^h_H and a coarse-grid near-kernel $V_H \in \mathbb{R}^{n_H \times d_k}$ such that

$$V =: V_h = P^h_H V_H. \tag{2.83}$$

For a vector $\boldsymbol{v} \in \mathbb{R}^{n_h}$ defined in the fine space, let $\boldsymbol{v}_{\mathcal{C}_j}, j \in \{1, \dots, N_H\}$ denote the vector of all fine-grid DOFs that correspond to the jth aggregate. If $\boldsymbol{w} \in \mathbb{R}^{n_H}$ is a coarse-grid vector, $\boldsymbol{w_j}$ will denote all DOFs of coarse-grid node j. Although we use the same index-vector notation for fine-grid vectors, it should be clear from context whether we refer to fine- or coarse-grid DOFs. Also note that the number of DOFs per coarse-grid node is not generally d, as for the fine grid nodes, but d_k. In addition to satisfying (2.83), the tentative prolongator should also have columns with localized support; more specifically, we only want overlap between the columns of P^h_H which correspond to neighbouring aggregates/coarse nodes, much in the same way as for the geometric interpolation prolongator. We could achieve both by choosing $(\mathrm{P}^h_H)_{\boldsymbol{j}} = [(\boldsymbol{v}_1)_{\mathrm{C}_j}, \dots, (\boldsymbol{v}_{d_k})_{\mathrm{C}_j}]$, that is, by defining the prolongator columns corresponding to coarse-grid node j as the restriction of the near-kernel V_h to aggregate C_j. The components of the coarse-grid near-kernel at node j would then simply be $(\mathrm{V}_H)_{\boldsymbol{j}:} = \mathrm{I}$. However, to ensure orthonormality of the columns of P^h_H, we instead use a QR factorisation of the localized near-kernel and define the prolongator and coarse-grid near-kernel as

$$\begin{aligned}
(\mathrm{P}^h_H)_{\mathrm{C}_j\boldsymbol{j}} &:= \mathrm{Q}_j \in \mathsf{R}^{|\mathrm{C}_j| \times d_k}, \\
(\mathrm{V}_H)_{\boldsymbol{j}:} &:= \mathrm{R}_j \in \mathsf{R}^{d_k \times d_k}, \\
\text{where} \quad (\mathrm{V}_h)_{\mathrm{C}_j,:} &= \mathrm{Q}_j\mathrm{R}_j \quad \text{and} \quad (\mathrm{Q}_j)^\mathsf{T}\mathrm{Q}_j = \mathrm{I}, \\
j &= 1, \dots, N_H.
\end{aligned} \tag{2.84}$$

Occasionally, $(\mathrm{V}_h)_{\mathrm{C}_j,:}$, the near-kernel localized to the jth aggregate, might not have full rank. This can occur whenever the number of DOFs of the coarse-grid node j is larger than the number of fine-grid DOFs of the entire aggregate, for example when $d < d_k$, i.e. the dimension of the near-kernel is larger than the problem dimension, or when fine-grid nodes have fewer than d DOFs due to boundary conditions. In that case, we drop as many columns from Q_j and rows from R_j as necessary to ensure that Q_j has full rank, effectively reducing the number of DOFs for coarse-grid node j. We thereby guarantee that P^h_H has full rank and also avoid excess storage and operator complexity.

As mentioned before, there is a second step to constructing the SA prolongation operator. If we compare the tentative prolongator as defined in (2.84) to the prolongation operator in the geometric multigrid method at the beginning of this section, we notice that there is no overlap at all between the supports of the columns of P^h_H for different aggregates. While a localized support of each coarse node was one of our requirements, a certain degree of “smoothness” of the column vectors of P^h_H is important for the overall convergence of the SA multigrid method (see [123] for details). To obtain the final prolongation operator, we therefore apply a smoother to the tentative prolongator:

$$I_H^h := S_h P_H^h . \tag{2.85}$$

A simple choice for the prolongation smoother S_h, which is also used in the convergence analysis in [123], is

$$S_h = I - \frac{4}{3\bar{\lambda}_h} A_h ,$$

where $\bar{\lambda}_h$ is an upper bound on the spectral radius of A_h, which can be obtained, for example, by the Gershgorin theorem, e.g. [114, (Thm. 4.6)]. An alternative prolongation smoother that yielded better results in our numerical experiments is given by the weighted Jacobi smoother

$$S_h = I - \frac{2}{3} D_h^{-1} A_h , \tag{2.86}$$

where D_h is the main diagonal of A_h. Having thus defined a prolongator, we define the restriction operator simply as its transpose.

Algorithm 3 summarizes the SA technique for constructing the coarse spaces and transfer operators, generalizing the concepts presented above to the multi-level case. We assume that information is available on which DOFs correspond to which nodes in the fine grid. The full SA multigrid method consists of Algorithm 3 followed by the standard multigrid method, see Algorithm 2.

Algorithm 3 Smoothed Aggregation

Let $L \geq 1$ be the maximum number of coarse levels to create and let $n_{\min}$ be the minimum number of total DOFs a coarse level should have. Let $V \in \mathbb{R}^{n \times d_k}$ be a basis for the near-kernel of $A =: A_L$.

```
1: for $l = L - 1, \ldots, 1$ do
2:     Create aggregates $C_1, \ldots, C_{N_l}$ of strongly-coupled neighbouring nodes
3:     Define $P_l^{l+1} \in \mathbb{R}^{n_{l+1} \times n_l}$ and $V_l \in \mathbb{R}^{n_l \times d_k}$ such that $V_{l+1} = P_l^{l+1} V_l$
4:     if $n_l < n_{\min}$ then
5:         $L \leftarrow L - l$, redefine all level indices accordingly
6:         stop
7:     end if
8:     $I_l^{l+1} := S_{l+1} P_l^{l+1}$ and $I_{l+1}^{l} := (I_l^{l+1})^\top$
9: end for
```

So far, we have assumed that we already know the near-kernel V. In cases where we do not know all basis vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d_k}$ that should define our coarse space, we can use the multigrid method itself to identify such vectors. This is the principle behind the *adaptive smoothed aggregation* (αSA) method [32]. It can obtain a basis of the near-kernel for a given matrix A even when not a single near-kernel vector is known a priori. Since we assume the existence of an underlying physical grid, basis vectors of the form (2.82) can – and in fact should [32] – always be used. We therefore focus on the task of "enriching" a pre-existing near-kernel basis by additional vectors.

Recall the basic multigrid paradigm as illustrated by the two-grid scheme: any error that is not effectively reduced to zero by the smoothing matrix is (approximately) in the range of the prolongation operator. Since the latter coincides with the kernel of the coarse-grid correction operator, see (2.80), it is eliminated in the coarse-grid correction step. If there is an error vector which the multigrid method, as a combination of smoothing and coarse-grid correction, does not effectively reduce, we can infer that it is missing from the range of the prolongator I_H^h. Consequently, if we augment $\mathrm{range}(\mathrm{I}_H^h)$ so as to include this vector, it should improve our multigrid method.

Following this argumentation, the first step of the αSA is to choose a random initial vector and apply the multigrid V-cycle a fixed number of times for $\mathrm{A}\boldsymbol{v} = \mathbf{0}$. Whatever error remains after this, if it has not been reduced by a certain (very small) factor, is a good candidate for a new near-kernel vector. Due to the zero right-hand side, the error is trivially equal to the result of the V-cycle itself, which we denote by $\boldsymbol{v}^L$. We will refer to $\boldsymbol{v}^L$ simply as a *candidate*.

In the next step of the αSA method, we improve the candidate by checking how effectively it is reduced on the lower levels $L-1, L-2,$ We first update the near-kernel matrix on the finest level by appending $\boldsymbol{v}^L$. We set $\hat{\mathrm{V}} := [\mathrm{V}, \boldsymbol{v}^L]$ and construct new transfer operators based on $\hat{\mathrm{V}}$, according to (2.83) and (2.85). By enriching the fine-grid near-kernel, we have increased the number of DOFs on grid $L-1$, so that we also need to update the system matrix A_{L-1} accordingly, using the Galerkin projection with the *enriched* transfer operators. Along with a new coarse-grid near-kernel candidate, we have created a new coarse-grid problem on level $L-1$. To further improve the candidate, we perform a fixed number of multigrid iterations, now starting at level $L-1$. This requires some technical adjustments, since the transfer operators between levels $L-1$ and $L-2$ are no longer valid, but we refer to [32] for details[1]. We use the vector returned after the last multigrid iteration as our new candidate on level $L-1$ and denote it by $\boldsymbol{v}^{L-1}$. Then, we repeat the above process: we enrich the near-kernel, construct new, enriched transfer operators between $L-1$ and $L-2$, update A_{L-2} and further improve the candidate on level $L-2$. We continue in this way until we reach level 2. Once we have obtained an improved candidate $\boldsymbol{v}^2$ on this level, we interpolate it all the way back up to the fine grid and overwrite our original fine-level candidate, $\boldsymbol{v}^L \leftarrow I^L_{L-1} I^{L-1}_{L-2} \ldots I^3_2 \, \boldsymbol{v}^2$.

Whereas $\hat{V}$ was just a provisional improved version of the near-kernel, we now permanently enrich the near-kernel using the updated fine-level candidate: $V \leftarrow [V, \boldsymbol{v}^L]$. Finally, we set up all coarse levels from scratch, using the regular SA method, see Algorithm 3.

The adaptive smoothed aggregation procedure described above can be repeated to give us any number of additional candidates. There is obviously a trade-off between the improved convergence behaviour expected from this and the growing operator complexity on all lower levels, as the numbers of DOFs and thus the size of the coarse spaces increase with the dimension of the near-kernel. It is not at all obvious, however, which number of candidates is right for a given problem. Another question arises when using αSA inside an iterative optimization method where the system matrix changes frequently, but we cannot reasonably consider reconstructing

[1]We only highlight that we also update the lower level system matrices after these adjustments. While this step is explicitly omitted in the original algorithm, we found that it was required to maintain the convergence of the lower-level multigrid method.

the transfer operators for each new system, as the overhead of both the SA and αSA algorithm is prohibitive. Section 3.2.3 presents numerical experiments which investigate these questions and inform the AMG strategy we ultimately chose to follow.

CHAPTER 3

MULTIGRID BARRIER METHODS FOR MINIMUM COMPLIANCE OPTIMIZATION

We now turn to the main contribution of this thesis. We apply the optimization methods described in Sections 2.1.3 and 2.1.4 to the VTS problem, using multigrid-preconditioned Krylov solvers as discussed in Section 2.4. Section 3.1 details our implementation of the IP and PBM method. We can easily extend the latter to consider unilateral contact constraints due to the fact that we apply it to the dual VTS problem.

In Section 3.2, we present various example problems and numerical results, on the basis of which we compare the performance of the IP and PBM method, as well as that of the OC method covered in Section 2.3.3. We also address the question of which Krylov solver to use within each method. Large scale problems involving several million finite elements are solved, both for uniformly structured and unstructured meshes. The latter warrant a discussion of different strategies by which to employ the SA AMG method as a preconditioner. Lastly, we include results for problems involving unilateral contact.

An obvious question that arises is whether our methods can be applied not just to the VTS problem but also to the SIMP problem. We comment on this in Section 3.3, noting the limitations of our approach and complications arising from the SIMP formulation.

3.1 Optimization Algorithms

In the following, we give a detailed description of our implementation of the IP and PBM method for the VTS problem. In particular, we apply the PBM method to the dual problem, as its structure is more amenable to the PBM methodology. Since primal and dual variables are not treated differently in the IP method, as opposed to the PBM method, it does not matter whether we apply it to the primal or dual formulation. Since the primal problem is the starting point for any potential extensions, for example to SIMP-based compliance minimization, it is the one we will use to derive our IP algorithm. At the end of Section 3.1.1, we show that we can apply the IP method to the dual problem and obtain the same algorithm.

For each optimization algorithm, we derive the linear system that needs to be solved in every iteration. We manipulate this system by means of a Schur complement to obtain a positive definite system with a sparsity structure resembling the stiffness matrix associated with the problem. This motivates the use of standard transfer operators when employing the multigrid method as a preconditioner, following [80]. Details of how we solve the linear system in each method are given in Section 3.1.4. The efficacy of this approach is confirmed by the numerical experiments in Section 3.2.

3.1.1 Primal-Dual Interior Point Method for the VTS Problem

The primal-dual IP method we use to solve (2.48) is based on earlier contributions by Kočvara and Mohammed [80] and Jarre, Kočvara, and Zowe [71]. Many features of the algorithm proposed in [80] had to be changed to make it more performant and viable for 3D problems. Maar and Schulz also used an IP method for topology optimization together with a multigrid method to solve the linear systems [85].

Their approach, however, involved solving an indefinite system, requiring the use of a much more complicated multigrid scheme.

We recall the KKT conditions (2.51) for the VTS problem, again eliminating the adjoint variable by (2.52):

$$\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f} = \mathbf{0}\,, \tag{3.1}$$

$$\boldsymbol{\rho}^{\mathsf{T}}\boldsymbol{a} - V = 0\,, \tag{3.2}$$

$$\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{B}(\boldsymbol{u}) - \alpha\,\boldsymbol{a} + \underline{\boldsymbol{\nu}} - \overline{\boldsymbol{\nu}} = \mathbf{0}\,, \tag{3.3}$$

$$\left(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}\right)^{\mathsf{T}}\underline{\boldsymbol{\nu}} = 0\,, \tag{3.4}$$

$$(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho})^{\mathsf{T}}\,\overline{\boldsymbol{\nu}} = 0\,, \tag{3.5}$$

$$(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho}),\ \left(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}\right),\ \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}} \geq \mathbf{0}\,, \tag{3.6}$$

where we have again used the notation $\mathrm{B}(\boldsymbol{u}) = [\mathrm{K}_1\boldsymbol{u}, \ldots, \mathrm{K}_m\boldsymbol{u}]$. Anticipating later developments, we introduce the variable $\tilde{\alpha} := -\alpha$. This substitution will guarantee symmetry of the system (3.9) further below. We then perturb the complementarity conditions, replacing them with

$$(\rho_i - \underline{\rho}_i)\underline{\nu}_i = \underline{s}, \quad i = 1, \ldots, m,$$

$$(\overline{\rho}_i - \rho_i)\overline{\nu}_i = \overline{s}, \quad i = 1, \ldots, m\,.$$

This gives us the following associated residuals:

$$\tilde{\boldsymbol{r}}(\boldsymbol{u}, \tilde{\alpha}, \boldsymbol{\rho}, \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}) = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \\ \boldsymbol{r}_4 \\ \boldsymbol{r}_5 \end{bmatrix} := \begin{bmatrix} \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} - \boldsymbol{f} \\ \boldsymbol{\rho}^{\mathsf{T}}\boldsymbol{a} - V \\ \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{B}(\boldsymbol{u}) + \tilde{\alpha}\,\boldsymbol{a} + \underline{\boldsymbol{\nu}} - \overline{\boldsymbol{\nu}} \\ \left[(\rho_i - \underline{\rho}_i)\underline{\nu}_i - \underline{s}\right]_{i=1,\ldots,m} \\ \left[(\overline{\rho}_i - \rho_i)\overline{\nu}_i - \overline{s}\right]_{i=1,\ldots,m} \end{bmatrix}.$$

In order to perform an iteration of the Newton method on this system, we compute the Jacobian of the residual function:

$$J(\tilde{\boldsymbol{r}}) := \nabla_{(\boldsymbol{u},\tilde{a},\boldsymbol{\rho},\underline{\boldsymbol{\nu}},\overline{\boldsymbol{\nu}})}\, \tilde{\boldsymbol{r}} = \begin{bmatrix} \mathrm{K}(\boldsymbol{\rho}) & \mathbf{0} & \mathrm{B}(\boldsymbol{u}) & 0 & 0 \\ \mathbf{0}^\top & 0 & \boldsymbol{a}^\top & 0 & 0 \\ \mathrm{B}(\boldsymbol{u})^\top & \boldsymbol{a} & 0 & -\mathrm{I} & -\mathrm{I} \\ 0 & 0 & \underline{\mathrm{N}} & \underline{\mathrm{P}} & 0 \\ 0 & 0 & -\overline{\mathrm{N}} & 0 & \overline{\mathrm{P}} \end{bmatrix}, \tag{3.7}$$

where we have used the notation

$$\underline{\mathrm{N}} = \operatorname{diag}\{\underline{\boldsymbol{\nu}}\}, \qquad \overline{\mathrm{N}} = \operatorname{diag}\{\overline{\boldsymbol{\nu}}\},$$

$$\underline{\mathrm{P}} = \operatorname{diag}\{\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}\}, \quad \overline{\mathrm{P}} = \operatorname{diag}\{\overline{\boldsymbol{\rho}} - \boldsymbol{\rho}\}.$$

The system matrix J in (3.7) is non-symmetric and indefinite. We can however reduce it to one that is symmetric positive definite. We do this in two steps. First, we construct the Schur complement of J with respect to its bottom right 2×2 block, giving us

$$\begin{bmatrix} \mathrm{K}(\boldsymbol{\rho}) & \mathbf{0} & \mathrm{B}(\boldsymbol{u}) \\ \mathbf{0}^\top & 0 & \boldsymbol{a}^\top \\ \mathrm{B}(\boldsymbol{u})^\top & \boldsymbol{a} & -\mathrm{D}_{IP} \end{bmatrix}, \tag{3.8}$$

where $\mathrm{D}_{IP} := \underline{\mathrm{P}}^{-1}\underline{\mathrm{N}} + \overline{\mathrm{P}}^{-1}\overline{\mathrm{N}}$. We then in turn form the Schur complement of (3.8) with respect to its bottom right block. This leaves us with the matrix

$$\mathrm{S}_{IP} := \begin{bmatrix} \mathrm{K}(\boldsymbol{\rho}) & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \begin{bmatrix} \mathrm{B}(\boldsymbol{u}) \\ \boldsymbol{a}^\top \end{bmatrix} \mathrm{D}_{IP}^{-1} \begin{bmatrix} \mathrm{B}(\boldsymbol{u})^\top & \boldsymbol{a} \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{3.9}$$

This matrix is symmetric and positive definite as long as $\boldsymbol{\rho}$ is strictly feasible and $\underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}} > 0$, both of which is guaranteed by the IP method. Indeed, both of the matrix summands are positive semidefinite and their sum is positive definite as long as their null spaces do not overlap. As $\mathrm{K}(\boldsymbol{\rho}) \succ 0$ for all strictly feasible $\boldsymbol{\rho}$, the null space of the left matrix summand is the $(n+1)$th canonical unit vector. It is easy to check that this vector does not lie in the null space of the right matrix summand.

In each iteration of the IP method, we perform a single iteration of the Newton method for the nonlinear system $\tilde{\boldsymbol{r}}(\boldsymbol{u}, \tilde{\alpha}, \boldsymbol{\rho}, \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}) = \mathbf{0}$. Rather than solving the indefinite linear system $J\,\Delta(\boldsymbol{u}, \tilde{\alpha}, \boldsymbol{\rho}, \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}) = -\tilde{\boldsymbol{r}}$, we solve the equivalent system

$$\mathrm{S}_{IP} \begin{bmatrix} \Delta \boldsymbol{u} \\ \Delta \tilde{\alpha} \end{bmatrix} = \boldsymbol{r}_{IP}, \tag{3.10}$$

where the right-hand side vector $\boldsymbol{r}_{IP}$ follows from the reduction of the system described further above as

$$\boldsymbol{r}_{IP} = -\begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{bmatrix} - \begin{bmatrix} \mathrm{B}(\boldsymbol{u}) \\ \boldsymbol{a}^\mathsf{T} \end{bmatrix} \mathrm{D}_{IP}^{-1} \left(\boldsymbol{r}_3 + \underline{\mathrm{P}}^{-1} \boldsymbol{r}_4 - \overline{\mathrm{P}}^{-1} \boldsymbol{r}_5 \right).$$

From the solution of (3.10), we can reconstruct the increment for $\boldsymbol{\rho}$ using the identity

$$\Delta \boldsymbol{\rho} = -\mathrm{D}_{IP}^{-1} \left(\boldsymbol{r}_3 + \underline{\mathrm{P}}^{-1} \boldsymbol{r}_4 - \overline{\mathrm{P}}^{-1} \boldsymbol{r}_5 - \mathrm{B}(\boldsymbol{u})^\mathsf{T} \Delta \boldsymbol{u} - \Delta \tilde{\alpha}\, \boldsymbol{a} \right). \tag{3.11}$$

The increments for the Lagrange multipliers $\underline{\boldsymbol{\nu}}$ and $\overline{\boldsymbol{\nu}}$ are computed based on the stable reduction proposed in [53], with a slight adjustment to account for the upper bound constraints not present in that paper. The multipliers are updated by the formulas below, where the second formula uses the result of the first.

$$\Delta \overline{\boldsymbol{\nu}} = \frac{1}{\overline{\boldsymbol{\rho}} - \underline{\boldsymbol{\rho}}} \left(\underline{\mathrm{P}} \left(\mathrm{B}(\boldsymbol{u})^\mathsf{T} \Delta \boldsymbol{u} + \Delta \tilde{\alpha}\, \boldsymbol{a} \right) - (\underline{\mathrm{N}} - \overline{\mathrm{N}}) \boldsymbol{\rho} - (\boldsymbol{r}_4 + \boldsymbol{r}_5 - \underline{\mathrm{P}} \boldsymbol{r}_3) \right), \tag{3.12}$$

$$\Delta \underline{\boldsymbol{\nu}} = \Delta \overline{\boldsymbol{\nu}} - \mathrm{B}(\boldsymbol{u})^\mathsf{T} \Delta \boldsymbol{u} - \Delta \tilde{\alpha}\, \boldsymbol{a} - \boldsymbol{r}_3. \tag{3.13}$$

Details about how we solve system (3.10) are given in Section 3.1.4.

Once the increments have been obtained from (3.10–3.13), we need to determine an appropriate step length. Our algorithm employs a long step strategy [132] in that it restricts the step length mainly to guarantee strict feasibility of the next iterate. We do not use the same step length for all increment components. Rather,

$\Delta\boldsymbol{\rho}$ and $\Delta\boldsymbol{u}$ use the same step length, the step length for $\Delta\tilde{\alpha}$ is always equal to 1 and different step lengths are calculated for both $\Delta\underline{\boldsymbol{\nu}}$ and $\Delta\overline{\boldsymbol{\nu}}$. For details, see Algorithm 4. This step length strategy proved to be the most effective in numerical experiments.

After each IP iteration, the barrier parameters are updated adaptively. For this, we compute the duality measure for the lower and upper bound constraint,

$$\frac{\underline{\boldsymbol{\nu}}^{\mathsf{T}}(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}})}{m} \quad \text{and} \quad \frac{\overline{\boldsymbol{\nu}}^{\mathsf{T}}(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho})}{m},$$

respectively. We then scale these measures by constants $0 < \sigma_{\underline{s}} < 1$ and $0 < \sigma_{\overline{s}} < 1$ to get a new value for $\underline{s}$ and $\overline{s}$. At this point, one unconventional feature[1] of our algorithm should be highlighted. The new values for $\underline{s}$ and $\overline{s}$ are not used to construct the right hand side term for the next iteration, but rather for the iteration after that. We found that this “iteration shift”, peculiar though it might seem, makes the algorithm significantly more efficient. Presumably, a similar behaviour could be obtained by other means, such as choosing larger values for $\sigma_{\underline{s}}$ and $\sigma_{\overline{s}}$, or even determining them adaptively. For the sake of simplicity, we decided to stick with this slightly unorthodox approach, acknowledging that it lacks theoretical justification.

Finally, we require a stopping criterion for the algorithm. We use the duality gap $\delta(\boldsymbol{u}, \alpha) = \delta(\boldsymbol{u}, -\tilde{\alpha})$ as a measure of optimality, scaled by the current objective function value $\frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u}$. The optimization terminates once the scaled duality gap is smaller than a set threshold while also being (nearly) positive, as a negative duality gap points to infeasibility. Algorithm 4 sums up our IP method. The parameters that we used in our experiments are $\varepsilon_{\mathrm{IP}} = 10^{-5}$, $\sigma_{\underline{s}} = \sigma_{\overline{s}} = 0.2$; we chose the initial values $\boldsymbol{u} = \mathbf{0}$, $\tilde{\alpha} = -1$, $\rho_i = V/(\ {}_i a_i)$ for all $i = 1, \dots, m$ and $\underline{\boldsymbol{\nu}} = \overline{\boldsymbol{\nu}} = \boldsymbol{a}$; the barrier parameter values in the very first iteration are $\underline{s} = \overline{s} = 10^{-2}$.

[1]A bug turned feature, actually.

Algorithm 4 Primal-dual IP method for the VTS problem

Let $\varepsilon_{\mathrm{IP}} > 0$ and $0 < \sigma_{\underline{s}}, \sigma_{\overline{s}} < 1$ be given. Choose initial vectors $(\boldsymbol{u}, \boldsymbol{\rho})$ and $(\tilde{\alpha}, \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}})$. Set barrier parameter update values to $\underline{s}^{+} = \sigma_{\underline{s}} \cdot \underline{\boldsymbol{\nu}}^{\mathsf{T}}(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}})/m$ and $\overline{s}^{+} = \sigma_{\overline{s}} \cdot \overline{\boldsymbol{\nu}}^{\mathsf{T}}(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho})/m$.

```
1: repeat
2:     Solve system (3.10) to obtain $(\Delta\boldsymbol{u}, \Delta\tilde{\alpha})$
3:     Reconstruct $(\Delta\boldsymbol{\rho}, \Delta\overline{\boldsymbol{\nu}}, \Delta\underline{\boldsymbol{\nu}})$ using (3.11–3.13)
4:     Compute step lengths
           $\kappa_{\boldsymbol{u}} = \kappa_{\boldsymbol{\rho}} = \min\left\{0.9 \cdot \min_{\Delta\rho_i<0} \frac{\underline{\rho}_i - \rho_i}{\Delta\rho_i},\ 0.9 \cdot \min_{\Delta\rho_i>0} \frac{\overline{\rho}_i - \rho_i}{\Delta\rho_i},\ 1\right\},$
           $\kappa_{\underline{\boldsymbol{\nu}}} = \min\left\{0.9 \cdot \min_{\Delta\underline{\nu}_i<0} \frac{-\underline{\nu}_i}{\Delta\underline{\nu}_i},\ 1\right\},\quad \kappa_{\overline{\boldsymbol{\nu}}} = \min\left\{0.9 \cdot \min_{\Delta\overline{\nu}_i<0} \frac{-\overline{\nu}_i}{\Delta\overline{\nu}_i},\ 1\right\},\quad \kappa_{\tilde{\alpha}} = 1$
5:     Update all variables
           $\boldsymbol{u} \leftarrow \boldsymbol{u} + \kappa_{\boldsymbol{u}}\Delta\boldsymbol{u},\quad \tilde{\alpha} \leftarrow \tilde{\alpha} + \kappa_{\tilde{\alpha}}\Delta\tilde{\alpha},\quad \boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \kappa_{\boldsymbol{\rho}}\Delta\boldsymbol{\rho},$
           $\underline{\boldsymbol{\nu}} \leftarrow \underline{\boldsymbol{\nu}} + \kappa_{\underline{\boldsymbol{\nu}}}\Delta\underline{\boldsymbol{\nu}},\quad \overline{\boldsymbol{\nu}} \leftarrow \overline{\boldsymbol{\nu}} + \kappa_{\overline{\boldsymbol{\nu}}}\Delta\overline{\boldsymbol{\nu}}$
6:     Update barrier parameters
           $\underline{s} = \underline{s}^{+},\quad \overline{s} = \overline{s}^{+}$
7:     Determine barrier parameters for shifted update
           $\underline{s}^{+} = \sigma_{\underline{s}} \cdot \frac{\underline{\boldsymbol{\nu}}^{\mathsf{T}}(\boldsymbol{\rho} - \underline{\boldsymbol{\rho}})}{m},\quad \overline{s}^{+} = \sigma_{\overline{s}} \cdot \frac{\overline{\boldsymbol{\nu}}^{\mathsf{T}}(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho})}{m}$
8: until $\varepsilon_{\mathrm{IP}} > \delta(\boldsymbol{u}, -\tilde{\alpha})/(\frac{1}{2}\boldsymbol{f}^{\mathsf{T}}\boldsymbol{u}) > -0.1\varepsilon_{\mathrm{IP}}$
```

Equivalence of primal and dual KKT systems

As mentioned earlier, it does not make any substantial difference whether we apply the IP method to the primal or dual VTS problem. The short explanation for this is that we use a primal-dual IP method, which solves the primal and dual problem simultaneously. In our case, however, the two formulations (2.48) and (2.54) do not conform to the strict definition of primal and dual problems, see Remark 2.3.9. We will therefore devote a few pages to showing that the KKT system of (2.54) is equivalent to that of (2.48), the consequence of this being that we can apply the IP approach to either and arrive at the same algorithm.

We start from a slight reformulation of the dual problem (2.54). We introduce slack variables $z_1, \dots, z_m \geq 0$ in order to write (2.54b) as a set of equality constraints:

$$\min_{\boldsymbol{u}\in\mathbb{R}^n,\ \alpha\in\mathbb{R},\ \underline{\boldsymbol{\nu}},\overline{\boldsymbol{\nu}},\boldsymbol{z}\in\mathbb{R}^m} \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} - \underline{\boldsymbol{\rho}}^\mathsf{T}\underline{\boldsymbol{\nu}} + \overline{\boldsymbol{\rho}}^\mathsf{T}\overline{\boldsymbol{\nu}} \tag{3.14}$$

$$\text{s.t.}\quad \frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} - \alpha a_i + \underline{\nu}_i - \overline{\nu}_i + z_i = 0, \quad i = 1, \dots, m, \tag{3.15}$$

$$\underline{\nu}_i \geq 0, \quad i = 1, \dots, m, \tag{3.16}$$

$$\overline{\nu}_i \geq 0, \quad i = 1, \dots, m, \tag{3.17}$$

$$z_i \geq 0, \quad i = 1, \dots, m. \tag{3.18}$$

We introduce the multipliers $\boldsymbol{\rho}, \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}, \xi \in \mathbb{R}^m$ for the respective sets of constraints, so that we get the Lagrange function

$$\begin{aligned} L(\boldsymbol{u}, \alpha, \underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}, \boldsymbol{z}; \boldsymbol{\rho}, \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}, \xi) = {} & \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} - \underline{\boldsymbol{\rho}}^\mathsf{T}\underline{\boldsymbol{\nu}} + \overline{\boldsymbol{\rho}}^\mathsf{T}\overline{\boldsymbol{\nu}} \\ & + \sum_{i=1}^m \rho_i \left(\frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} - \alpha a_i + \underline{\nu}_i - \overline{\nu}_i + z_i\right) - \sum_{i=1}^m \underline{\mu}_i\underline{\nu}_i - \sum_{i=1}^m \overline{\mu}_i\overline{\nu}_i - \sum_{i=1}^m \xi_i z_i, \end{aligned}$$

from which we can derive the KKT conditions:

$$\nabla_{\boldsymbol{u}} L = \quad -\boldsymbol{f} + \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \mathbf{0}\,, \tag{3.19a}$$

$$\nabla_{a} L = \quad V - \boldsymbol{\rho}^{\mathsf{T}}\boldsymbol{a} = 0\,, \tag{3.19b}$$

$$\nabla_{\underline{\boldsymbol{\nu}}} L = \quad -\underline{\boldsymbol{\rho}} + \boldsymbol{\rho} - \underline{\boldsymbol{\mu}} = 0\,, \tag{3.19c}$$

$$\nabla_{\overline{\boldsymbol{\nu}}} L = \quad \overline{\boldsymbol{\rho}} - \boldsymbol{\rho} - \overline{\boldsymbol{\mu}} = 0, \tag{3.19d}$$

$$\nabla_{\boldsymbol{z}} L = \quad \boldsymbol{\rho} - \boldsymbol{\xi} = 0, \tag{3.19e}$$

$$\underline{\mu}_i \underline{\nu}_i = 0,\, i = 1, \ldots, m, \tag{3.19f}$$

$$\overline{\mu}_i \overline{\nu}_i = 0,\, i = 1, \ldots, m\,, \tag{3.19g}$$

$$\xi_i z_i = 0,\, i = 1, \ldots, m, \tag{3.19h}$$

$$\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{B}(\boldsymbol{u}) - \alpha\,\boldsymbol{a} + \underline{\boldsymbol{\nu}} - \overline{\boldsymbol{\nu}} + \boldsymbol{z} = \mathbf{0}, \tag{3.19i}$$

$$\underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}, \boldsymbol{z}, \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}, \boldsymbol{\xi} \geq \mathbf{0}\,. \tag{3.19j}$$

First, we use the simple linear equations (3.19c–3.19e) to eliminate the multipliers $\underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}$ and ξ. Together with (3.19j), we thus get $\boldsymbol{\rho} - \underline{\boldsymbol{\rho}} \geq \mathbf{0}$ and $\overline{\boldsymbol{\rho}} - \boldsymbol{\rho} \geq \mathbf{0}$. The complementarity conditions (3.19f–3.19h) become

$$(\rho_i - \underline{\rho_i})\underline{\nu_i} = 0\,,\; i = 1, \ldots, m, \tag{3.20a}$$

$$(\overline{\rho}_i - \rho_i)\,\overline{\nu}_i = 0,\, i = 1, \ldots, m\,, \tag{3.20b}$$

$$\rho_i z_i = 0,\, i = 1, \ldots, m. \tag{3.20c}$$

Second, we eliminate the slack variables $\boldsymbol{z}$ along with the set of complementarity conditions (3.20c). We can do this by including $\boldsymbol{z}$ in $\underline{\boldsymbol{\nu}}$. To be more precise, we introduce $\underline{\tilde{\boldsymbol{\nu}}} := \underline{\boldsymbol{\nu}} + \boldsymbol{z} \geq \mathbf{0}$ to replace both $\underline{\boldsymbol{\nu}}$ and $\boldsymbol{z}$. If $\underline{\rho_i} > 0$ for a particular i, then the corresponding z_i has to be zero due to (3.20c). Knowing this, we can ignore $\rho_i z_i = 0$. If, on the other hand, $\underline{\rho_i} = 0$, then we have $\rho_i z_i = (\rho_i - \underline{\rho_i})z_i = 0$. This, together with (3.20a), is equivalent to $(\rho_i - \underline{\rho_i})\,\underline{\tilde{\nu}_i} = 0$, since $z_i, \underline{\nu_i} \geq 0$. In either case, we can safely replace (3.20a) and (3.20c) by $(\rho_i - \underline{\rho_i})\underline{\tilde{\nu}}_i = 0$. In summary, the

KKT conditions (3.19) for the dual VTS problem are equivalent to

$$
\begin{aligned}
-\boldsymbol{f} + \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} &= \mathbf{0}\,,\\
V - \boldsymbol{\rho}^{\mathsf{T}}\boldsymbol{a} &= 0\,,\\
(\rho_i - \underline{\rho}_i)\,\underline{\nu}_i &= 0\,, i = 1,\ldots,m\,,\\
(\overline{\rho}_i - \rho_i)\,\overline{\nu}_i &= 0\,, i = 1,\ldots,m\,,\\
\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{B}(\boldsymbol{u}) - \alpha\,\boldsymbol{a} + \underline{\boldsymbol{\nu}} - \overline{\boldsymbol{\nu}} &= \mathbf{0}\,,\\
\underline{\boldsymbol{\nu}}, \overline{\boldsymbol{\nu}}, (\boldsymbol{\rho} - \underline{\boldsymbol{\rho}}), (\overline{\boldsymbol{\rho}} - \boldsymbol{\rho}) &\geq \mathbf{0}\,,
\end{aligned}
$$

which is the same as system (3.1) which we started out with when deriving Algorithm 4.

3.1.2 Penalty-Barrier Multiplier Method for the Dual VTS Problem

The IP method presented above works very well for medium-sized problems. When the number of elements in a mesh increases to the order of 10^6 and above, however, it struggles to converge within a reasonable number of iterations – or at all. Rather than conduct an exhaustive study of the many parameters of the IP method one can adjust in the hope of improving its performance for a particular set of problems, we propose using an alternative algorithm altogether. The PBM method reviewed in Section 2.1.4 has been applied successfully to both (truss) topology design problems [79] as well as large-scale problems, for example in the optimization software Pennon [76, 77], and we will apply it to the VTS compliance minimization problem.

As mentioned at the beginning of Section 3.1, we apply the PBM method to the dual VTS problem. If we were to apply it to the primal problem (2.48), problems would arise due to the fact that it does not guarantee strict feasibility of the densities, in contrast to the IP method. As a result, the stiffness matrix $\mathrm{K}(\boldsymbol{\rho})$ might become indefinite. Not only would this require the use of a different and much more complicated solver setup, but it could also lead to spurious solutions since we would effectively model a mechanical structure with negative elasticity coefficients in areas of negative density. We can circumvent these issues by instead applying the PBM method to the dual problem, in which the densities play the role of Lagrange multipliers. The updating formula (2.24b) guarantees that these are always greater than zero. While some might converge to zero – at least if we set the lower bounds $\underline{\boldsymbol{\rho}} = \mathbf{0}$ – in practice, we did not encounter any serious problems related to $\mathrm{K}(\boldsymbol{\rho})$ becoming ill-conditioned, (although a regularization of $\mathrm{K}(\boldsymbol{\rho})$ improves the algorithm somewhat, see Section 3.1.4). We therefore derive our PBM method for the simplified dual VTS problem (2.60) with zero lower bounds.

We begin with the definition of the augmented Lagrangian for problem (2.60):

$$\begin{aligned} \mathrm{L}(\boldsymbol{u}, \alpha, \boldsymbol{v}; \boldsymbol{\rho}, \boldsymbol{\mu}) = {} & \alpha V - \boldsymbol{f}^\mathsf{T}\boldsymbol{u} + \bar{\boldsymbol{\rho}}^\mathsf{T}\boldsymbol{v} \\ & + \sum_{i=1}^{m} \rho_i p \phi\left(\frac{1}{p}\left(\frac{1}{2}\boldsymbol{u}^\mathsf{T}\mathrm{K}_i\boldsymbol{u} - \alpha a_i - v_i\right)\right) \\ & + \sum_{i=1}^{m} \mu_i p \phi\left(\frac{-v_i}{p}\right), \end{aligned} \tag{3.21}$$

with Lagrangian multipliers $\boldsymbol{\rho} > \mathbf{0}$ and $\boldsymbol{\mu} > \mathbf{0}$, penalty parameter $p > 0$, and the logarithmic-quadratic penalty-barrier function ϕ defined in (2.27).

Recall the general form of the gradient (2.25) and Hessian (2.26) of the augmented Lagrangian. They differ from the gradient and Hessian, respectively, of the regular Lagrangian in that they feature penalty parameters and derivatives of the penalty function ϕ. These extra terms can be seen as scaling factors for the Lagrange multipliers. To highlight the similarity between the regular and augmented Lagrangian, but also in order to condense the notation, we introduce the “scaled”

Lagrange multipliers

$$\rho_i := \rho_i \phi\left(\frac{1}{p}\left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i - v_i\right)\right), \quad i = 1, \ldots, m,$$

$$\rho_i := \frac{\rho_i}{p} \phi\left(\frac{1}{p}\left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha a_i - v_i\right)\right), \quad i = 1, \ldots, m,$$

$$\mu_i := \mu_i \phi\left(\frac{-v_i}{p}\right), \quad i = 1, \ldots, m,$$

$$\mu_i := \frac{\mu_i}{p} \phi\left(\frac{-v_i}{p}\right), \quad i = 1, \ldots, m.$$

Furthermore, we denote the diagonal matrices constructed from the vectors $\boldsymbol{\rho}$, $\boldsymbol{\mu}$, and $\boldsymbol{a}$ respectively, by

$$\mathrm{P} := \mathrm{diag}\{\boldsymbol{\rho}\}, \quad \mathrm{M} := \mathrm{diag}\{\boldsymbol{\mu}\}, \quad \mathrm{A} := \mathrm{diag}\{\boldsymbol{a}\}.$$

We now compute the gradient of the augmented Lagrangian and, using the above shorthand, write it as

$$\nabla_{\boldsymbol{u}} \mathrm{L} = -\boldsymbol{f} + \sum_{i=1}^{m} \rho_i \mathrm{K}_i\boldsymbol{u} = -\boldsymbol{f} + \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} =: \boldsymbol{r}_1 \tag{3.22a}$$

$$\nabla_{\alpha} \mathrm{L} = V - \sum_{i=1}^{m} \rho_i a_i =: \boldsymbol{r}_2 \tag{3.22b}$$

$$\nabla_{\boldsymbol{v}} \mathrm{L} = \bar{\boldsymbol{\rho}} - \boldsymbol{\rho} - \boldsymbol{\mu} =: \boldsymbol{r}_3. \tag{3.22c}$$

For later reference, note that we will often write $\nabla_{(\boldsymbol{u},\alpha,\boldsymbol{v})}\mathrm{L} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ simply as $\nabla\mathrm{L}$. Finally, we determine the Hessian of L in order to set up the Newton system, which takes the form

$$\begin{bmatrix} \mathrm{K}(\boldsymbol{\rho}) + \mathrm{B}(\boldsymbol{u})\mathrm{P}\,\mathrm{B}(\boldsymbol{u})^{\mathsf{T}} & -\mathrm{K}(\mathrm{A}\boldsymbol{\rho})\boldsymbol{u} & -\mathrm{B}(\boldsymbol{u})\mathrm{P} \\ -\boldsymbol{u}^{\mathsf{T}}\mathrm{K}(\mathrm{A}\boldsymbol{\rho}) & \sum_{i=1}^{m} a_i^2\rho_i & \boldsymbol{\rho}^{\mathsf{T}}\mathrm{A} \\ -\mathrm{P}\,\mathrm{B}(\boldsymbol{u})^{\mathsf{T}} & \mathrm{A}\boldsymbol{\rho} & \mathrm{D}_{PBM} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u} \\ \Delta\alpha \\ \Delta\boldsymbol{v} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{bmatrix}, \tag{3.23}$$

where $\mathrm{D}_{PBM} = \mathrm{P} + \mathrm{M}$ and $\mathrm{B}(\boldsymbol{u}) = [\mathrm{K}_1\boldsymbol{u}, \ldots, \mathrm{K}_m\boldsymbol{u}]$. The above matrix is symmetric and, because it is the Hessian of the augmented Lagrangian, which is strictly convex, it is also positive definite. We could therefore solve the system in this form both by the MINRES and CG method. However, we perform a reduction similar to that of system (3.8) in the IP method. The advantage of this approach will be discussed in Section 3.1.4.

Since ϕ is strictly convex, $\phi > 0$ and therefore the diagonal matrix D_{PBM} is positive definite and (easily) invertible. We can thus eliminate $\Delta \boldsymbol{v}$ and the third line of (3.23) and obtain the reduced system

$$S_{PBM}\begin{pmatrix}\Delta \boldsymbol{u}\\ \Delta\alpha\end{pmatrix} = \boldsymbol{r}_{PBM} \tag{3.24}$$

that features the Schur complement of D_{PBM} in the Newton system matrix,

$$\begin{aligned} S_{PBM} := &\begin{pmatrix} K(\boldsymbol{\rho}) + B(\boldsymbol{u})P\,B(\boldsymbol{u})^\top & -K(A\boldsymbol{\rho})\boldsymbol{u}\\ -\boldsymbol{u}^\top K(A\boldsymbol{\rho}) & \sum_{i=1}^{m} a_i^2\rho_i \end{pmatrix}\\ &- \begin{pmatrix} -B(\boldsymbol{u})P\\ \boldsymbol{\rho}^\top A\end{pmatrix} D_{PBM}^{-1}\begin{pmatrix} -P\,B(\boldsymbol{u})^\top & A\boldsymbol{\rho}\end{pmatrix}, \end{aligned} \tag{3.25}$$

and the right-hand side vector

$$\boldsymbol{r}_{PBM} = -\begin{pmatrix}\boldsymbol{r}_1\\ \boldsymbol{r}_2\end{pmatrix} + \begin{pmatrix} -B(\boldsymbol{u})P\\ \boldsymbol{\rho}^\top A\end{pmatrix} D_{PBM}^{-1}\,\boldsymbol{r}_3. \tag{3.26}$$

S_{PBM} is positive definite, since it is the Schur complement of a symmetric positive definite matrix and D_{PBM} is non-singular [135, (Theorem 1.12)]. After solving (3.26), we can obtain $\Delta\boldsymbol{v}$ from

$$\Delta\boldsymbol{v} = -D_{PBM}^{-1}\left(\boldsymbol{r}_3 + \begin{pmatrix} -P\,B(\boldsymbol{u})^\top & A\boldsymbol{\rho}\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{u}\\ \Delta\alpha\end{pmatrix}\right). \tag{3.27}$$

To determine a step-size, we perform backtracking and apply the Armijo condition, see Line 4 in Algorithm 5, after which we update the variables $(\boldsymbol{u}, \alpha, \boldsymbol{v})$. This procedure comprises one Newton iteration and we repeat it until $\|\nabla L\|_\infty < \varepsilon_{\mathrm{NWT}}$

for a threshold $\varepsilon_{\text{NWT}} > 0$, at which point we have completed the first step of the PBM iteration. Next, we update the Lagrange multipliers according to (2.24b). Additionally, we impose the safeguard rule used in [17]: in order to avoid extreme changes in the multiplier values and stabilize the algorithm, we bound the multiplier update above and below by a factor $\beta_{LM} \in (0, 1)$, see Line 8 in Algorithm 5. The final step of the PBM iteration is the update of the penalty parameter. For this, we first introduce the *merit function*

$$\theta(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu}) := \max \left\{ \|\nabla L\|_\infty, \frac{\|(\boldsymbol{g})_+\|_1}{m}, \frac{\|(-\boldsymbol{v})_+\|_1}{m}, \right. \\ \left. -\boldsymbol{\rho}, -\boldsymbol{\mu}, \frac{|\boldsymbol{\rho}|^\top|\boldsymbol{g}|}{m} + \frac{|\boldsymbol{\mu}|^\top|\boldsymbol{v}|}{m} \right\}, \tag{3.28}$$

where the vector $\boldsymbol{g}$ denotes the first set of inequality constraints, i.e.

$$g_i = \frac{1}{2}\boldsymbol{u}^\top K_i \boldsymbol{u} - \alpha a_i - v_i, \quad i = 1, \dots, m,$$

and $(\cdot)_+ := \max\{\cdot, 0\}$. The merit function θ serves as an optimality measure. Clearly, $\theta \geq 0$ holds everywhere and if $\theta(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu}) = 0$ then $(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu})$ is an optimal solution. We will use θ to choose the value of the penalty parameter adaptively based on how close to optimality we currently are. We achieve this by evaluating θ at the updated solution and setting $p = \sigma\theta$ for some constant factor $\sigma > 0$. However, if p changes too much, the current values of $(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu})$ might be a bad initial guess for the Newton method in the next PBM iteration. (This consideration is similar to the path following paradigm of IP methods mentioned in Section 2.1.3). Therefore, we impose the condition that p is not reduced by more than a factor $\underline{\beta}_p$. At the same time, we want to avoid stagnation of the method at a point that is too far away from the solution to guarantee that θ is decreased by an acceptable factor in each iteration. Hence, we also cap the next p by a value that is $\overline{\beta}_p$ times that of the current penalty parameter, where $\overline{\beta}_p \in (\underline{\beta}_p, 1)$.

The merit function also has two other purposes in our algorithm. First, we change the tolerance ε_{NWT} for the Newton method based on the current value of θ, similarly to the penalty parameter. Second, we use θ in the stopping criterion for the PBM method. While the duality gap $\delta(\boldsymbol{u}, \alpha)$ given in (2.62) is a very useful optimality measure in practice, see Section 3.2.1, we simultaneously keep track of θ. This is because it quantifies not just the distance to optimality but also the feasibility of the solution and we need to ensure that the PBM algorithm does not terminate prematurely due to spuriously low duality gap values attained at strongly infeasible points. Although we only observed this pathological behaviour in a very few cases, satisfying the extra stopping criterion does not usually require a lot more iterations. We include it just to be on the safe side and because we do not have to compromise on efficiency for it.

Algorithm 5 gives the details of the PBM algorithm for the dual VTS problem. For the results in Section 3.2, we used the parameters $\beta_{LM} = 0.01$, $\sigma = 0.5$, $\beta_p \equiv 0.3$, $\overline{\beta}_p = 0.9$, and $\gamma = 0.01$. The Newton tolerance starts at $\varepsilon_{\text{NWT}} = 1$ and is bounded below by $\varepsilon_{\text{NWT}}^{\min} = 10^{-5}$. The stopping threshold for the PBM method is $\varepsilon_{\text{PBM}} = 10^{-5}$. The initial guesses for the variables are $\boldsymbol{u} = \mathbf{0}$, $\alpha = 1$, $\boldsymbol{v} = \boldsymbol{a}$. For the Lagrange multipliers, they are $\rho_i = V/(\quad_i a_i)$ for all $i = 1, \ldots, m$, and $\boldsymbol{\mu} = \overline{\boldsymbol{\rho}} - \boldsymbol{\rho}$.

The adaptive updating scheme for the penalty parameter p in our implementation is taken from [101, 102] – an improved version of the one proposed by Griva and Polyak in [60]. It is one of the main ingredients of a class of nonlinear rescaling methods with local superlinear convergence introduced in those references. The other ingredient, however, a primal-dual step that updates the primal variables and Lagrange multipliers simultaneously, did not provide any improvement when we tested it on our examples. This might be due to the fact that we stop our optimization algorithm relatively early, with $\varepsilon_{\text{PBM}} = 10^{-5}$, presumably before the neighbourhood of superlinear convergence is reached. In all of the aforementioned references, the theoretical convergence results assume that strict complementarity holds at the solution, as well as the LICQ and the sufficient second order optimality

Algorithm 5 PBM method for the dual VTS problem

```
Let $0 < \beta_{LM} < 1$, $\sigma > 0$, $0 < \underline{\beta}_p < \overline{\beta}_p < 1$, $0 < \gamma < 1$, $\varepsilon_{\mathrm{PBM}} > 0$, $\varepsilon_{\mathrm{NWT}} > 0$ and
$\varepsilon_{\mathrm{NWT}}^{\min} > 0$ be given. Choose initial vectors $(\boldsymbol{u}, \alpha, \boldsymbol{v})$ and $(\boldsymbol{\rho}, \boldsymbol{\mu})$. Set $p = 1$.
 1: repeat
 2:     repeat                                                                ▷ Step 1
 3:         Solve (3.24) for $(\Delta\boldsymbol{u}, \Delta\alpha)$ and compute $\Delta\boldsymbol{v}$ by (3.27)
 4:         Find the largest $\kappa \in \{1, 0.75, 0.75^2, \dots\}$ such that
                $\mathrm{L}(\boldsymbol{u} + \kappa\Delta\boldsymbol{u}, \alpha + \kappa\Delta\alpha, \boldsymbol{v} + \kappa\Delta\boldsymbol{v}; \boldsymbol{\rho}, \boldsymbol{\mu}) - \mathrm{L}(\boldsymbol{u}, \alpha, \boldsymbol{v}; \boldsymbol{\rho}, \boldsymbol{\mu})$
                $\leq \kappa\gamma\nabla\mathrm{L}(\boldsymbol{u}, \alpha, \boldsymbol{v}; \boldsymbol{\rho}, \boldsymbol{\mu})^{\mathsf{T}}(\Delta\boldsymbol{u}, \Delta\alpha, \Delta\boldsymbol{v})$
 5:         Update the variables
                $(\boldsymbol{u}, \alpha, \boldsymbol{v}) \leftarrow (\boldsymbol{u} + \kappa\Delta\boldsymbol{u}, \alpha + \kappa\Delta\alpha, \boldsymbol{v} + \kappa\Delta\boldsymbol{v})$
 6:     until $\|\nabla\mathrm{L}(\boldsymbol{u}, \alpha, \boldsymbol{v})\|_\infty < \varepsilon_{\mathrm{NWT}}$
 7:     Update the multipliers                                                ▷ Step 2
            $\rho_i^+ = \rho_i\phi\left(\frac{1}{p}\left(\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\mathrm{K}_i\boldsymbol{u} - \alpha + v_i - v_i\right)\right), \quad i = 1, \dots, m,$
            $\mu_i^+ = \mu_i\phi\left(\frac{-v_i}{p}\right), \quad i = 1, \dots, m$
 8:     Apply safeguard rule
            $\rho_i \leftarrow \min\{\max\{\beta_{LM}\rho_i, \rho_i^+\}, \frac{\rho_i}{\beta_{LM}}\}, \quad i = 1, \dots, m,$
            $\mu_i \leftarrow \min\{\max\{\beta_{LM}\mu_i, \mu_i^+\}, \frac{\mu_i}{\beta_{LM}}\}, \quad i = 1, \dots, m$
 9:     Update the penalty parameter                                          ▷ Step 3
            $p \leftarrow \max\{\min\{\sigma\,\theta(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu}), \overline{\beta}_p p\}, \underline{\beta}_p p\}$
10:     Update the Newton tolerance
            $\varepsilon_{\mathrm{NWT}} \leftarrow \max\{\min\{\theta(\boldsymbol{u}, \alpha, \boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\mu}), \varepsilon_{\mathrm{NWT}}\}, \varepsilon_{\mathrm{NWT}}^{\min}\}$
11: until $\delta(\boldsymbol{u}, \alpha)/(\alpha V - \boldsymbol{f}^{\mathsf{T}}\boldsymbol{u} + \overline{\boldsymbol{\rho}}^{\mathsf{T}}\boldsymbol{v}) < \varepsilon_{\mathrm{PBM}}$ and $\theta < 10\varepsilon_{\mathrm{PBM}}$
```

conditions. While we cannot guarantee this for our problem, the PBM method described in this section still proved to be both reliable and efficient in all of our numerical experiments.

3.1.3 Including Unilateral Contact Constraints

In order to handle problems with unilateral contact constraints within the presented PBM framework, we merely need to consider the additional set of inequality constraints in (2.65). This leads to an extra term in the augmented Lagrangian. Since the contact constraints, and consequently this extra term, only depend on ***u***, not many changes are necessary in the resulting Newton system in order to adapt the PBM method for unilateral contact constraints. In the following, we briefly summarize these changes.

Firstly, the augmented Lagrangian (3.21) becomes

$$
\begin{aligned}
\mathrm{L}(\boldsymbol{u},\alpha,\boldsymbol{v};\boldsymbol{p},\boldsymbol{\mu}) = {} & \alpha V - \boldsymbol{f}^{\mathsf T}\boldsymbol{u} + \bar{\boldsymbol{\rho}}^{\mathsf T}\boldsymbol{v} \\
& + \sum_{i=1}^{m} \rho_i p\,\phi\left(\frac{1}{p}\left(\frac{1}{2}\boldsymbol{u}^{\mathsf T}\mathrm{K}_i\boldsymbol{u} - \alpha\, a_i - v_i\right)\right) \\
& + \sum_{i=1}^{m} \mu_i p\,\phi\left(\frac{-v_i}{p}\right) \\
& + \sum_{i\in\chi} \eta_i p\,\phi\left(\frac{\boldsymbol{c}_i^{\mathsf T}\boldsymbol{u} - b_i}{p}\right),
\end{aligned}
$$

where χ is the contact constraint index set defined in (2.41). For the gradient and Hessian, we use a shorthand notation analogous to the one in the previous section, denoting the "scaled" Lagrange multipliers for the contact constraints by

$$
\begin{aligned}
\eta_i &:= \eta_i\phi\left(\frac{\boldsymbol{c}_i\boldsymbol{u} - b_i}{p}\right), \quad i\in\chi, \\
\eta^i &:= \frac{\eta_i}{p}\phi\left(\frac{\boldsymbol{c}_i\boldsymbol{u} - b_i}{p}\right), \quad i\in\chi.
\end{aligned}
$$

With this, the partial derivative of the augmented Lagrangian with respect to ***u***, cf.

(3.22a), is given by

$$\nabla_{\boldsymbol{u}} \mathrm{L} = -\boldsymbol{f} + \mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} + \mathrm{C}\boldsymbol{\eta} \qquad =: \ \boldsymbol{r}_1 \,.$$

Finally, defining the diagonal matrix $\mathrm{H} := \mathrm{diag}\{\eta_i\}$, the upper left block of the Hessian in (3.23) changes to

$$\mathrm{K}(\boldsymbol{\rho}) + \mathrm{B}(\boldsymbol{u})\mathrm{P}\,\mathrm{B}(\boldsymbol{u})^\mathsf{T} + \mathrm{CH}\,\mathrm{C}^\mathsf{T} \,. \tag{3.29}$$

This change obviously carries through to the Schur complement. The adaptation of Algorithm 5 to include the contact stresses $\boldsymbol{\eta}$ in the Lagrange multiplier updating step is straightforward.

3.1.4 Multigrid Preconditioner for MINRES and CG

All of the linear systems arising in any of the optimization algorithms discussed in this thesis are solved either by the MINRES or the CG method. These are in turn preconditioned by a multigrid V-cycle. In Section 2.4.2, we presented the multigrid method as intrinsically connected to a discretized elliptic problem defined on an FE mesh. The transfer operators were motivated by interpolation of DOFs defined on the mesh nodes. However, of all the variables featured in our optimization problem, only the displacements $\boldsymbol{u}$ conform to this particular geometric interpretation. The question of how to incorporate a multigrid method preconditioner in our algorithms comes down to the appropriate choice of prolongator, which is not completely obvious. In the following, we describe our approach in detail. We also address other solver-related questions, such as the stopping criterion and regularization of the system matrices.

For the OC method, applying multigrid is easy enough, since the only linear system that we need to solve is given by the equilibrium equations $\mathrm{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$, which is precisely the kind of system that the multigrid method is intended for. In our IP and PBM method, we start with a linear system that is considerably larger and, in the case of the IP method, not even positive definite. By taking the Schur complement, we arrive at a different system, which turns out to strongly resemble the equilibrium equations in a particular way.

Recall the linear systems (3.10) and (3.24) that we solve in the IP and PBM method, respectively. The unknowns in these systems are the increments $\Delta \boldsymbol{u}$ and $\Delta \alpha$. The former can, just as $\boldsymbol{u}$, be interpreted as a vector of DOFs defined at the mesh nodes, while the second one is a scalar. One can devise a block-wise prolongation operator for these variables by

$$\mathrm{I}_H^h = \begin{pmatrix} (\mathrm{I}_H^h)_{\boldsymbol{u}} & \\ & 1 \end{pmatrix},$$

where $(\mathrm{I}_H^h)_{\boldsymbol{u}}$ is a standard (geometric or algebraic) multigrid prolongation operator for the displacements. The scalar α can be seen as a "global" variable, constant over the entire domain, so that its value is the same on all grid levels. Using the standard operator for the displacements can however not only be motivated by the variables appearing in the linear system, but also by the system's structure.

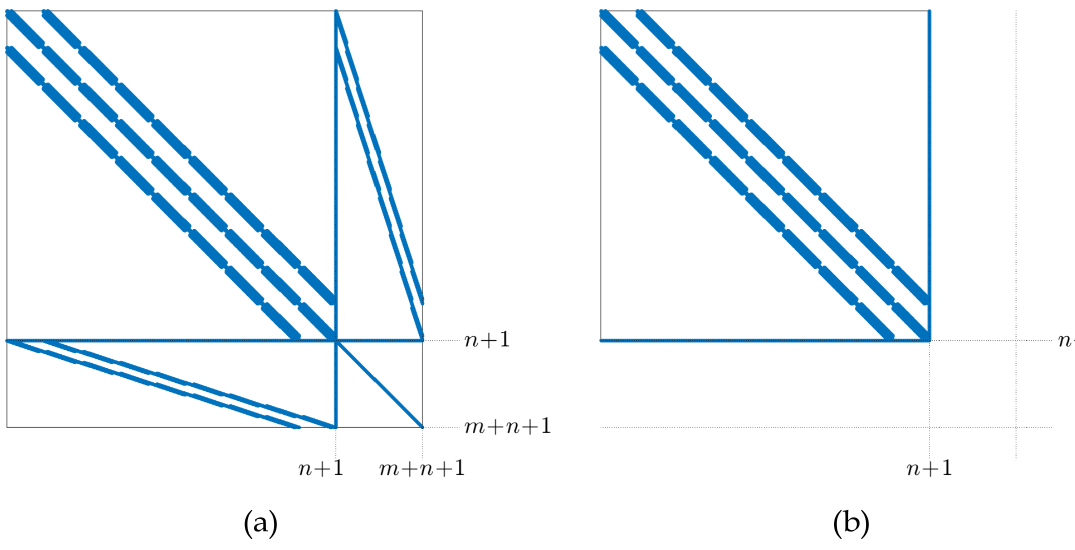
Regard the system matrices for the IP and PBM method, defined in (3.9) and (3.25), respectively. First of all, they are both positive definite. Second of all, their sparsity structure is the same as that of the stiffness matrix $\mathrm{K}(\boldsymbol{\rho}) = \sum_{i=1}^{m} \rho_i \mathrm{K}_i$ for any $\boldsymbol{\rho} > 0$. Indeed, both system matrices can be written in the form

$$\mathrm{S} = \begin{pmatrix} \mathrm{K} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{pmatrix} + \begin{pmatrix} \mathrm{B}(\boldsymbol{u}) \\ \pm \boldsymbol{a}^\top \end{pmatrix} \mathrm{D} \begin{pmatrix} \mathrm{B}(\boldsymbol{u})^\top & \pm \boldsymbol{a} \end{pmatrix},$$

where K is the stiffness matrix $\mathrm{K}(\boldsymbol{\rho})$ for different choices of $\boldsymbol{\rho}$ and D is a diagonal matrix. For S_{IP}, we have $\mathrm{K} = \mathrm{K}(\boldsymbol{\rho})$ and $\mathrm{D} = \mathrm{D}_{IP}^{-1}$, whereas for S_{PBM}, we have $\mathrm{K} = \mathrm{K}(\boldsymbol{\rho})$ and $\mathrm{D} = \mathrm{P} - \mathrm{P}\,\mathrm{D}_{PBM}^{-1}\,\mathrm{P}$. We now want to show that the upper left block of S, which is $\mathrm{K} + \mathrm{B}(\boldsymbol{u})\mathrm{D}\mathrm{B}(\boldsymbol{u})^\top$, has the same sparsity structure as the stiffness matrix. That this is true of K is obvious, so we must show it for the second summand.

First, we observe the identity $B(\boldsymbol{u})B(\boldsymbol{u})^{\mathsf{T}} = {}^{m}_{i=1} B_{\cdot i}B_{\cdot i}^{\mathsf{T}} = {}^{m}_{i=1} K_i\boldsymbol{u}\ (K_i\boldsymbol{u})^{\mathsf{T}}$. The only non-zero components of $K_i\boldsymbol{u}$ are those corresponding to indices of non-zero entries of K_i, hence the dyadic product $K_i\boldsymbol{u}\ (K_i\boldsymbol{u})^{\mathsf{T}}$ has the same sparsity structure as K_i. The sum over all of these terms will therefore have the same sparsity structure as $K(\boldsymbol{\rho})$. This property extends to any matrices $B(\boldsymbol{u})AB(\boldsymbol{u})^{\mathsf{T}}$ where A is a diagonal matrix, and thus, in particular, to $B(\boldsymbol{u})DB(\boldsymbol{u})^{\mathsf{T}}$. The second diagonal block of S is a scalar and the off-diagonal blocks are row- and column-vectors, respectively, which are not generally sparse. Figure 3.1 shows a typical example of the sparsity structure of S as well as the matrix that S is a Schur complement of – which is given by (3.8) for S_{IP} and by (3.23) for $S_{P\ BM}$.

Figure 3.1: Sparsity structure of (a) the saddle-point matrix and (b) the final system matrix S for a three-dimensional uniform FE mesh with 512 elements.



When including contact constraints in the PBM method, we also add the term CHC^{T} to the top left block of S, see (3.29) in the previous section. Since each column of the the contact matrix C is non-zero only at indices corresponding to the displacement components of a single node, and because H is a diagonal matrix, CHC^{T} does not contain any off-diagonal terms that fall outside the sparsity structure of $K(\boldsymbol{\rho})$. Therefore, the sparsity structure of S also remains the same.

The above discussion was meant to explain our decision to solve the Schur complement systems instead of the larger systems (3.8) or (3.23) and to motivate the use of a (nearly) standard multigrid approach. However, it should be noted that systems of the form (3.8) or (3.23), typically called *saddle-point systems*, can also be solved by multigrid techniques, for example using so-called transforming smoothers [117]. These are required since saddle-point systems are not generally positive definite, case in point (3.8), so the traditional smoothers mentioned in Section 2.4.2 are not applicable. Maar and Schulz used a transforming smoother multigrid method for medium-scale two-dimensional topology optimization in [85]. It is questionable though whether that approach is preferable, firstly, because it requires additional transfer operators for variables defined element-wise rather than node-wise; secondly, because transforming smoothers are extremely involved compared to any of the standard smoothers for positive definite systems.

As explained in Section 2.4.1, both the CG and MINRES method use the residual norm to define a stopping criterion. In particular, for a linear system $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$, each method stops once $\|\boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k\|/\|\boldsymbol{b}\|$ drops below a given threshold value. In our numerical experiments, we set this threshold to 10^{-2}. It has to be noted that inexact IP methods usually require the solver tolerance to decrease during the optimization in order to guarantee convergence [14, 36]. And while our IP algorithm does fail to converge for some problems, as we will see in Section 3.2, this could not be fully remedied even with a very strict solver tolerance. Over all, fixing the value at 10^{-2} lead to the best performance in terms of overall computational time. The same is true for the PBM and OC method.

To improve the condition number of the system matrix S, we added regularization terms to those components that were affected by ill-conditioning in practice. In the IP method, small barrier parameter values lead to very small terms in the diagonal matrix D_{IP}, consequently to very large eigenvalues of D_{IP}^{-1} and thus of S_{IP}. We compensated for this by adding 10^{-5} to all diagonal entries in D_{IP}. In the PBM method, the penalty parameters never had to drop as low as the barrier parameters in the IP method, so that a regularization of D_{PBM} was not necessary. However, the values in $\boldsymbol{\rho}$ generally got a lot smaller than in the IP method, even when the latter was used with a lower bound of $\underline{\boldsymbol{\rho}} = \mathbf{0}$. Since this translates directly to small values of $\boldsymbol{\rho}$, the matrix $K(\boldsymbol{\rho})$ displayed very small eigenvalues. We therefore regularized it by adding 10^{-12} to each diagonal entry in $K(\boldsymbol{\rho})$. We observed a reduction in total solver iterations in some cases which was noticeable enough to warrant these regularization measures. However, it was not crucial to the general efficiency or convergence behaviour of the PBM method.

For the OC method, we used no regularization. In our numerical experiments, the number of solver iterations to solve a linear system was generally quite small and nearly constant over the course of the entire optimization. Hence, there was no need to regularize the stiffness matrix.

3.2 Numerical Results

We now test the algorithms proposed in the previous section on a variety of examples with different loading scenarios and geometries. First we will consider only cuboid design domains with uniform structured meshes that allow the use of the geometric multigrid method. We will compare the performance of the OC, IP and PBM method on these scenarios for FE meshes of medium size ($\sim 10^5$ elements) in Section 3.2.1, before looking at large-scale problems ($\geq 10^6$ elements) in Section 3.2.2. In Section 3.2.3, we will discuss results for unstructured meshes, comparing different strategies of using the (adaptive) smoothed aggregation multigrid method. Both Section 3.2.2 and Section 3.2.3 include examples with unilateral contact constraints. All computations in this thesis were performed in MATLAB R2019b (9.7.0.1190202) using the University of Birmingham's BlueBEAR High Performance Computing service [24]. Routines were written in MATLAB as well as C, using the C MEX API provided by MATLAB. Parallel processing is not available for the MATLAB installation on the BlueBEAR system, so that the code's performance is not competitive, but a representative comparison of the different optimization methods can still be drawn based on the results.

Optimization scenarios

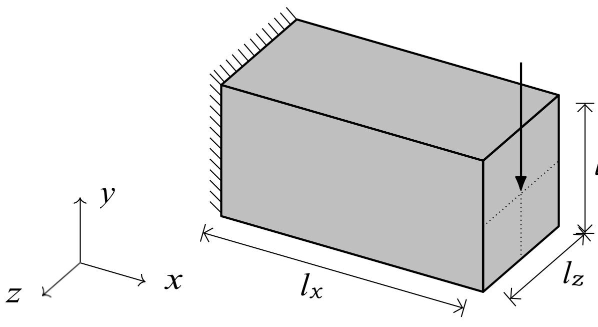
In the first part of this section, we consider three optimization scenarios with a cuboid design domain and different boundary conditions. In order to investigate how the algorithms' performances scale with the size of the problem, we vary both the proportions of the design domain and the mesh resolution. The FE mesh is a Cartesian grid of cube elements. We define it by specifying a coarse grid of m_x by m_y by m_z cube elements and the number of grid levels L. The grid hierarchy is obtained by splitting each coarse grid element into d^2 equal-sized fine grid elements, so that the finest grid in three dimensions has $m = 2^{L-1}m_x \cdot 2^{L-1}m_y \cdot 2^{L-1}m_z$ elements. Because the scaling of the element volume vector $\boldsymbol{a}$ has a big influence on the performance of our optimization algorithms, we scale the mesh so that the finest grid elements all have size 1. We also scale the load vector $\boldsymbol{f}$ so that $\|\boldsymbol{f}\| = 1$. Since we are dealing with linear elasticity, such a linear scaling is unproblematic.

Figure 3.2 shows the different optimization scenarios. The first one is a simple cantilever beam, completely fixed at one end and with a central point load applied at the other end. The second one is a table- or bridge-like structure, fixed at each of the four bottom corners and with a surface load applied to the top surface. Finally, the third scenario is a very popular academic example in topology optimization: an MBB beam modelled via symmetry constraints. Since we will consider these examples with different proportions and mesh sizes, we will refer to them as Cantilever m_x-m_y-m_z-L, Bridge m_x-m_y-m_z-L and MBB m_x-m_y-m_z-L.

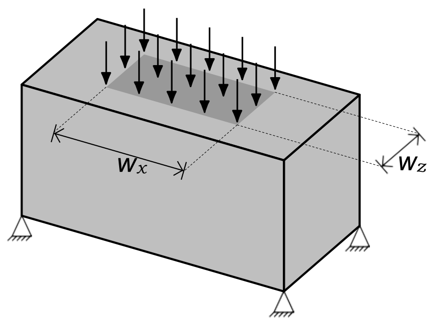
The volume constraint is defined as a fixed ratio of the volume of a fully solid design domain, with $V = 0.2(\;_i\bar{\rho}_i a_i)$ for three-dimensional and $V = 0.5(\;_i\bar{\rho}_i a_i)$ for two-dimensional problems. The upper bounds are $\bar{\rho}_i = 1$ for all $i = 1, \ldots, m$ and we choose zero lower bounds $\rho_i = 0, i = 1, \ldots, m$, which deserves some justification.

–

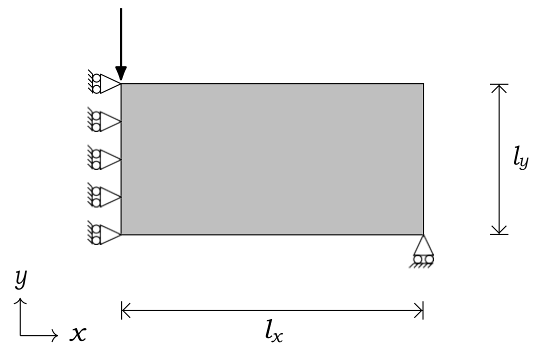
Figure 3.2: Loading scenarios and design domain for compliance minimization with geometric multigrid. The measurements of each design domain are $l_x = 2^{L-1}m_x$, $l_y = 2^{L-1}m_y$, $l_z = 2^{L-1}m_z$.



(a) Cantilever m_x-m_y-m_z-L



(b) Bridge m_x-m_y-m_z-L. Surface load is placed centrally, $w_x = l_x/16$, $w_z = l_z/16$.



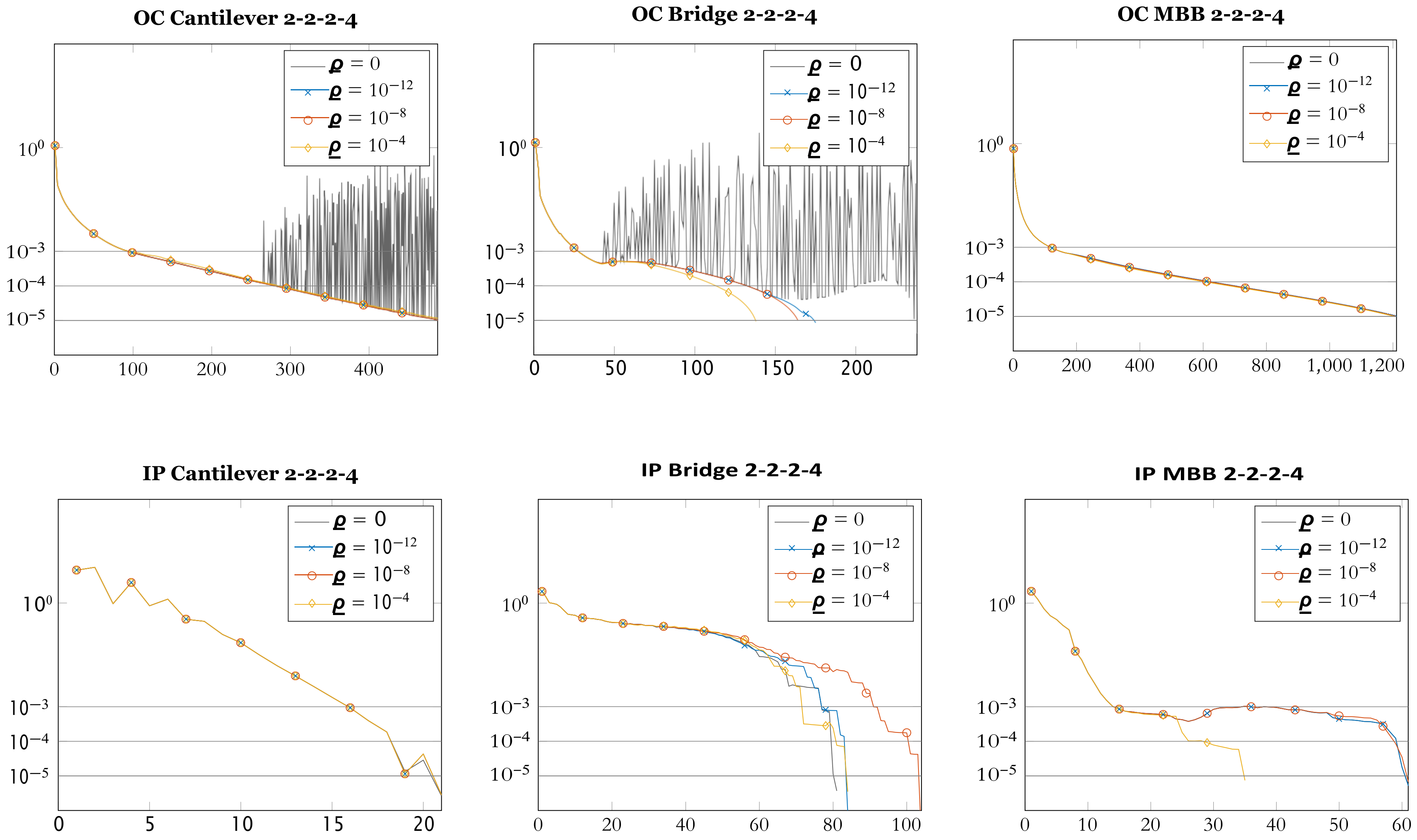
(c) MBB m_x-m_y-m_z-L. In three dimensions, boundary conditions and loads are extruded in the z-direction. Sliding in the z-direction is permitted on the left end but not on the right.

Lower density bound

We are ultimately trying to model a structure that is completely void of material in some areas of the design domain. If possible, we therefore want to permit the element densities to attain the value 0, so that we do not need to concern ourselves with the question of how much a non-zero bound value might distort the result. For the dual problem, this effectively just means dropping a set of variables and solving (2.60) rather than (2.54), which, if anything, simplifies the application of the PBM method. When solving the primal problem (2.48), one could however argue that allowing $\underline{\rho}_i = 0$ for some or all i means the problem is no longer well defined. First, the uniqueness of the displacements is lost; second, the stiffness matrix is no longer guaranteed to be invertible. Especially in the OC method, where we solve the nested formulation of the VTS problem and require the solution of $K(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$ in each iteration, this might cause the convergence behaviour to be compromised. In practise, we did not encounter such problems. To address the effect of zero lower density bounds on the convergence behaviour of the OC and IP method, let us inspect a few small scale examples for illustration. We solve each scenario on a mesh configuration with $m_x = m_y = m_z = 2$, $L = 4$, in turn setting all lower density bound values $\underline{\rho}_i$, $i = 1, \dots, m$, to 10^{-4}, 10^{-8}, 10^{-12} and 0. The value of the scaled duality gap $\tilde{\delta}(\boldsymbol{u}, \alpha)$ over the course of the optimization for each lower bound value is shown in Figure 3.3 for all scenarios and both the OC and IP method.

Let us first consider the IP results. For the cantilever and MBB beam scenario, the choice of the lower bound value seems to almost make no difference at all, with the exception of 10^{-4}. In the case of the bridge scenario, we see some variation in the duality gap trajectory in the final optimization iterations. Importantly, though, there is no indication that a zero lower bound has any detrimental effect on the algorithm. Now, we turn to the plots for the OC method. While the MBB beam scenario shows no effect of the lower bound values whatsoever, for the other two examples, we observe peculiar oscillations of the scaled duality gap value for $\underline{\boldsymbol{\rho}} = \mathbf{0}$

Figure 3.3: Scaled duality gap vs. iterations for the OC and IP method for different values of $\underline{\rho}_i$, $i = 1, \ldots, m$.



after a certain number of iterations. Note, however, that even the oscillatory graphs in Figure 3.3 can be tightly bounded below by smooth curves. In other words, in each case, there is a subsequence of iterates $\tilde{\delta}_k$ of scaled duality gap values that follows a smooth trajectory. For the cantilever problem, this subsequence even closely follows the same trajectory as the duality gap iterates for $\rho_i > 0$. For the bridge scenario, considerably more iterations are required when zero lower bounds are used. However, the OC method generally appears to be more sensitive to the lower bound values in this case. Moreover, the sequence of duality gap iterates still clearly converges in a "$\lim\ \inf$"-sense.

Still, two important questions need to be answered: first, whether the oscillations are due to the zero lower bounds; second, whether or not the algorithm's convergence behaviour suffers from this. Regarding the first question, the answer seems to be yes. While, in theory, density values can converge towards the lower bound 0 in both the OC and the IP method, they do so a lot faster in the former. In our examples, densities can reach values smaller than 10^{-100} before the OC method terminates, whereas they generally stay well above 10^{-10} in the IP method, which does not display any kind of oscillatory convergence behaviour. The condition number of the stiffness matrix $K(\boldsymbol{\rho})$ becomes very large in later OC iterations, much larger than that of the system matrix of the IP method. It seems plausible that the accuracy of the Krylov solver[1] approximation of the solution of $K(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$ would therefore be greatly compromised. Indeed, when using a direct method[2], we do not observe these oscillations. Interestingly, the OC method converges a lot quicker when using the approximate, albeit inaccurate, solution. It therefore appears that the answer to the second question – whether the algorithm's convergence deteriorates when choosing zero lower bounds – is no, for all practical purposes. It is also worth mentioning that the number of solver iterations required per OC iteration was the same for all lower bound values, so that the computational time does also not generally increase

[1]The results shown in Figure 3.3 were obtained using the MINRES method. When using the CG solver, oscillations are rare, but the OC method struggles to converge in general.
[2]Matlab's inbuilt Cholesky factorization solver

for $\underline{\boldsymbol{\rho}} = \mathbf{0}$.

From the examples included here for illustration and several similar ones omitted for the sake of brevity, we conclude that the choice $\underline{\rho}_i = 0, i = 1, \ldots, m,$ is justifiable even for the OC and IP method. The numerical artefacts that we observe in the OC method do not appear to impede the convergence of iterate subsequences to a solution.

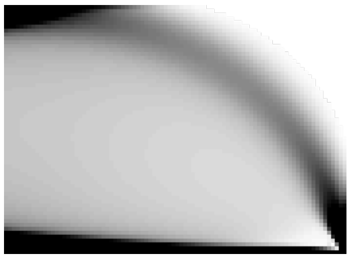
Stopping criterion

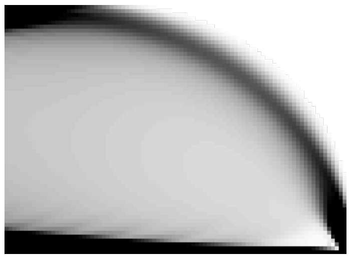
In the sections describing the IP, PDNR and OC method, it was mentioned that we terminate each algorithm once $\tilde{\delta}(\boldsymbol{u}, \alpha)$ falls below 10^{-5}, where $\tilde{\delta}$ is the duality gap (2.62) scaled by either the primal or dual objective function. This stopping criterion appeared to be more indicative of a convergence of the design than others which are more common for the OC method, such as a minimum difference in the design variables between iterations or a minimum objective function change. To illustrate this, we use the example MBB 8-2-0-6, OC solutions of which are shown in Figure 3.4 for decreasing stopping threshold values. We zoom in on the right end of the design domain, as this is where the design change is most evident. While a vague approximation of the optimal design is reached early on, the contours remain blurry until $\tilde{\delta}(\boldsymbol{u}, \alpha) < 10^{-5}$. Any further change obtained when pushing $\tilde{\delta}(\boldsymbol{u}, \alpha)$ below 10^{-6} is barely visible. We observed the same behaviour for the IP and PBM method, which justifies the choice $\varepsilon_{\text{IP}} = \varepsilon_{\text{PBM}} = \varepsilon_{\text{OC}} = 10^{-5}$.

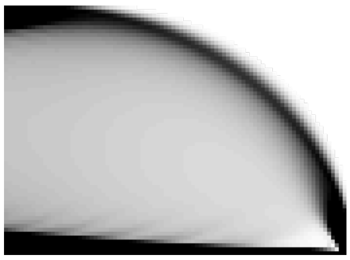
3.2.1 Comparison of Optimization Methods and Solvers

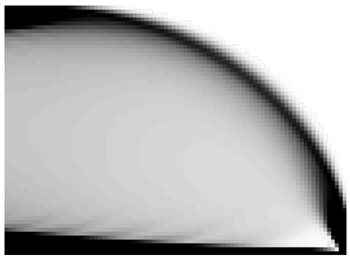
To compare the efficiency of the IP, PBM and OC method, we apply them to all three scenarios seen in Figure 3.2. Furthermore, for each scenario, we consider two different sets of mesh dimensions m_x-m_y-m_z-L. The first mesh, defined by $m_x = 4$, $m_y = 2$, $m_z = 2$ and $L = 6$, has $m = 524\,288$ elements. The second mesh has the specifications $m_x = 16$, $m_y = 2$, $m_z = 2$ and $L = 5$, ($m = 262\,144$), which

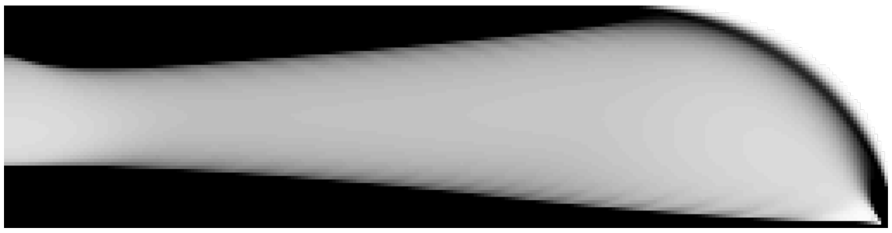
Figure 3.4: MBB 8-2-0-6 solutions for $\tilde{\delta}(\boldsymbol{u},\alpha) < \varepsilon_{\text{OC}} = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$. Pixel-colour corresponds to element density, (black: $\rho_i = \overline{\rho_i}$, white: $\rho_i = \underline{\rho_i}$).


(a) $\varepsilon_{\text{OC}} = 10^{-3}$


(b) $\varepsilon_{\text{OC}} = 10^{-4}$


(c) $\varepsilon_{\text{OC}} = 10^{-5}$


(d) $\varepsilon_{\text{OC}} = 10^{-6}$


(e) $\varepsilon_{\text{OC}} = 10^{-6}$, entire design domain

gives us a strongly elongated design domain. While the first problem is relatively well-behaved, as it were, the second one is more of a pathological example, as the stiffness matrix (at the optimal solution) becomes more ill-conditioned as the design domain aspect ratio increases.

For each optimization method, we present the results obtained with the solver that worked best overall. In both the IP and PBM algorithm, the choice of the solver did not make a very big difference. The tables in Appendix A.1 compare the CG and MINRES solver on an exhaustive range of problems that are roughly the same size as those discussed here. For the sake of completeness, Section 3.2.4 also looks at the performance of the two Krylov solvers on single linear systems and for a stricter solver tolerance. On average, the MINRES method required fewer iterations, as expected, but the CG method was faster in terms of CPU time. The OC method, interestingly, converged considerably faster when using the MINRES method, not just in terms of total solver iterations, but in terms of OC iterations required. Solving the same problem using the CG method sometimes took up to five times as long and up to ten times as many solver iterations. A possible explanation is that the MINRES method can solve even singular systems, whereas the CG method does not converge for any given solver tolerance in such cases. As mentioned earlier, using zero lower bounds for the densities in the OC method can lead to strongly ill-conditioned stiffness matrices, which are presumably harder for the CG method to cope with, even for a high stopping tolerance like 10^{-2}. Since MINRES so clearly outperformed CG, we omit a detailed comparison of the solvers for the OC method.

Tables 3.1 to 3.6 show the performance of the different optimization algorithms for each problem, in terms of iterations and CPU time. The column titled "Nwt" lists the total number of linear systems solved over the course of the iteration – which is the same as the number of Newton iterations in the case of the IP and PBM method. For the sake of completeness, we have also included the final objective function

Table 3.1: Cantilever 4-2-2-6 results for the PBM, IP and OC method. Problem dimensions: m = 524 288, n = 1 622 400. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	82	378	59	11	1.655 186
IP	463	2177	311	81	1.655 160
OC	267	534	165	40	1.655 396
OC ($\varepsilon_{\mathrm{OC}} = 10^{-4}$)	222	444	137	33	1.655 411

Table 3.2: Bridge 4-2-2-6 results for the PBM, IP and OC method. Problem dimensions: m = 524 288, n = 1 635 063. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	85	555	63	16	42.000 503
IP	77	422	56	16	42.000 251
OC	100	847	87	40	42.001 466
OC ($\varepsilon_{\mathrm{OC}} = 10^{-4}$)	82	685	71	32	42.002 165

value[1], even though the differences seen between the methods do not correlate with any visible differences in the final design. The first thing that becomes clear from our results is that the OC method is generally a lot slower than both the IP and the PBM method. In practice, the OC method often converges to a rough approximation of the optimal design in just a few iterations and one might therefore argue that a stopping threshold of $\varepsilon_{\mathrm{OC}} = 10^{-4}$ is enough to obtain reasonable results. To pre-

[1] While the solver tolerance used throughout the optimization is only 10^{-2}, in order to determine a more accurate value for $\frac{1}{2}\boldsymbol{f}^T\boldsymbol{u}$, we obtained $\boldsymbol{u}$ from the equilibrium equations (2.48b) for the final $\boldsymbol{\rho}$, solved to an accuracy of 10^{-8}.

Table 3.3: MBB 4-2-2-6 results for the PBM, IP and OC method. Problem dimensions: m = 524 288, n = 1 630 720. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	68	364	52	11	71.246 932
IP	41	394	29	11	71.247 72
OC	845	4221	584	209	71.247 551
OC ($\varepsilon_{\mathrm{OC}} = 10^{-4}$)	497	2481	348	124	71.247 719

Table 3.4: Cantilever 16-2-2-5 results for the PBM, IP and OC method. Problem dimensions: m = 262 144, n = 836 352. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	54	320	20	5	66.193 288
IP	26	111	7	2	66.192 963
OC	956	4775	304	108	66.194 033
OC ($\varepsilon_{\rm OC} = 10^{-4}$)	555	2770	178	63	66.194 118

Table 3.5: Bridge 16-2-2-5 results for the PBM, IP and OC method. Problem dimensions: m = 262 144, n = 839 607. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	69	492	33	8	236.928 109
IP	34	314	17	6	236.928 016
OC	2080	16 650	765	333	236.928 115
OC ($\varepsilon_{\rm OC} = 10^{-4}$)	1290	10 330	474	208	236.928 139

Table 3.6: MBB 16-2-2-5 results for the PBM, IP and OC method. Problem dimensions: m = 262 144, n = 838 464. Time values are rounded to the nearest integer.

	iterations		time [min]		
method	Nwt	solver	total	solver	obj fun
PBM	35	299	19	5	2194.268 965
IP	20	106	8	2	2193.685 598
OC	3371	20 224	1263	459	2194.260 331
OC ($\varepsilon_{\rm OC} = 10^{-4}$)	463	2776	183	69	2194.262 766

empt such a suggestion, we have also included the results for this case in the tables, showing that even with this relaxed stopping criterion, the OC method is still much slower. In particular, it seems to struggle more with problems on elongated design domains, possibly due to the condition number of the stiffness matrix[1], whereas the IP and PBM method do not seem to be influenced by this.

Figure 3.5 shows the trajectory of the duality gap. We again observe an oscillatory progression for the OC method with a convergent subsequence, due to the zero lower density bounds. Appendix A.1 contains more plots like Figure 3.5, with the duality gap plotted against the total number of Newton and solver iterations.

In Figure 3.6, we can see how the number of solver iterations needed to solve each linear system evolves over the course of the optimization. While this number is constant for the OC method (the trend observed for the iterations within the plot range is continued), we can see that more solver iterations are required per linear system as the IP and PBM method progress. The slopes of the curves suggest that the systems encountered in those methods are more difficult to solve than any system occurring in the OC method. Nevertheless, the superior convergence behaviour of the IP and PBM algorithm more than make up for this.

Since the OC method is very clearly no competition, let us now turn to a comparison of the IP and PBM algorithm. In all but one example, the former is considerably faster than the latter. However, in that one exception – Cantilever 4-2-2-6 – the IP method converges so slowly that it is even outperformed by the OC method. This is a first indication of a lack of robustness that is even more pronounced in large-scale problems, as we will see in the next part of this section.

A final observation that is of note is the difference between the total CPU time and the total time taken up by calls to the solver routines, as seen in Tables 3.1 to 3.6. It turns out that a major part of the overall computation time is taken up by

[1] Although it should be noted that even with a non-zero lower density bound or a regularization of the stiffness matrix, the OC method did not perform any better.

Figure 3.5: Scaled duality gap vs. CPU time (in minutes) for the PBM, IP and OC method. A black diamond marks the termination of the algorithm, which can be outside the plot range.

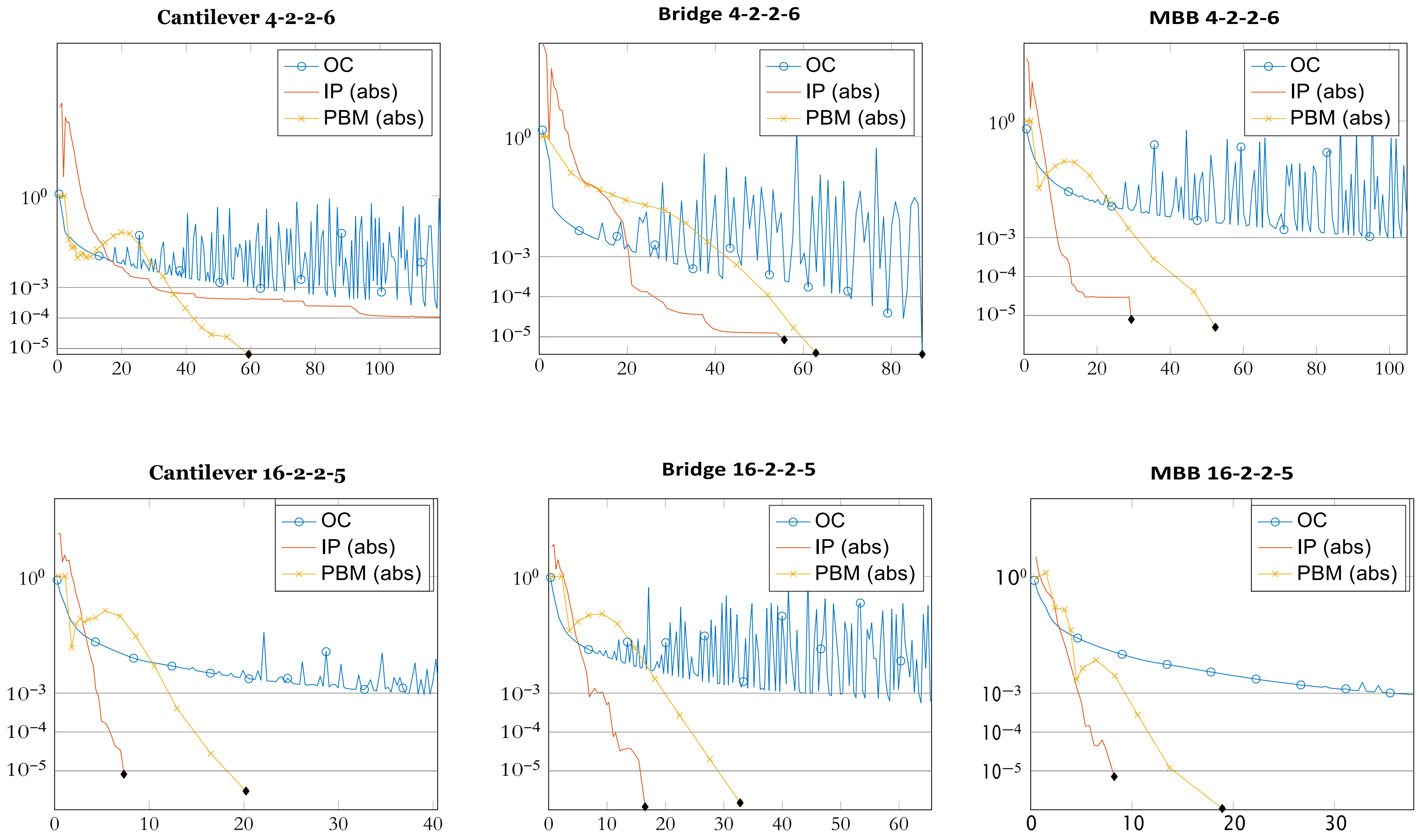
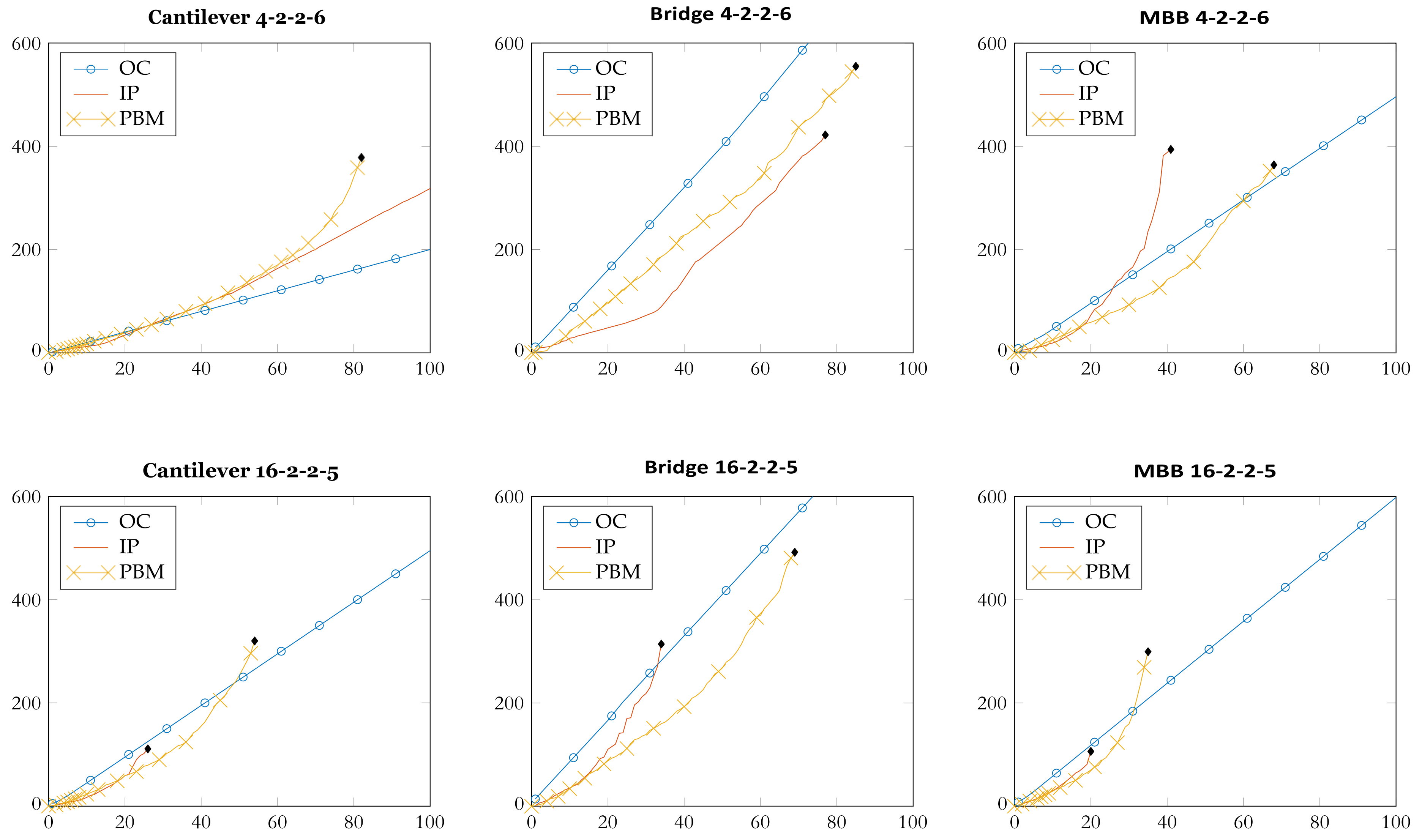


Figure 3.6: Accumulated solver iterations vs. Newton iterations for the PBM, IP and OC method. A black diamond marks the termination of the algorithm, which can be outside the plot range.



the construction of the system matrices. This is true even for the OC method, where the system matrix is just the stiffness matrix, the assembly of which is implemented in C. It is noticeable, though, that the proportion of the CPU time taken up by the solver is even smaller for the IP and PBM method. Indeed, for those methods, the construction of the system matrix, which involves the concatenation, multiplication and addition of other large sparse matrices, takes up roughly half of the total time required for the optimization.

3.2.2 Large Scale Problems

We now turn to scenarios with more than a million finite elements. For this, we consider the same three problems as before, again varying the mesh dimensions, this time setting the number of grid levels to $L = 7$. Judging by our previous results, we can reject the OC method as a viable candidate and only consider results for the IP and PBM method. Tables 3.7 to 3.9 list the number of iterations and time required for different instances of each of the three scenarios. In order to see how the algorithms' performances scale with the problem size, the tables also include results for problems defined by the same scenario and design domain proportions, but with a smaller number of grid levels. Figures 3.7 to 3.9 show the optimal design for the largest example of each scenario. Since the large proportion of "grey" elements, which are due to the linear material interpolation in the VTS formulation, don't allow for a straightforward interpretation of the optimal solution as a $0-1$ design, we visualize each result for a range of different density thresholds.

Regarding the choice of linear solver, CG again slightly but consistently outperforms MINRES when used in the IP method. When solving problems with $L = 7$ by the PBM method, on the other hand, the smaller number of iterations needed by the MINRES solver offsets its lower efficiency enough for it to be faster on average than the CG solver. For the purpose of consistency, all PBM results presented in Tables 3.7 to 3.9 have been obtained using the MINRES solver, even though for

Table 3.7: Cantilever m_x-m_y-m_z-L, medium- to large-scale. Newton iterations, solver iterations and CPU time (in minutes) for the IP and PBM method. Missing IP entries indicate that the IP method either timed out at 48 hours or reached the maximum iteration number (500) before converging.

Problem dimensions			IP			PBM		
m_x-m_y-m_z-L	m	n	Nwt	solver	time	Nwt	solver	time
2-2-2-5	32 768	104 544	51	230	2	71	244	3
4-2-2-5	65 536	209 088	44	227	4	58	218	5
6-2-2-5	98 304	313 632	54	391	7	60	297	9
8-2-2-5	131 072	418 176	41	258	8	57	274	11
2-2-2-6	262 144	811 200	–	–	–	118	411	42
4-2-2-6	524 288	1 622 400	463	2177	265	82	378	69
6-2-2-6	786 432	2 433 600	344	1982	335	74	297	86
8-2-2-6	1 048 576	3 244 800	–	–	–	72	314	122
2-2-2-7	2 097 152	6 390 144	–	–	–	130	206	413
4-2-2-7	4 194 304	12 780 288	–	–	–	104	239	799
6-2-2-7	6 291 456	19 170 432	–	–	–	93	223	1030
8-2-2-7	8 388 608	25 560 576	–	–	–	87	217	1092

$L < 7$, the CG solver is usually a bit faster.

The first thing that becomes evident from a comparison of the two optimization methods is that the IP algorithm does not do well on large-scale problems. It often stagnates and terminates due to a time or iteration limit. In many cases in which it does converge, it loses its advantage over the PBM method, requiring many more Newton and solver iterations. One may ask whether this simply comes down to the wrong parameters. For example, the choice of a constant value for the solver tolerance goes against the typical IP paradigm of decreasing the solver tolerance as one approaches the optimum. Indeed, the original version of our IP method, which was proposed in [34], featured an adaptive scaling of the solver tolerance. While this made the method more successful for large-scale problems, it also considerably slowed it down for many other problems due to an unnecessarily low solver tolerance. Furthermore, even with this scaling mechanism, the IP method could not solve all of the large-scale problems and was not generally more efficient.

Turning now towards the PBM method, it clearly seems to be much more robust,

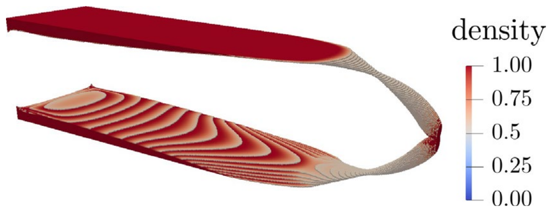
Table 3.8: Bridge m_x-m_y-m_z-L, medium- to large-scale. Newton iterations, solver iterations and CPU time (in minutes) for the IP and PBM method. Missing IP entries indicate that the IP method either timed out at 48 hours or reached the maximum iteration number (500) before converging.

Problem dimensions			IP			PBM		
m_x-m_y-m_z-L	m	n	Nwt	solver	time	Nwt	solver	time
2-2-2-5	32 768	107 799	31	139	1	66	352	3
4-2-2-5	65 536	212 343	33	155	3	66	369	6
6-2-2-5	98 304	316 887	30	173	4	73	457	10
8-2-2-5	131 072	421 431	28	198	5	66	374	12
2-2-2-6	262 144	823 863	64	320	20	76	451	29
4-2-2-6	524 288	1 635 063	77	422	46	85	555	68
6-2-2-6	786 432	2 446 263	73	466	77	83	517	96
8-2-2-6	1 048 576	3 257 463	70	530	100	84	575	132
2-2-2-7	2 097 152	6 440 055	186	1005	479	144	945	549
4-2-2-7	4 194 304	12 830 199	53	268	360	131	808	1120
6-2-2-7	6 291 456	19 220 343	–	–	–	144	977	1794
8-2-2-7	8 388 608	25 610 487	–	–	–	129	881	2254

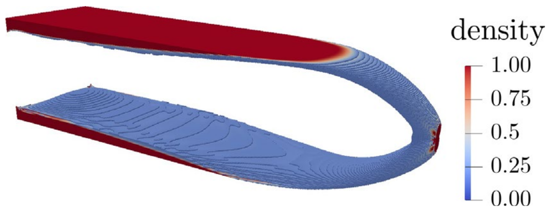
Table 3.9: MBB m_x-m_y-m_z-L, medium- to large-scale. Newton iterations, solver iterations and CPU time (in minutes) for the IP and PBM method. Missing IP entries indicate that the IP method either timed out at 48 hours or reached the maximum iteration number (500) before converging.

Problem dimensions			IP			PBM		
m_x-m_y-m_z-L	m	n	Nwt	solver	time	Nwt	solver	time
2-2-2-5	32 768	106 656	23	130	1	53	286	2
4-2-2-5	65 536	211 200	22	124	2	51	287	5
6-2-2-5	98 304	315 744	23	122	3	43	226	7
8-2-2-5	131 072	420 288	19	81	3	40	228	8
2-2-2-6	262 144	819 520	40	262	12	63	323	24
4-2-2-6	524 288	1 630 720	41	394	32	68	364	55
6-2-2-6	786 432	2 441 920	46	550	65	59	347	71
8-2-2-6	1 048 576	3 253 120	37	418	64	63	415	105
2-2-2-7	2 097 152	6 423 168	220	1753	623	81	271	279
4-2-2-7	4 194 304	12 813 312	249	2176	1995	81	283	685
6-2-2-7	6 291 456	19 203 456	171	1551	2009	79	305	937
8-2-2-7	8 388 608	25 593 600	193	2235	2819	69	248	843

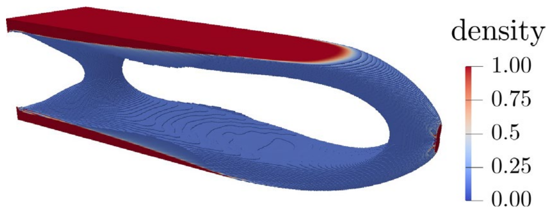
Figure 3.7: Cantilever 8-2-2-7 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.



(a) $c = 0.6$



(b) $c = 0.7$

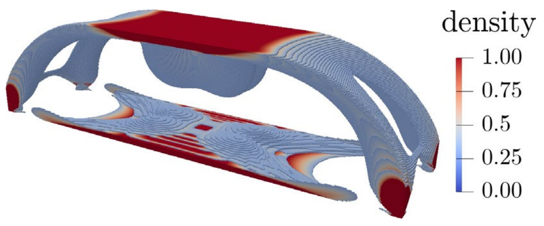


(c) $c = 0.8$

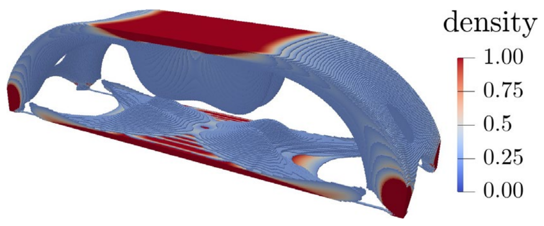
as well as much more efficient for large-scale problems, than our IP algorithm. The number of both Newton and solver iterations increases along with the problem size, presumably because the problems simply become more difficult for a higher mesh resolution. The fact that the average number of solver iterations per Newton iteration, i.e. per linear system, remains roughly the same, while the solver tolerance is kept constant, suggests that the multigrid preconditioner works as intended. The same observation can be made for the IP method – when it converges successfully.

To showcase our PBM algorithm's ability to handle contact constraints, we also include results for the problem shown in Figure 3.10. The mesh dimensions and loading are the same as for Cantilever 7-2-2-7, but the structure is not fixed at the left end. Instead, we restrict vertical movement through obstacles that are flush with the top and bottom surface of the design domain. The optimal design is seen in Figure 3.11. The depicted mesh has 7 340 032 elements and 22 415 427 DOFs for this scenario. Solving this problem took 115 Newton iterations, 273 MINRES iterations and 1378 minutes of CPU time. Note that this problem is technically

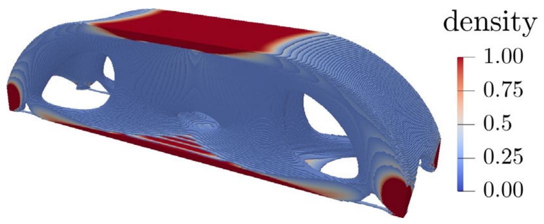
Figure 3.8: Bridge 8-2-2-7 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.



(a) $c = 0.6$

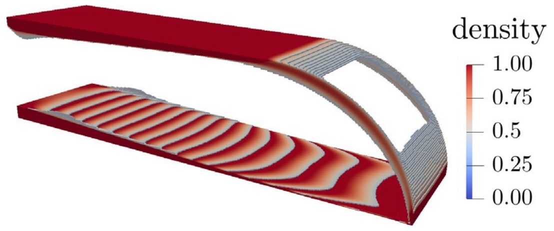


(b) $c = 0.7$

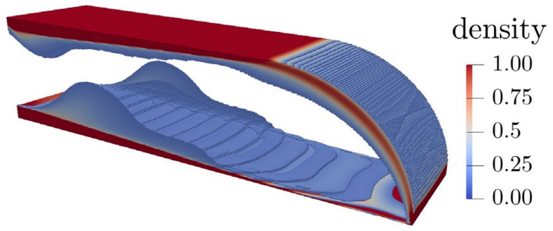


(c) $c = 0.8$

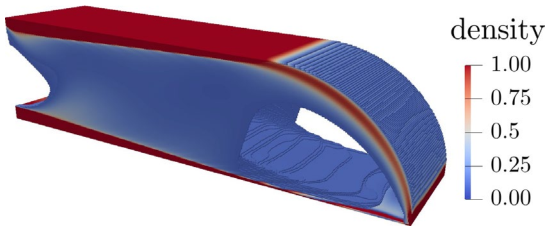
Figure 3.9: MBB 8-2-2-7 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.



(a) $c = 0.6$



(b) $c = 0.7$



(c) $c = 0.8$

Figure 3.10: Cantilever with clamping boundary conditions (cf. example Cantilever m_x-m_y-m_z-L in Figure 3.2). The darker grey regions are obstacles with a no-friction surface, modelled by unilateral contact constraints. In three dimensions, the load is a point load applied to the centre of the right end.

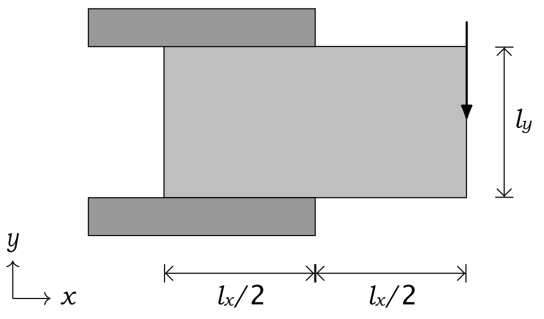
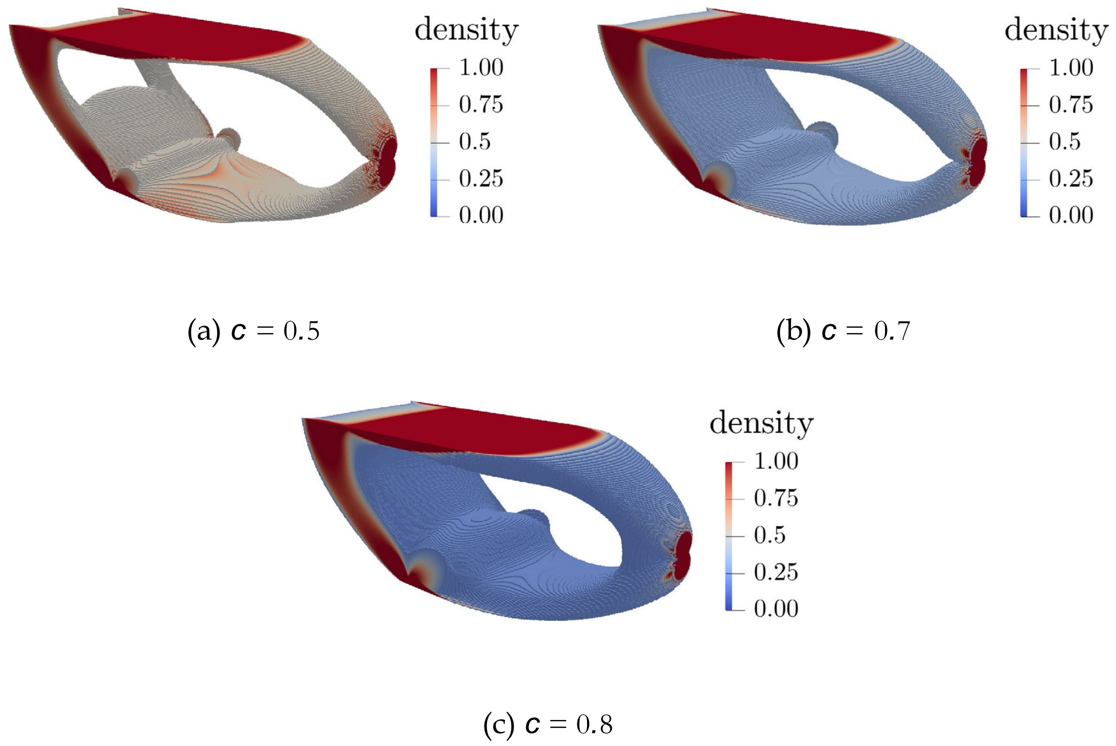


Figure 3.11: Clamped cantilever optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$. The mesh dimensions are the same as for Cantilever 7-2-2-7.



(a) $c = 0.5$ (b) $c = 0.7$

(c) $c = 0.8$

not kinematically determinate, but due to the vertical load direction and geometric linearity it still has a solution. The regularization of the stiffness matrix, however, is critical in this case, because the system matrix is not positive definite otherwise. We could also have included an additional minimal boundary condition to ensure regularity of the stiffness matrix. For example, one could simply fix two or three nodes at the left end of the design domain. The reason we decided against this is that it causes a spurious strip of grey or solid material to appear that connects the fixed nodes to the rest of the design, which does not otherwise have any material at the left end of the design domain.

3.2.3 Unstructured Meshes and Algebraic Multigrid

We now extend our method to problems defined on unstructured meshes, which necessitates the use of an algebraic multigrid method. See Section 2.4.2 for a description of the smoothed aggregation algorithm that we will be using. For all of the following results, we used the PBM method, since it has so far proven to be the most reliable method for large-scale problems, and because we want to include contact constraints. As problems on unstructured grids seem to require more solver iterations in general, we use the MINRES method, because it appears to be slightly more efficient than CG for large-scale problems.

In a recent paper [96], Peetz and Elbanna looked into the application of AMG methods to topology optimization problems based on the SIMP formulation. They used the MMA, equipped with an SA preconditioned solver, but considered only structured meshes that can also be treated by geometric multigrid methods. Their focus was the ability of AMG methods to handle the material anisotropy seen in SIMP solutions. New transfer operators were computed each time the system matrix changed. The authors reported fewer overall solver iterations compared to the geometric multigrid, however, in terms of CPU time, the AMG method was only more efficient for some three dimensional problems, when the reduction in iteration numbers was enough to compensate for the large overhead of the smoothed aggregation. They also obtained promising results with a hybrid GMG-AMG method, using GMG transfer operators on a certain number of finer grid levels before switching to the AMG method for the coarser levels. A similar approach was used in [3], although without any discussion of the reasons or benefits. In the following, we will focus more on the use of AMG where it is without alternative, i.e. for design domains discretized by unstructured meshes. Nevertheless, we begin by considering a few structured grid examples, in order to compare the performance of GMG and AMG.

First of all, we go through some of the details of our SA implementation and how we used it within our PBM algorithm. Again, everything was written from scratch in MATLAB, with a few auxiliary routines written in C, and the code was not run in parallel. The sparsity of the coarse-level operators is critical to the efficiency of the multigrid, and this is determined by the threshold parameter ε_{N} in the definition of the strongly-coupled neighbourhoods (2.81). After some trial and error, we set $\varepsilon_{\mathrm{N}} = 0.04$. Following [124], this parameter is reduced by a constant factor on each level, which we set to 0.1. We use the weighted Jacobi prolongation smoother (2.86).

As discussed in Section 3.1.4, we define the multigrid transfer operators blockwise: since the upper left $n \times n$ block of the system matrix has the same sparsity structure as the stiffness matrix, we use the standard multigrid operator for the first n of the system's degrees of freedom, while we simply use a 1 for the remaining scalar DOF. We will follow the same approach now, only that the first block of the transfer operator is determined by the SA method.

In order to properly assess the AMG preconditioner, we provide detailed statistics for the GMG preconditioner in Table 3.10 as a reference. As before, we list the total Newton and solver iterations, the total CPU time, and the time it took to solve all linear systems. We now further include the time required for the multigrid setup. This comprises the construction of transfer operators and the computation of all coarse-level system matrices, all of which is done outside of the call to the solver routines. The operations that contribute the most to the solver time, on the other hand, are matrix-vector multiplications, including those inside the multigrid preconditioner which use the previously prepared coarse-level operators.

When solving the same problems with an AMG preconditioner, the number of coarse levels is not known a priori. We therefore define a maximum number of levels $L_{\max}$, as well as a minimum number of DOFs that a coarse level can have, denoted by $n_{\min}$. The coarsest algebraic grid should have roughly the same size as the coarsest geometric grid so that computational cost for the direct solver on that level is comparable. Therefore, we set $L_{\max} = L = 5$ and $n_{\min} = 3^d d = 81$, which is approximately the number of DOFs for a 2–by–2–by–2 element mesh. In practice, the number of algebraic grid levels was 3 or 4 and the minimum system size ranged from 81 to 918.

Were we to apply the SA method naively as a black box algorithm, we would construct new transfer operators in each Newton iteration, performing the aggregation based on the system matrix (or rather, its upper left $n \times n$ block). The results can be seen in Table 3.11. They are obviously very bad. Not only is the average overhead for the multigrid setup roughly fifty times larger than for the geometric multigrid; even the number of solver iterations have increased dramatically. The inefficiency of the above approach is not too surprising when one considers that many of the central multigrid paradigms are closely related to the properties of matrices that stem from elliptic problems. The matrix $\mathrm{S}_{P\ BM}$ of the system that we need to solve does not itself correspond to any such problem. Moreover, the GMG method which we have used successfully as a preconditioner for this system is not an optimal method for this matrix, but for the stiffness matrix $\mathrm{K}(\boldsymbol{\rho})$ – even though, strictly speaking, only when $\boldsymbol{\rho}$ is constant. The logical next step is to adjust the AMG preconditioner such that the aggregation is performed not on the actual system matrix, but on the current stiffness matrix. Results for this approach are given in Table 3.12. We can

Table 3.10: Detailed statistics for different problems solved by the PBM method with a GMG-preconditioned MINRES solver. Time values are rounded.

	iterations		time [seconds]		
Problem	Nwt	solver	total	solver	GMG
Cantilever 2-2-2-5	82	213	226	47	31
Cantilever 4-2-2-5	57	131	304	57	40
Cantilever 6-2-2-5	61	170	612	120	91
Cantilever 8-2-2-5	57	165	656	135	86
Bridge 2-2-2-5	68	286	207	52	27
Bridge 4-2-2-5	71	295	417	103	59
Bridge 6-2-2-5	71	353	728	187	99
Bridge 8-2-2-5	71	280	816	204	108
MBB 2-2-2-5	60	263	184	47	24
MBB 4-2-2-5	56	252	336	89	48
MBB 6-2-2-5	42	144	388	85	54
MBB 8-2-2-5	39	107	489	91	73
average	61	222	447	101	62

Table 3.11: Detailed statistics for different problems solved by the PBM method with an SA-AMG-preconditioned MINRES solver; SA is used on S_{PBM} in each Newton iteration. Time values are rounded.

	iterations		time [seconds]		
Problem	Nwt	solver	total	solver	SA(S)
Cantilever 2-2-2-5	66	436	601	111	394
Cantilever 4-2-2-5	89	1881	4886	1137	3471
Cantilever 6-2-2-5	63	712	3289	384	2610
Cantilever 8-2-2-5	85	1305	9324	1032	7776
Bridge 2-2-2-5	71	601	700	186	401
Bridge 4-2-2-5	70	773	1627	388	1018
Bridge 6-2-2-5	96	1593	7733	1799	5466
Bridge 8-2-2-5	100	1876	8668	2005	5973
MBB 2-2-2-5	59	608	663	196	370
MBB 4-2-2-5	68	1718	1782	658	916
MBB 6-2-2-5	100	7378	9542	4661	4419
MBB 8-2-2-5	80	3754	9608	3765	5227
average	79	1886	4869	1360	3170

Table 3.12: Detailed statistics for different problems solved by the PBM method with an SA-AMG-preconditioned MINRES solver; SA is used on $\mathrm{K}(\boldsymbol{\rho})$ in each Newton iteration. Time values are rounded.

	iterations		time [seconds]		
Problem	Nwt	solver	total	solver	SA($\mathrm{K}(\boldsymbol{\rho})$)
Cantilever 2-2-2-5	74	297	619	102	393
Cantilever 4-2-2-5	68	408	1603	189	1216
Cantilever 6-2-2-5	59	432	2103	290	1506
Cantilever 8-2-2-5	81	711	4977	663	3720
Bridge 2-2-2-5	69	292	637	110	414
Bridge 4-2-2-5	78	549	1740	306	1182
Bridge 6-2-2-5	83	686	3128	513	2225
Bridge 8-2-2-5	78	717	4895	689	3681
MBB 2-2-2-5	53	285	466	92	287
MBB 4-2-2-5	56	509	1078	236	668
MBB 6-2-2-5	53	539	2254	382	1575
MBB 8-2-2-5	46	477	2892	423	2123
average	66	492	2199	333	1582

see a considerable improvement over the system matrix–based AMG preconditioner. However, the total number of solver iterations is still more than twice as large on average as for the GMG method. This is interesting as Peetz and Elbanna [96] reported a decrease in the number of solver iterations for the AMG preconditioner. A possible explanation is that these authors looked at problems based on the SIMP formulation, where one typically sees much sharper density changes in the design. Thus, differences in the material properties of geometrically neighbouring elements are more pronounced, which leads to a stronger distinction between geometric and algebraic neighbours. In other words, the local anisotropy in designs resulting from the SIMP formulation is much greater, which is precisely one of the cases in which AMG methods are known to hold an advantage over GMG methods. Since solutions to the VTS problem usually display much smaller density gradients, it seems plausible that the geometric multigrid is better suited after all to eliminate error components which correspond to the design's low-frequency eigenmodes.

Seeing as the use of AMG transfer operators tailored to each individual system matrix is not only much less effective overall than using GMG transfer operators,

Table 3.13: Detailed statistics for different problems solved by the PBM method with an SA-AMG-preconditioned MINRES solver; SA is used on $\mathrm{K}(\overline{\boldsymbol{\rho}})$ once at the start of the optimization. Time values are rounded.

	iterations		time [seconds]		
Problem	Nwt	solver	total	solver	SA($\mathrm{K}(\overline{\boldsymbol{\rho}})$)
Cantilever 2-2-2-5	76	337	233	66	51
Cantilever 4-2-2-5	65	367	535	167	125
Cantilever 6-2-2-5	64	473	899	336	206
Cantilever 8-2-2-5	67	564	1138	429	253
Bridge 2-2-2-5	67	297	269	82	66
Bridge 4-2-2-5	70	466	597	208	137
Bridge 6-2-2-5	80	634	1108	442	248
Bridge 8-2-2-5	59	483	1029	409	239
MBB 2-2-2-5	59	398	266	100	58
MBB 4-2-2-5	54	399	413	149	93
MBB 6-2-2-5	52	449	763	313	169
MBB 8-2-2-5	40	361	749	294	176
average	63	436	667	250	152

but also has an enormous computational overhead, we venture an informed guess as to what might be a more efficient approach: Setting up the transfer operators once at the beginning, performing the aggregation on the stiffness matrix $\mathrm{K}(\overline{\boldsymbol{\rho}})$, which corresponds to an all-solid design domain[1]. Using the resulting transfer operator throughout the entire optimization is essentially as close as we can get to emulating the methodology of the GMG preconditioner. The results are given in Table 3.13 and are the best AMG results yet, both in terms of solver iterations and time.

While it does not look as if the AMG preconditioner can match the efficiency of the GMG preconditioner for problems on structured meshes, the observations we have made so far can inform the way we apply it to problems with unstructured meshes. We will consider three examples, shown in Figure 3.12. We will again vary the dimension d and discretization resolution of the design domains, although we keep the geometric proportions fixed. The size of the design domain and mesh are again defined by the parameter L. The relationship between the number of elements m and L is now less straightforward, but generally m is in the order of

[1]Recall that we chose $\overline{\rho}_i = 1$ for all $i = 1, \dots, m$.

$2^{d(L-1)}$, which is consistent with the structured mesh specifications at the beginning of this section. For scaling purposes, the size of each element is around 1, unless otherwise specified, and the load vector is normalized. All meshes use quadrilateral ($d = 2$) or hexahedral ($d = 3$) finite elements. They were created in Gmsh [54], version 4.7.1. The first example is a loaded knee structure, fixed at the top end and subjected to a vertical load at the right end. It is shown in Figure 3.12a and referred to as Knee *d*-*L*. The design domain of the second example, which we refer to as Lug *d*-*L*, see Figure 3.12b, represents the lug of a hinge[1,2]. The hinge axis is a rigid obstacle modelled by unilateral contact constraints. The third scenario, shown in Figure 3.12c, is a simplified representation of a crack under tension, modelled by symmetry conditions[3]. It is an example of a structural problem where an unstructured mesh is not required due to geometric irregularities, but because a finer mesh resolution is needed at points where stress singularities are expected, which in this case is the point A in Figure 3.12c. The elements around point A have an edge length of approximately $0.2 - 0.5$. Along the slanted edges, this factor increases to about $0.5 - 0.7$ at point B.

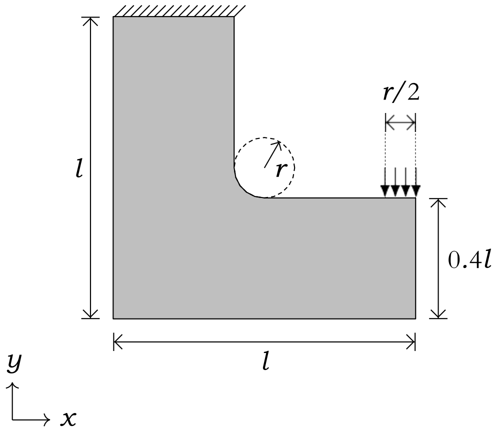
In light of a recent article published on arXiv.org [122] which investigated the mesh-dependence of topology optimization algorithms in the context of non-uniform meshes, it should be acknowledged that we did not apply the same rigour in our approach. The aforementioned preprint argued, based on an analysis of the underlying infinite-dimensional problem, that special inner products, which account for varying element sizes, should be incorporated in first and second order derivatives. It was shown that ignoring these inner products can indeed have a substantial impact on the performance of the optimization method. However, the focus of the paper was

[1] The idea for this problem is taken from [120].

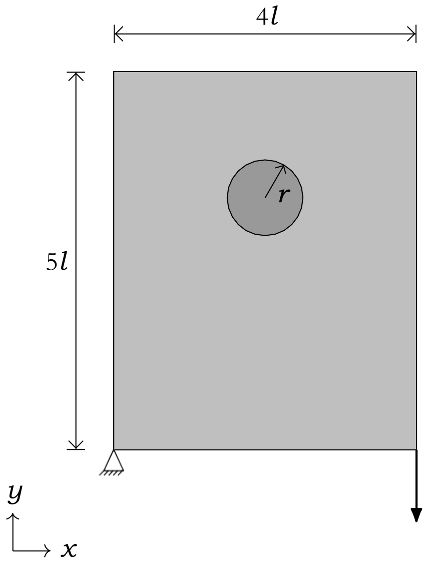
[2] The asymmetric combination of load and bearing is chosen purely for the reason that an asymmetric design is more interesting to look at. We could replace the bearing by another vertical load to create a symmetric but kinematically indeterminate problem, which our algorithm can solve thanks to the regularization of the stiffness matrix, cf. the clamped cantilever in Figure 3.10.

[3] Properly modelled symmetry would entail sliding constraints rather than a completely fixed lower boundary. Once again, our choice of boundary conditions comes down to a more interesting looking optimal design, which better illustrates certain properties of the VTS solution.

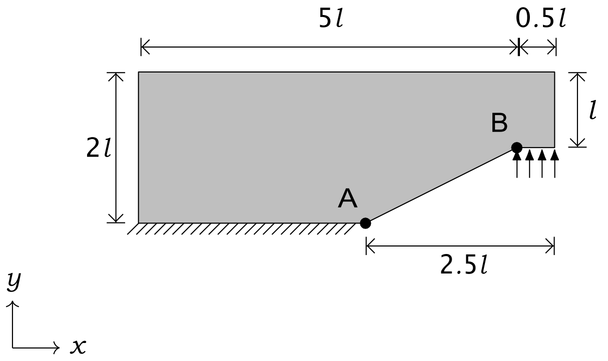
Figure 3.12: Loading scenarios and design domain for compliance minimization with SA-AMG. In three dimensions, all boundary conditions and meshes are extruded.



(a) Knee *d*-*L*: $l = 2 \cdot 2^{(L-1)}$, $r = 0.1l$. For $d = 3$, the depth of the design domain is $l/2$.



(b) Lug *d*-*L*: $l = 2^{(L-1)}$, $r = 0.5l$. For $d = 3$, the depth of the design domain is $2l$. The dark grey area is an obstacle modelled by unilateral contact constraints.



(c) Crack *d*-*L*: $l = 2^{(L-1)}$. For $d = 3$, the depth of the design domain is $2l$.

Table 3.14: Mesh dimensions of problems used in testing different aggregation strategies.

problem	m	n
Knee 2-9	194 388	390 396
Knee 3-6	176 896	560 538
Lug 2-8	406 098	814 906
Lug 3-5	214 464	680 427
Crack 2-8	465 440	932 160
Crack 3-5	236 256	738 837

adaptive mesh refinement, which leads to elements differing in their sizes by orders of magnitude. The variation in element sizes that we see in our examples is much smaller, perhaps unrealistically so. But since we are concerned mainly with the effect of AMG preconditioners on the Krylov solver's performance, this simplification can be justified by avoiding unnecessary complications in the optimization problem.

In an attempt to identify the most efficient AMG setup strategy, we propose three different approaches of employing the SA algorithm in our problems. We will test these on a two- and three-dimensional instance of each of the scenarios in Figure 3.12, as listed in Table 3.14. The first strategy is the one we have previously described: We apply SA once at the beginning of the optimization, to a solid-domain stiffness matrix. This strategy will be denoted by "K1". We can try to further improve it by availing ourselves of the adaptive SA method, varying the number of candidates that we add to the near-kernel of the transfer operator. For the second strategy, we consider the possibility that, while the cost of re-computing the transfer operators in each iteration is prohibitive, we might benefit from updating them occasionally. Whenever the solver requires more than 20 iterations for a linear system, we take this as a cue to perform SA on the stiffness matrix for the *current* densities. We refer to this strategy as "K+" and, again, we test it for different numbers of additional candidates (including zero). The third and final strategy can be seen as a combination of the previous two. We start with a transfer operator that is based on a solid-domain stiffness matrix. We set a maximum number of extra candidates but do not apply αSA just yet. Once the solver needs more than

Table 3.15: Total solver iterations for different aggregation strategies and number of extra candidates. Rows: a) Knee 2-9, b) Knee 3-6, c) Lug 2-8, d) Lug 3-5, e) Crack 2-8, f) Crack 3-5, g) average.

	K1				K+				K++			
	0	1	2	3	0	1	2	3	0	1	2	3
a)	557	288	624	624	612	527	675	674	557	478	478	478
b)	620	424	279	345	889	643	327	396	620	1059	1032	814
c)	1579	907	721	721	1568	1012	904	679	1579	1038	928	928
d)	799	747	709	730	781	790	760	650	799	664	720	838
e)	559	779	632	632	812	804	625	645	559	480	473	473
f)	669	1395	497	464	1141	789	509	597	669	625	432	431
g)	797	757	577	586	967	761	633	607	797	724	677	660

Table 3.16: Total CPU time (in minutes) for different aggregation strategies and number of extra candidates. Rows: a) Knee 2-9, b) Knee 3-6, c) Lug 2-8, d) Lug 3-5, e) Crack 2-8, f) Crack 3-5, g) average.

	K1				K+				K++			
	0	1	2	3	0	1	2	3	0	1	2	3
a)	13	18	36	36	23	55	156	118	14	24	24	26
b)	46	58	71	98	79	145	97	165	50	87	98	116
c)	68	86	121	123	339	628	1150	705	72	94	135	133
d)	76	113	153	200	109	271	549	574	84	100	129	216
e)	48	106	176	176	160	591	624	1258	49	82	140	124
f)	77	142	128	164	151	357	254	544	69	90	106	166
g)	55	87	114	133	144	341	472	561	56	80	105	130

20 iterations in a single Newton step, we add one candidate. Importantly, this is an extra candidate for the near-kernel of the *current* stiffness matrix. We add a further candidate in every subsequent Newton iteration in which the number of solver iterations exceeds 20, until the maximum number of extra candidates has been reached. This strategy will be referred to as "K++". Note that strategies K1 and K++ are equivalent if the maximum number of candidates is zero.

The results for all strategies are listed side by side in Tables 3.15 and 3.16. Table 3.15 shows the total number of solver iterations, while Table 3.16 shows the total CPU time. As one would expect, adding more candidates reduces the number of solver iterations in nearly all cases. This is true for all three strategies. On average, aggregation strategy K1 with 2 candidates seems to achieve the lowest number of solver iterations. The computational time, however, paints a different picture. An increase in the number of candidates very consistently leads to an increase in overall time. This is caused by the extra work required for each call to the αSA routines, but also by the inflated size of the coarse-level operators brought about by a larger near-kernel. For a more informed interpretation of the results, we can compare the computational time taken up by aggregation and the preparation of the multigrid V-cycle to the time required for the actual solver routines. The accumulated CPU times for the multigrid setup and solver calls, respectively, are given in Tables A.5 and A.6 in Appendix A.2. They allow us to draw the following conclusions.

As soon as we either run the smoothed aggregation more than once or adaptively add at least one candidate, more time is spent on the setup of the multigrid V-cycle than on the actual call to the preconditioned Krylov solver. The additional computational effort for any type of scheme that is adaptive, whether in the sense that extra near-kernel candidates are computed or just in the sense of updating the transfer operators throughout the optimization, appears cost-prohibitive. Discarding these strategies completely might, however, be rash, since a parallel (or simply more efficient) implementation might speed up the smoothed aggregation enough to make them viable. It is therefore worthwhile to consider the solver CPU times separately. Here, the situation is more nuanced. It appears that we achieve an overall decrease in solver time only when the number of required solver iterations drops enough to compensate for the increase in system size that is due to larger coarse-level operators.

For the problems we have considered, the simplest strategy – K1 without any additional near-kernel candidates – appears to be the most efficient. Adaptivity does not warrant the extra computational effort and performing the aggregation on any matrix other than the solid-domain stiffness matrix does not improve the solver iteration count. However, it would be interesting to try all of the proposed

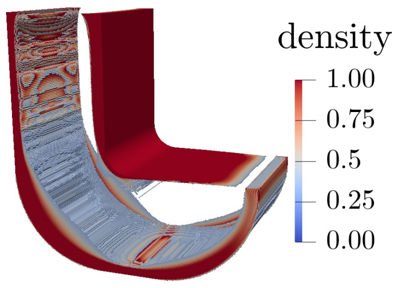
Table 3.17: Mesh dimensions and numerical results for large-scale unstructured mesh scenarios. The reference row contains the average values for a range of large-scale structured mesh problems.

			iterations		time [minutes]		
problem	m	n	Nwt	solver	total	solver	MG
Knee 3-8	6 326 656	19 285 371	89	1344	3719	1151	1820
Lug 3-6	1 481 280	4 579 380	101	821	647	195	241
Crack 3-6	1 745 216	5 353 725	103	2131	1050	425	393
reference	–	–	104	459	965	241	142

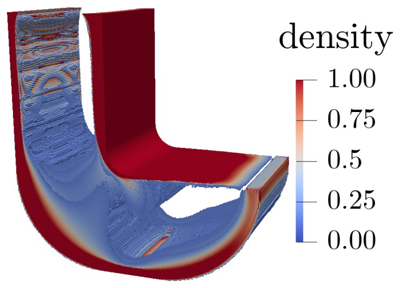
strategies in a SIMP context. Updating the transfer operators only every couple of iterations could be an alternative to the hybrid GMG-AMG preconditioner used in [96]. Furthermore, since even the standard SA method leads to a noticeable reduction of solver iterations for the SIMP problem – something we do not observe for the VTS problem – perhaps the αSA would lead to an even higher reduction, one which is enough to offset the additional computational cost.

To finish the section, we present results for each of the three scenarios on a three-dimensional mesh with more than a million elements, see Figures 3.13 to 3.15. Table 3.17 shows the problem sizes, and the iterations and times required. For comparison, we have also included an extra row which contains the average values of a wide range of large-scale structured mesh problems – more specifically, the problems Cantilever m_x-m_y-m_z-L, Bridge m_x-m_y-m_z-L and MBB m_x-m_y-m_z-L, for $m_x = 2, 3, \ldots, 8$, $m_y = m_z = 2$ and $L = 7$. We can see that, while the unstructured mesh scenarios require about the same number of Newton iterations, solving the linear systems requires a lot more iterations. Not only that, but upon closer inspection, we can see that the solver and multigrid setup make up a bigger part of the overall CPU time. Judging from our experience profiling smaller scale problems on unstructured meshes, this is due to the fact that SA transfer operators lead to denser coarse-level systems than the GMG operators.

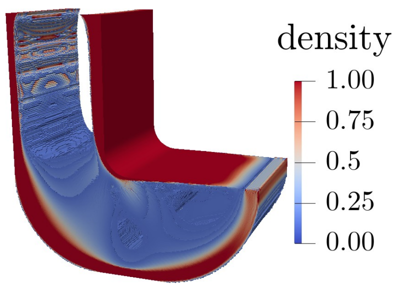
Figure 3.13: Knee 3-8 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.



(a) $c = 0.6$

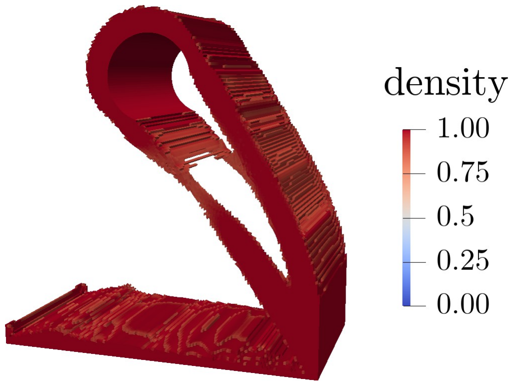


(b) $c = 0.8$

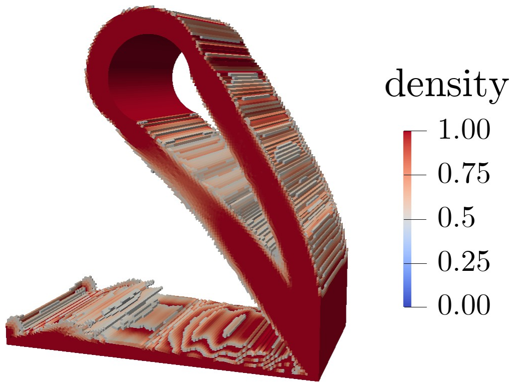


(c) $c = 0.9$

Figure 3.14: Lug 3-6 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.

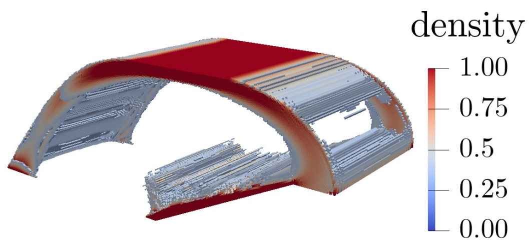


(a) $c = 0.7$

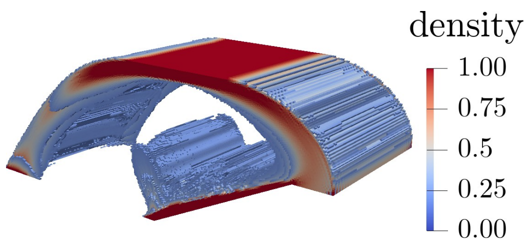


(b) $c = 0.9$

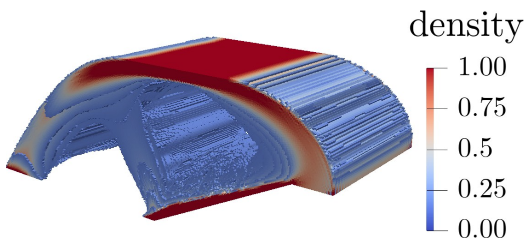
Figure 3.15: Crack 3-6 optimal design, visualized with different density thresholds so that visible elements add up to $c \cdot V$.



(a) $c = 0.5$



(b) $c = 0.7$



(c) $c = 0.8$

3.2.4 Detailed Comparison of MINRES and CG

In this section, we take a closer look at the performance of the CG and MINRES solver. So far, we have only considered the number of iterations and CPU time required over the course of an entire optimization, where the solver tolerance was quite high at 10^{-2}. A more common and rigorous numerical analysis should compare the solvers on a single linear system and with a stricter tolerance. In order to further back up our claim that both solvers, preconditioned by a geometric or algebraic multigrid method, are (equally) efficient and optimal with respect to the mesh size, we use them to solve distinct linear systems with a tolerance of 10^{-8}. These systems are taken from the respective last iterations of the PBM and IP algorithm applied to the problem MBB m_x-2-2-L, where $m_x = 2, 4, 6, 8$ and $L = 4, 5, 6$. We test both the GMG and AMG preconditioner. For the latter, we use the K1 setup strategy and the same parameters as in Section 3.2.3.

Tables 3.18 and 3.19 list the numbers of iterations and CPU times required by the

Table 3.18: Iterations required by the MG preconditioned CG and MINRES solvers for the final PBM system.

Problem dimensions		GMG		AMG	
m_x-m_y-m_z-L	$n+1$	CG	MR	CG	MR
2-2-2-4	14 417	77	71	86	84
4-2-2-4	28 289	137	133	178	172
6-2-2-4	42 161	124	119	165	164
8-2-2-4	56 033	163	157	219	212
2-2-2-5	106 657	347	313	377	349
4-2-2-5	211 201	160	150	231	215
6-2-2-5	315 745	124	118	168	162
8-2-2-5	420 289	191	170	234	220
2-2-2-6	819 521	75	68	105	98
4-2-2-6	1 630 721	109	105	174	165
6-2-2-6	2 441 921	123	103	195	183
8-2-2-6	3 253 121	231	211	364	379

different solvers for the final PBM system, along with the system size[1], for various different design domain proportions and mesh resolutions. Tables 3.20 and 3.21 show the corresponding results for the systems taken from the final IP iteration. We can see that the MINRES method requires fewer iterations than the CG method in most cases, for both types of preconditioners. The difference is often negligible for the systems stemming from the PBM method but more pronounced for IP systems. The ratio between the CG and MR iteration numbers is roughly constant over all different PBM systems and all IP systems, respectively, which indicates that the two methods scale similarly with the size of the mesh.

Since the MINRES method has a slightly larger overhead than the CG method, it usually takes a bit longer to run even when the number of iterations are slightly lower. When the difference in the iteration number becomes larger, as seen in the IP examples, this overhead is offset.

As expected, the geometric multigrid preconditioner generally performs better than the algebraic multigrid, with just a single exception to this rule. However,

[1]Recall that $S_{PBM} \in \mathbb{R}^{(n+1)\times(n+1)}$.

Table 3.19: CPU time (in seconds) required by the MG preconditioned CG and MINRES solvers for the final PBM system. Time values are rounded and do not include prolongation operator setup time.

Problem dimensions		GMG		AMG	
m_x-m_y-m_z-L	$n+1$	CG	MR	CG	MR
2-2-2-4	14 417	1	1	1	2
4-2-2-4	28 289	4	4	6	6
6-2-2-4	42 161	6	6	8	9
8-2-2-4	56 033	9	10	13	14
2-2-2-5	106 657	39	40	54	56
4-2-2-5	211 201	37	40	71	73
6-2-2-5	315 745	41	44	74	78
8-2-2-5	420 289	81	83	144	148
2-2-2-6	819 521	63	68	98	106
4-2-2-6	1 630 721	187	209	339	366
6-2-2-6	2 441 921	310	304	561	592
8-2-2-6	3 253 121	835	885	1541	1760

Table 3.20: Iterations required by the MG preconditioned CG and MINRES solvers for final IP system.

Problem dimensions		GMG		AMG	
m_x-m_y-m_z-L	$n+1$	CG	MR	CG	MR
2-2-2-4	14 417	366	335	452	407
4-2-2-4	28 289	832	718	1089	972
6-2-2-4	42 161	1383	1145	1875	1630
8-2-2-4	56 033	1185	977	1685	1471
2-2-2-5	106 657	192	154	204	159
4-2-2-5	211 201	284	167	392	265
6-2-2-5	315 745	329	272	534	479
8-2-2-5	420 289	319	228	508	408
2-2-2-6	819 521	103	82	90	61
4-2-2-6	1 630 721	115	77	134	102
6-2-2-6	2 441 921	185	127	295	119
8-2-2-6	3 253 121	237	167	416	267

Table 3.21: CPU time (in seconds) required by the MG preconditioned CG and MINRES solvers for the final IP system. Time values are rounded and do not include prolongation operator setup time.

Problem dimensions		GMG		AMG	
m_x-m_y-m_z-L	$n+1$	CG	MR	CG	MR
2-2-2-4	14 417	5	5	6	6
4-2-2-4	28 289	23	23	33	33
6-2-2-4	42 161	62	59	88	86
8-2-2-4	56 033	65	61	99	98
2-2-2-5	106 657	24	22	37	31
4-2-2-5	211 201	68	46	130	106
6-2-2-5	315 745	110	105	264	261
8-2-2-5	420 289	132	113	354	308
2-2-2-6	819 521	103	96	106	84
4-2-2-6	1 630 721	213	167	299	259
6-2-2-6	2 441 921	488	390	938	384
8-2-2-6	3 253 121	896	735	1877	1322

the AMG appears to scale the same as the GMG. In most cases, it leads to roughly 1.2–1.8 times as many iterations as the GMG preconditioner. The ratio for the CPU times is a bit larger, sometimes surpassing 2. As mentioned in previous sections, this is probably due to a higher operator complexity on coarse grid levels and thus a higher computational cost per solver iteration.

There does not appear to be a relationship between the problem dimension and the number of solver iterations, which points to an optimality with respect to the problem size of both the GMG and AMG preconditioner. One should of course be careful drawing such conclusions based on comparisons of system matrices taken from different optimization problems. The distribution of the eigenvalues of the system matrices (3.9) and (3.25) depends on the primal and dual variables as well as the barrier/penalty parameters. Their values in the final optimization iteration depend on the specific optimization problem and various algorithm parameters and thus they are nigh impossible to control or predict with any kind of accuracy. We can still gain some limited insight into the spectral properties of the system matrices, by considering the condition numbers $\mathrm{cond}(S) = \lambda_{n+1}/\lambda_1$ of the respective PBM and

IP system matrices. Using Matlab's `eigs` method for sparse matrix eigenvalues we have computed these for the cases with $L = 4, 5$ mesh levels[1]. The results are shown in Table 3.22. As many of the matrices were ill-conditioned, the precision of the results might vary but should be sufficient for a qualitative analysis.

From observations made in smaller test problems, we know that the largest eigenvalue is typically an outlier which corresponds closely to the value of the lower right (scalar) diagonal block of the matrix. The remaining eigenvalues are generally scattered more homogeneously over the range of the spectrum. We therefore also include the ratio λ_n/λ_1 of the second largest to the lowest eigenvalue.

Table 3.22: Approximate condition numbers for the final PBM and IP system matrices.

	PBM		IP	
m_x-m_y-m_z-L	λ_{n+1}/λ_1	λ_n/λ_1	λ_{n+1}/λ_1	λ_n/λ_1
2-2-2-4	2.00×10^{18}	1.38×10^{14}	5.11×10^{14}	3.30×10^{10}
4-2-2-4	3.66×10^{17}	2.17×10^{13}	2.70×10^{16}	1.02×10^{12}
6-2-2-4	2.31×10^{17}	1.58×10^{13}	1.75×10^{17}	6.88×10^{12}
8-2-2-4	2.59×10^{16}	2.14×10^{12}	2.09×10^{16}	1.38×10^{12}
2-2-2-5	9.21×10^{20}	1.19×10^{15}	1.07×10^{17}	9.27×10^{10}
4-2-2-5	8.50×10^{20}	9.58×10^{14}	1.12×10^{17}	7.36×10^{10}
6-2-2-5	3.32×10^{20}	4.37×10^{14}	4.04×10^{17}	2.73×10^{11}
8-2-2-5	4.80×10^{20}	7.74×10^{14}	3.64×10^{16}	4.99×10^{10}

The condition number (at least when discounting the largest eigenvalue) for one type of method and the same mesh resolution varies by up to two orders of magnitude. When comparing the values for different mesh resolutions, the variation is even higher. Still, there is no obvious correlation between these condition numbers and the number of solver iterations. This is further evidence that our multigrid preconditioners are spectrally equivalent to the system matrices encountered in the optimization problems. Interestingly, the IP system matrices generally have lower condition numbers than the PBM matrices; they even appear to grow more rapidly

[1]We have not included any results for $L = 6$ due to time- and memory-limitations. The eigenvalue computation for the two largest matrices included in Table 3.22 required over 128 GB RAM and around 13 and 15 hours on BlueBEAR [24], respectively.

with increasing mesh resolution in the latter case. Still, both Krylov solvers perform better on the PBM systems. A possible explanation could be that the PBM method leads to a more favourable clustering of eigenvalues in the system matrices, although this is only a speculation at this point.

One could easily devote more computational time and many more pages to the analysis of our Krylov solvers' performance and its relation to the size of the optimization problem, the spectrum of the system matrices and various algorithm parameters. The objective of this section was only to provide some evidence for the claim that our multigrid preconditioners are spectrally equivalent. Indeed, if our preconditioners were sub-optimal, leading in general to larger numbers of solver iterations for larger problem sizes, this trend would in all likelihood have been revealed in the test cases we have presented here. We can therefore say with some confidence that our solver setup seems to be optimal with respect to the resolution of the FE mesh used in the optimization problem.

3.3 A Note on Density Penalization

It has been very clearly demonstrated that problem (2.48) can be solved far more efficiently by both the IP and PBM method proposed in this thesis than by the OC method. From an optimization point of view, this is not surprising, as the latter is a first order method, while the former also use second order derivative information. Other first order methods, such as the MMA – which is even more prevalent in topology optimization than the OC method – can be expected to fare equally poorly in a comparison. However, we have so far only considered the VTS formulation, which often produces optimal designs with large grey areas. It would be negligent not to address the limitations of our approach with regards to the SIMP formulation, as it is the standard in topology optimization and much more common than the VTS formulation. Designs based on SIMP are more "black and white" and thus more viable for manufacturing. This is achieved by a penalization of intermediate density values, such that the material cost of a grey element is disproportionately larger than its stiffness. A classic example of this is the polynomial penalization

$$\mathbf{0} \le \boldsymbol{\rho} \; 1 \to \mathrm{K}(\boldsymbol{\rho}) = \sum_{i=1}^{m} \rho_i^q \mathrm{K}_i + \mathrm{K}_0 \in \mathbb{R}^{n\times n}, \tag{3.30}$$

where $q > 2$, see for example [22]. The matrix K_0 ensures $\mathrm{K}(\boldsymbol{\rho}) \; 0$ and corresponds to a lower bound Young modulus of, e.g., $E_{\min} = 10^{-9}$. More recently, the relaxed Heaviside projection [126] has gained popularity as a penalization function, see for example [50]. For simplicity's sake, we will stick with (3.30) for the following considerations. It is well known that the SIMP approach on its own leads to so-called "checkerboarding" [43], a numerical artefact where an alternating pattern of void and solid elements yields a large overall stiffness in the FE analysis. This can be prevented, for instance, by a filtering of the design variables [35, 26], leading to a redefinition of the physical densities as

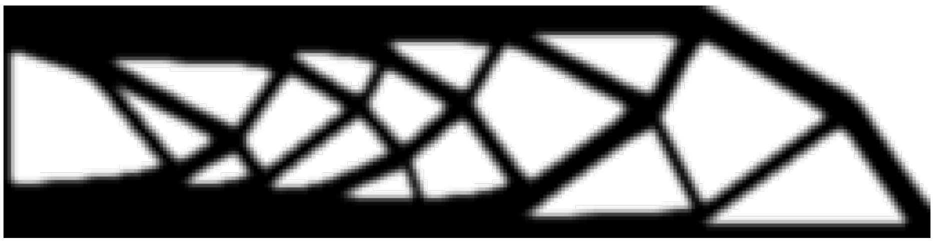
$$\tilde{\rho}_i := \frac{\sum_j w_{ij}\rho_j}{\sum_j w_{ij}}, \; i = 1, \ldots, m. \tag{3.31}$$

Here, $w_{ij} = \max\{0, r_W - \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$, where $\boldsymbol{x}_i$ is the geometric centre of the ith element, $r_W > 0$ is the filter radius and dist is some distance measure. We will use $\operatorname{dist}(\cdot) = \|\cdot\|_1$, for ease of implementation. Since (3.31) is a linear expression in $\boldsymbol{\rho}$, we can write it as $\tilde{\boldsymbol{\rho}} = \mathrm{W}\boldsymbol{\rho}$, where $\mathrm{W} = [w_{ij}]_{i,j=1,\ldots,m}$. Together with (3.30), we thus obtain the SIMP version of the minimum compliance problem:

$$\begin{aligned}
\min_{\boldsymbol{\rho}\in\mathbb{R}^m, \boldsymbol{u}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{f}^\top \boldsymbol{u} \\
\text{s.t.} \quad & \mathrm{K}(\mathrm{W}\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}, \\
& (\mathrm{W}\boldsymbol{\rho})^\top \boldsymbol{a} = V, \\
& \underline{\boldsymbol{\rho}} \le \boldsymbol{\rho} \le \overline{\boldsymbol{\rho}}, \quad i = 1, \ldots, m\,.
\end{aligned} \tag{3.32}$$

For a typical example of the kind of design one obtains for this problem, see Figure 3.16.

Figure 3.16: MBB 8-2-0-6 solution for the SIMP formulation with a filter radius of $r_W = 2.4$, cf. Figure 3.4.



The first problem that arises when we attempt to apply the algorithms we used for (2.48) to (3.32) is that the latter is not a convex problem. Our PBM method previously relied on the existence of a dual problem in a closed form which satisfied strict duality. While it is possible in theory to apply the PBM approach to (3.32), we run into complications due to the fact that it does not guarantee strict feasibility of the iterates for $\boldsymbol{\rho}$. The definition of the stiffness mapping (3.30) has to be extended to cover the negative real numbers, and it needs to be done in such a way that $K(\boldsymbol{\rho})\ 0$ even if $\rho_i < 0$ for some i. We can achieve this by replacing ρ^q_i with $\max\{0, \rho^q\}$, which is twice continuously differentiable for $q \geq 3$. While this satisfies $K(\boldsymbol{\rho})\ 0$ for all $\boldsymbol{\rho} \in \mathbb{R}^m$, the resulting algorithm struggles to converge in practice. The cause of this seems to be that negative densities effectively raise the volume limit. At the same time, density values larger than the upper bound lead to an improved local stiffness. As a consequence, the algorithm can stagnate in local minima of the augmented Lagrangian that lie outside the feasible set of (3.32). It is possible that the iterates will be pushed closer to or into the feasible set once the penalty parameters have become small enough to flatten out these local minima, but this kind of convergence behaviour is anything but ideal.

The IP method, on the other hand, naturally maintains strict feasibility throughout all iterations. And while a modified version of Algorithm 4 can indeed solve (3.32), it does so very slowly. The reason for this is a denser system matrix caused by the density filtering. Regard once more the matrices (3.8) and (3.9) which we obtained in the derivation of our IP algorithm. If we follow the same approach for problem (3.32), we eventually arrive at the system matrix

$$\begin{bmatrix} K(W\boldsymbol{\rho}) & 0 & B(\boldsymbol{u})W \\ 0 & 0 & \boldsymbol{a}^\top W \\ W^\top B(\boldsymbol{u})^\top & W^\top \boldsymbol{a} & -D_{IP} + \nabla^2_{\boldsymbol{\rho}}\left(\frac{1}{2}\boldsymbol{u}^\top K \boldsymbol{u}\right) \end{bmatrix}, \tag{3.33}$$

where $B(\boldsymbol{u}) := q\left[\tilde{\rho}_1^{q-1}K_1\boldsymbol{u}, \dots, \tilde{\rho}_m^{q-1}K_m\boldsymbol{u}\right]$. This matrix closely resembles (3.8), but there are two important differences. The first one is that, instead of just the matrix $B(\boldsymbol{u})$, which has the same sparsity structure as $B(\boldsymbol{u})$ in (3.8), we now have a "filtered" version of it in the off-diagonal blocks. The second difference is the bottom right block, which is no longer a negative definite diagonal matrix. Since this was crucial to our original approach, as it allowed us to easily invert the block and maintain a certain sparsity structure, we will drop the second derivative of the energy term[1]. This is essentially a local convexification of the problem. We can now proceed as in Section 3.1.1, taking the Schur complement. We obtain a matrix that is similar to the IP system matrix (3.9):

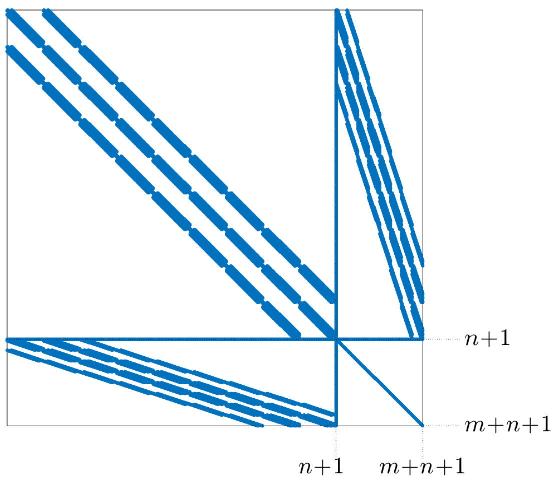
$$\begin{bmatrix} K(\boldsymbol{\rho}) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B(\boldsymbol{u})W \\ \boldsymbol{a}^\top W \end{bmatrix} D_{IP}^{-1} \begin{bmatrix} W^\top B(\boldsymbol{u})^\top & W^\top \boldsymbol{a} \end{bmatrix}. \tag{3.34}$$

Because of the filter matrix that appears in the upper left block of this matrix, its sparsity structure is not the same as that of the stiffness matrix. The "blurring" of the densities also creates a blurring of the sensitivities. This is illustrated in Figure 3.17.

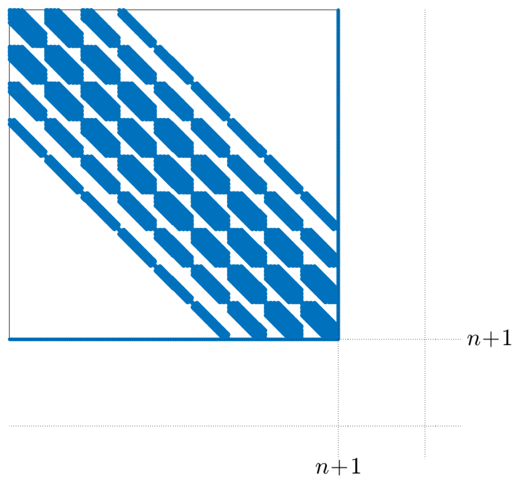
To give an example of the impact which the compromised sparsity of the system matrix has on the IP method's performance, let us consider problem MBB 8-2-0-6. Using a filter radius of $r_W = 2.4$, we solved problem (3.32) with the OC method as well as the IP method. Running both on a 2015 MacBook Pro with a 2.8 GHz quad-core processor and 16 GB of memory, the average time required to solve a

[1]This approach was also used in [80].

Figure 3.17: Sparsity structure of the IP system matrices when using the SIMP formulation ($r_W = 2$), for a three-dimensional uniform FE mesh with 512 elements; cf. Figure 3.1.



(a) (3.33)



(b) (3.34)

linear system in the OC method was $1.47 \cdot 10^{-2}$ seconds, while for the IP method it was $7.2 \cdot 10^{-2}$. If we take into account that one role of the filter radius is to set a lower bound for the width of the members in the final design, it is easy to see that the difference in computational efficiency will only be exacerbated for large-scale problems: If we want to maintain the same minimal member width on FE meshes of increasing resolution the number of neighbouring elements within the filter radius increases. This will lead to more coupling of DOFs, which is reflected by the off-diagonal terms in the system matrix. Considering three-dimensional problems – and thus three-dimensional neighbourhoods – further adds to this problem.

The above results are consistent with observations made by Rojas-Labanda and Stolpe in [109], where an IP method using Hessian data performed very well in terms of the precision of the results and the number of iterations, but was not competitive in terms of CPU time. Among the algorithms tested in that paper, the OC method turned out to be one of the most efficient for the minimum compliance problem.

CHAPTER 4

CONCLUSIONS

In this thesis, we proposed an interior point (IP) method and a penalty-barrier multiplier (PBM)/nonlinear rescaling method for variable thickness sheet (VTS) compliance minimization. With the primary focus on large-scale optimization and the use of multigrid-preconditioned Krylov solvers, we performed a reduction of the Newton system matrix in order to use standard multigrid transfer operators. While both optimization methods clearly outperform the well-known optimality criteria (OC) method on medium-scale problems, requiring fewer iterations and less CPU time, the PBM method is much more robust and efficient than the IP method for large-scale problems with more than a million finite elements. Furthermore, since it solves the dual of the VTS problem, unilateral contact constraints can be easily integrated, although this is a feature of the problem formulation rather than the algorithm.

Regarding an extension to the SIMP formulation, the PBM methodology has several inherent limitations which make it unviable. While the IP algorithm can solve the SIMP-based compliance minimization problem with just a few adjustments to our approach, it is no longer competitive due to the compromised sparsity of the system matrices. This confirms previous results in the literature. One potential way to reduce the computational cost of this approach could be to replace the preconditioner. Instead of using a multigrid approximation of the inverse of the system matrix S, one could use a multigrid approximation of the inverse of a different matrix, say $\tilde{\mathsf{S}}$, which is spectrally equivalent but has a more tractable sparsity structure. In our implementation, a considerable part of the total CPU time goes into the construction of the system matrix, on the one hand, and into the setup of and subsequent matrix-vector products involving the coarse-level operators, on the other. All of these processes become more costly as the sparsity of S decreases. Using a sparser $\tilde{\mathsf{S}}$ would not only reduce the complexity of the multigrid method, but would eliminate the necessity to explicitly construct the matrix S: Rather than performing a series of matrix-matrix multiplications, one could then replace the operation $\mathsf{S}\boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^{n+1}$, by a series of matrix-vector multiplications in each Krylov solver iteration. This by itself could be more efficient, although by how much would need to be tested. A possible next step would be to use a matrix-free solver on a GPU, see for example [116], which could remove the necessity of assembling even the stiffness matrix. This would however considerably complicate the implementation, in particular since the assembly of the system matrix S is more involved than that of only the stiffness matrix, which is usually the only one required in topology optimization algorithms. The success of the whole approach further hinges on the choice of $\tilde{\mathsf{S}}$, for which we can currently make no informed suggestions.

Building on the promising results of our approach using a geometric multigrid (GMG) preconditioner, we have replaced it by the smoothed aggregation (SA) method, a type of algebraic multigrid (AMG) technique. In contrast to observations made for SIMP-based problems elsewhere in the literature, the AMG preconditioner does not improve the performance of the Krylov solver on uniform structured meshes when GMG can be used. This is presumably due to a less pronounced local material anisotropy in the VTS solution. We have therefore concentrated on unstructured mesh problems, where GMG is not applicable. Rather than applying AMG as a black box preconditioner in each solver iteration, we have proposed and tested several different setup strategies based on reusing transfer operators. We have found the most efficient strategy to be that which constructs transfer operators by performing SA only once on a solid-design stiffness matrix and uses these operators throughout the entire optimization. Expanding the range of the prolongation operators by means of adaptive smoothed aggregation lowers the overall number of solver iterations, but is on the whole less time-efficient due to the larger size of the coarse-level system matrices.

Any of the proposed SA setup strategies could be worthwhile applying to problems based on the SIMP formulation. We suggest a study similar to that in [96], using an established topology optimization algorithm such as the OC method or the method of moving asymptotes. However, rather than just comparing out-of-the-box GMG and AMG techniques, we conjecture that reusing transfer operators for several optimization iterations could decrease the oftentimes prohibitive overhead of the AMG while still providing a considerable reduction in solver iterations. Such an approach might therefore prove to be a better compromise than the GMG-AMG hybrid used in [96] or [3]. Furthermore, adaptive schemes should also not be discounted. The AMG's capability to handle strong material anisotropy appears to have a bigger impact when using an AMG preconditioner for systems arising in SIMP-based problems than it does in the VTS case. This important property can be reinforced by extending the range of the transfer operators using the adaptive smoothed aggregation method. Perhaps this could improve the AMG enough to justify the increased overhead that comes with larger coarse-level system sizes. Finally, instead of only reusing the transfer operators, we could go one step further and reuse the coarse-grid operators, as has previously been done in [8]. Since the construction of the coarse-level system matrices contributes a large portion of the computational work in each iteration, such an approach could considerably reduce the computation time. However, we expect that coarse-grid operators would need to be updated much more frequently than the transfer operators and this would warrant a detailed study.

BIBLIOGRAPHY

[1] N. Aage, E. Andreassen, and B. S. Lazarov. “Topology optimization using PETSc: an easy-to-use, fully parallel, open source topology optimization framework”. In: *Structural and Multidisciplinary Optimization* 51.3 (Aug. 2014), pp. 565–572.

[2] N. Aage and B. S. Lazarov. “Parallel framework for topology optimization using the method of moving asymptotes”. In: *Structural and Multidisciplinary Optimization* 47.4 (2013), pp. 493–505.

[3] N. Aage et al. “Giga-voxel computational morphogenesis for structural design”. In: *Nature* 550.7674 (Oct. 2017), pp. 84–86.

[4] W. Achtziger. “Optimierung von einfach und mehrfach belasteten Stabwerken”. PhD thesis. Universität Bayreuth, 1993.

[5] W. Achtziger. “Multiple-load truss topology and sizing optimization: some properties of minimax compliance”. In: *Journal of Optimization Theory and Applications* 98.2 (1998), pp. 255–280.

[6] W. Achtziger et al. “Equivalent displacement based formulations for maximum strength truss topology design”. In: *IMPACT of Computing in Science and Engineering* 4.4 (1992), pp. 315–345.

[7] G. Allaire, F. Jouve, and A.-M. Toader. “Structural optimization using sensitivity analysis and a level-set method”. In: *Journal of Computational Physics* 194 (2004), pp. 363–393.

[8] O. Amir. “Revisiting approximate reanalysis in topology optimization: on the advantages of recycled preconditioning in a minimum weight procedure”. In: *Structural and Multidisciplinary Optimization* 51.1 (2015), pp. 41–57.

[9] O. Amir, N. Aage, and B. S. Lazarov. “On multigrid-CG for efficient topology optimization”. In: *Structural and Multidisciplinary Optimization* 49.5 (May 2014), pp. 815–829.

[10] O. Amir, M. P. Bendsøe, and O. Sigmund. “Approximate reanalysis in topology optimization”. In: *International Journal for Numerical Methods in Engineering* 78.12 (2009), pp. 1474–1491.

[11] O. Amir, M. Stolpe, and O. Sigmund. “Efficient use of iterative solvers in nested topology optimization”. In: *Structural and Multidisciplinary Optimization* 42.1 (2010), pp. 55–72.

[12] E. Andreassen et al. “Efficient topology optimization in MATLAB using 88 lines of code”. In: *Structural and Multidisciplinary Optimization* 43.1 (2011), pp. 1–16.

[13] A. El-Bakry et al. “On the formulation and theory of the Newton interior-point method for nonlinear programming”. In: *Journal of Optimization Theory and Applications* 89.3 (1996), pp. 507–541.

[14] S. Bellavia. “Inexact interior-point method”. In: *Journal of Optimization Theory and Applications* 96.1 (1998), pp. 109–121.

[15] A. Ben-Tal, B. Yuzefovich, and M. Zibulevsky. “Penalty-barrier multipliers methods for minimax and constrained smooth convex optimization”. In: *Optimization Laboratory, Technion, Israel. Research Report* (1992), pp. 9–92.

[16] A. Ben-Tal and M. P. Bendsøe. “A new method for optimal truss topology design”. In: *SIAM Journal on Optimization* 3.2 (1993), pp. 322–358.

[17] A. Ben-Tal and M. Zibulevsky. “Penalty/barrier multiplier methods for convex programming problems”. In: *SIAM Journal on Optimization* 7.2 (1997), pp. 347–366.

[18] M. P. Bendsøe. “Optimal shape design as a material distribution problem”. In: *Structural Optimization* 1.4 (1989), pp. 193–202.

[19] M. P. Bendsøe and O. Sigmund. “Material interpolation schemes in topology optimization”. In: *Archive of Applied Mechanics* 69.9 (1999), pp. 635–654.

[20] M. P. Bendsøe, A. Díaz, and N. Kikuchi. “Topology and generalized layout optimization of elastic structures”. In: *Topology design of structures*. Springer, 1993, pp. 159–205.

[21] M. P. Bendsøe and N. Kikuchi. “Generating optimal topologies in structural design using a homogenization method”. In: *Computer Methods in Applied Mechanics and Engineering* 71.2 (1988), pp. 197–224.

[22] M. P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

[23] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic Press, 2014.

[24] *BlueBEAR High Performance Computing service*. July 2021. url: `http://www.birmingham.ac.uk/bear`.

[25] T. Borrvall and J. Petersson. “Large-scale topology optimization in 3D using parallel computing”. In: *Computer Methods in Applied Mechanics and Engineering* 190.46 (2001), pp. 6201–6229.

[26] B. Bourdin. “Filters in topology optimization”. In: *International Journal for Numerical Methods in Engineering* 50.9 (2001), pp. 2143–2158.

[27] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[28] D. Brackett, I. Ashcroft, and R. Hague. “Topology optimization for additive manufacturing”. In: *Proceedings of the solid freeform fabrication symposium, Austin, TX*. Vol. 1. 2011, pp. 348–362.

[29] A. Brandt. “Multi-level adaptive solutions to boundary-value problems”. In: *Mathematics of Computation* 31.138 (1977), pp. 333–390.

[30] A. Brandt. "Algebraic multigrid theory: the symmetric case". In: *Applied Mathematics and Computation* 19.1-4 (1986), pp. 23–56.

[31] M. G. Breitfeld and D. F. Shanno. "Computational experience with penalty-barrier methods for nonlinear programming". In: *Annals of Operations Research* 62.1 (1996), pp. 439–463.

[32] M. Brezina et al. "Adaptive smoothed aggregation (αSA) multigrid". In: *SIAM Review* 47.2 (2005), pp. 317–346.

[33] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial (2Nd Ed.)* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[34] A. Brune and M. Kočvara. "On barrier and modified barrier multigrid methods for three-dimensional topology optimization". In: *SIAM Journal on Scientific Computing* 42.1 (2020), A28–A53.

[35] T. E. Bruns and D. A. Tortorelli. "Topology optimization of non-linear elastic structures and compliant mechanisms". In: *Computer Methods in Applied Mechanics and Engineering* 190.26-27 (2001), pp. 3443–3459.

[36] S. Cafieri et al. "Stopping criteria for inner iterations in inexact potential reduction methods: a computational study". In: *Computational Optimization and Applications* 36.2 (2007), pp. 165–193.

[37] M. Cavazzuti et al. "High performance automotive chassis design: a topology optimization based approach". In: *Structural and Multidisciplinary Optimization* 44.1 (2011), pp. 45–56.

[38] J. Cea and K. Malanowski. "An example of a max-min problem in partial differential equations". In: *SIAM Journal on Control* 8.3 (1970), pp. 305–316.

[39] V. Challis, A. Roberts, and J. Grotowski. "High resolution topology optimization using graphics processing units (GPUs)". In: *Structural and Multidisciplinary Optimization* 49 (2014), pp. 315–325.

[40] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[41] A. Clausen. "Topology optimization for additive manufacturing". PhD thesis. Department of Mechanical Engineering, Technical University of Denmark, 2016.

[42] N. P. van Dijk et al. "Level-set methods for structural topology optimization: a review". In: *Structural and Multidisciplinary Optimization* 48.3 (2013), pp. 437–472.

[43] A. Díaz and O. Sigmund. "Checkerboard patterns in layout optimization". In: *Structural Optimization* 10.1 (1995), pp. 40–45.

[44] G. Duvaut and J.-L. Lions. *Les inéquations en mécanique et en physique*. Dunod, Paris, 1972.

[45] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation, 2014.

[46] A. Evgrafov et al. "Large-scale parallel topology optimization using a dual-primal substructuring solver". In: *Structural and Multidisciplinary Optimization* 36.4 (2008), pp. 329–345.

[47] R. D. Falgout. "An introduction to algebraic multigrid". In: *IEEE Annals of the History of Computing* 8.06 (2006), pp. 24–33.

[48] E. A. Fancello. "Topology optimization for minimum mass design considering local failure constraints and contact boundary conditions". In: *Structural and Multidisciplinary Optimization* 32.3 (2006), pp. 229–240.

[49] F. Fernandez et al. "Topology optimization of multiple deformable bodies in contact with large deformations". In: *Computer Methods in Applied Mechanics and Engineering* 371 (2020), p. 113288.

[50] F. Ferrari and O. Sigmund. "A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D". In: *Structural and Multidisciplinary Optimization* 62.4 (2020), pp. 2211–2228.

[51] A. V. Fiacco and G. P. McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. SIAM, 1990.

[52] A. Forsgren, P. E. Gill, and M. H. Wright. “Interior methods for nonlinear optimization”. In: *SIAM Review* 44.4 (2002), pp. 525–597.

[53] R. W. Freund and F. Jarre. “A QMR-based interior-point algorithm for solving linear programs”. In: *Mathematical Programming* 76.1 (Jan. 1997), pp. 183–210.

[54] C. Geuzaine and J.-F. Remacle. “Gmsh: a 3-D finite element mesh generator with built-in pre-and post-processing facilities”. In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331.

[55] P. E. Gill et al. “On projected Newton barrier methods for linear programming and an equivalence to Karmarkar’s projective method”. In: *Mathematical Programming* 36.2 (1986), pp. 183–209.

[56] D. Goldfarb et al. “A modified barrier-augmented Lagrangian method for constrained minimization”. In: *Computational optimization and applications* 14.1 (1999), pp. 55–74.

[57] N. I. Gould et al. “Superlinear convergence of primal-dual interior point algorithms for nonlinear programming”. In: *SIAM Journal on Optimization* 11.4 (2001), pp. 974–1002.

[58] A. Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[59] I. Griva and R. A. Polyak. “Primal–dual nonlinear rescaling method with dynamic scaling parameter update”. In: *Mathematical Programming* 106.2 (2006), pp. 237–259.

[60] I. Griva and R. A. Polyak. “1.5-q-superlinear convergence of an exterior-point method for constrained optimization”. In: *Journal of Global Optimization* 40.4 (2008), pp. 679–695.

[61] W. Hackbusch. *Multi-grid methods and applications*. Springer, 1985.

[62] R. T. Haftka and Z. Gürdal. *Elements of structural optimization*. Vol. 11. Springer Science & Business Media, 2012.

[63] W. Hemp and H. Chan. *Optimum structures*. Tech. rep. Cranfield University, 1965.

[64] M. R. Hestenes and E. Stiefel. “Methods of conjugate gradients for solving linear systems”. In: *Journal of Research of the National Bureau of Standards* 49 (1952), pp. 409–436.

[65] D. Hilding, A. Klarbring, and J. Petersson. “Optimization of structures in unilateral contact”. In: *Applied Mechanics Reviews* 52.4 (1999), pp. 139–160.

[66] M. Hinze and A. Rösch. “Discretization of optimal control problems”. In: *Constrained optimization and optimal control for partial differential equations*. Springer, 2012, pp. 391–430.

[67] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I*. Springer Berlin, 1993.

[68] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2004.

[69] I. Hlavácek et al. *Solution of variational inequalities in mechanics*. Vol. 66. Springer Science & Business Media, 2012.

[70] R. H. W. Hoppe, S. I. Petrova, and V. Schulz. “Primal-dual newton-type interior-point method for topology optimization”. In: *Journal of Optimization Theory and Applications* 114.3 (2002), pp. 545–571.

[71] F. Jarre, M. Kočvara, and J. Zowe. “Optimal truss design by interior-point methods”. In: *SIAM Journal on Optimization* 8.4 (1998), pp. 1084–1107.

[72] N. Karmarkar. “A new polynomial-time algorithm for linear programming”. In: *Proceedings of the sixteenth annual ACM symposium on theory of computing*. 1984, pp. 302–311.

[73] N. Kikuchi and J. T. Oden. *Contact problems in elasticity: a study of variational inequalities and finite element methods*. SIAM, 1988.

[74] U. Kirsch, M. Kočvara, and J. Zowe. “Accurate reanalysis of structures by a preconditioned conjugate gradient method”. In: *International Journal for Numerical Methods in Engineering* 55.2 (2002), pp. 233–251.

[75] A. Klarbring, J. Petersson, and M. Rönnqvist. “Truss topology optimization including unilateral contact”. In: *Journal of Optimization Theory and Applications* 87.1 (1995), pp. 1–31.

[76] M. Kočvara and M. Stingl. “PENNON: a code for convex nonlinear and semidefinite programming”. In: *Optimization Methods and Software* 18.3 (2003), pp. 317–333.

[77] M. Kočvara and M. Stingl. “On the solution of large-scale SDP problems by the modified barrier method using iterative solvers”. In: *Mathematical Programming* 109.2–3 (2007), pp. 413–444.

[78] M. Kočvara and M. Stingl. “Truss topology design by conic linear optimization”. In: *Advances and Trends in Optimization with Engineering Applications*. Ed. by T. Terlaky, M. Anjos, and S. Ahmed. SIAM, 2017. Chap. 11.

[79] M. Kočvara, M. Zibulevsky, and J. Zowe. “Mechanical design problems with unilateral contact”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 32.3 (1998), pp. 255–281.

[80] M. Kočvara and S. Mohammed. “Primal-dual interior point multigrid method for topology optimization”. In: *SIAM Journal on Scientific Computing* 38.5 (2016), B685–B709.

[81] M. Kružík and T. Roubíček. *Mathematical methods in continuum mechanics of solids*. Springer, 2019.

[82] M. Lawry and K. Maute. “Level set shape and topology optimization of finite strain bilateral contact problems”. In: *International Journal for Numerical Methods in Engineering* 113.8 (2018), pp. 1340–1369.

[83] B. S. Lazarov. “Topology optimization using multiscale finite element method for high-contrast media”. In: *International Conference on Large-Scale Scientific Computing*. Ed. by I. Lirkov, S. Margenov, and J. Waśniewski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 339–346.

[84] G. Leugering et al. *Constrained optimization and optimal control for partial differential equations*. Vol. 160. Springer Science & Business Media, 2012.

[85] B. Maar and V. Schulz. “Interior point multigrid methods for topology optimization”. In: *Structural and Multidisciplinary Optimization* 19.3 (May 2000), pp. 214–224.

[86] N. D. Mankame and G. Ananthasuresh. “Topology optimization for synthesis of contact-aided compliant mechanisms using regularized contact modeling”. In: *Computers & structures* 82.15-16 (2004), pp. 1267–1290.

[87] A. G. M. Michell. “The limits of economy of material in frame-structures”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 8.47 (1904), pp. 589–597.

[88] S. G. Nash, R. A. Polyak, and A. Sofer. “A numerical comparison of barrier and modified barrier methods for large-scale bound-constrained optimization”. In: *Large Scale Optimization*. Springer, 1994, pp. 319–338.

[89] J. Necas and I. Hlavácek. *Mathematical theory of elastic and elasto-plastic bodies: an introduction*. Elsevier, 2017.

[90] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[91] T. H. Nguyen et al. “A computational paradigm for multiresolution topology optimization (MTOP)”. In: *Structural and Multidisciplinary Optimization* 41 (2010), pp. 525–539.

[92] T. H. Nguyen et al. “Improving multiresolution topology optimization via multiple discretizations”. In: *International Journal for Numerical Methods in Engineering* 92 (2012), pp. 507–530.

[93] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[94] J. T. Oden and J. N. Reddy. *An introduction to the mathematical theory of finite elements*. Courier Corporation, 2012.

[95] C. C. Paige and M. A. Saunders. “Solution of sparse indefinite systems of linear equations”. In: *SIAM Journal on Numerical Analysis* 12.4 (1975), pp. 617–629.

[96] D. Peetz and A. Elbanna. “On the use of multigrid preconditioners for topology optimization”. In: *Structural and Multidisciplinary Optimization* 63.2 (2021), pp. 835–853.

[97] J. Petersson. “On stiffness maximization of variable thickness sheet with unilateral contact”. In: *Quarterly of Applied Mathematics* 54.3 (1996), pp. 541–550.

[98] J. Petersson. “A finite element analysis of optimal variable thickness sheets”. In: *SIAM Journal on Numerical Analysis* 36.6 (1999), pp. 1759–1778.

[99] J. Petersson and M. Patriksson. “Topology optimization of sheets in contact by a subgradient method”. In: *International Journal for Numerical Methods in Engineering* 40.7 (1997), pp. 1295–1321.

[100] R. A. Polyak. “Modified barrier functions: theory and methods”. In: *Mathematical Programming* 54.1-3 (1992), pp. 177–222.

[101] R. A. Polyak. “Primal–dual exterior point method for convex optimization”. In: *Optimisation Methods and Software* 23.1 (2008), pp. 141–160.

[102] R. A. Polyak. “On the local quadratic convergence of the primal–dual augmented lagrangian method”. In: *Optimization Methods and Software* 24.3 (2009), pp. 369–379.

[103] R. A. Polyak and M. Teboulle. “Nonlinear rescaling and proximal-like methods in convex optimization”. In: *Mathematical Programming* 76.2 (Feb. 1997), pp. 265–284.

[104] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*. Vol. 37. Springer Science & Business Media, 2010.

[105] R. T. Rockafellar. “The multiplier method of Hestenes and Powell applied to convex programming”. In: *Journal of Optimization Theory and applications* 12.6 (1973), pp. 555–562.

[106] R. T. Rockafellar. “Augmented Lagrange multiplier functions and duality in nonconvex programming”. In: *SIAM Journal on Control* 12.2 (1974), pp. 268–285.

[107] R. T. Rockafellar. “Augmented Lagrangians and applications of the proximal point algorithm in convex programming”. In: *Mathematics of Operations Research* 1.2 (1976), pp. 97–116.

[108] R. T. Rockafellar. *Convex analysis*. Vol. 28. Princeton University Press, 1970.

[109] S. Rojas-Labanda and M. Stolpe. “Benchmarking optimization solvers for structural topology optimization”. In: *Structural and Multidisciplinary Optimization* 52.3 (2015), pp. 527–547.

[110] M. Rossow and J. Taylor. “A finite element method for the optimal design of variable thickness sheets”. In: *AIAA Journal* 11.11 (1973), pp. 1566–1569.

[111] G. I. N. Rozvany and M. Zhou. “The COC algorithm, part I: cross-section optimization or sizing”. In: *Computer Methods in Applied Mechanics and Engineering* 89.1-3 (1991), pp. 281–308.

[112] G. I. N. Rozvany and T. Lewiński. *Topology optimization in structural and continuum mechanics*. Springer Vienna, 2014.

[113] J. W. Ruge and K. Stüben. “Algebraic multigrid”. In: *Multigrid Methods*. SIAM, 1987, pp. 73–130.

[114] Y. Saad. *Iterative methods for sparse linear systems*. Vol. 82. SIAM, 2003.

[115] Y. Saad. “Iterative methods for linear systems of equations: a brief historical journey”. In: *75 Years of Mathematics of Computation*. Vol. 754. Contemporary Mathematics, 2020.

[116] S. Schmidt and V. Schulz. “A 2589 line topology optimization code written for the graphics card”. In: *Computing and Visualization in Science* 14.6 (Aug. 2011), pp. 249–256.

[117] V. Schulz and G. Wittum. “Transforming smoothers for PDE constrained optimization problems”. In: *Computing and Visualization in Science* 11.4 (2008), pp. 207–219.

[118] J. R. Shewchuk. *An introduction to the Conjugate Gradient method without the agonizing pain*. Tech. rep. Pittsburgh, PA, USA, 1994.

[119] R. Stainko. “Advanced Multilevel Techniques Advanced Multilevel Techniques to Topology Optimization”. PhD thesis. Johannes Kepler Universität Linz, 2006.

[120] N. Strömberg and A. Klarbring. “Topology optimization of structures in unilateral contact”. In: *Structural and Multidisciplinary Optimization* 41.1 (2010), pp. 57–64.

[121] K. Svanberg. “A class of globally convergent optimization methods based on conservative convex separable approximations”. In: *SIAM Journal on Optimization* 12.2 (2002), pp. 555–573.

[122] M. A. S. de Troya et al. *Another source of mesh dependence in topology optimization*. 2021. arXiv: `2106.12098 [math.OC]`.

[123] P. Vaněk, M. Brezina, and J. Mandel. “Convergence of algebraic multigrid based on smoothed aggregation”. In: *Numerische Mathematik* 88.3 (May 2001), pp. 559–579.

[124] P. Vaněk, J. Mandel, and M. Brezina. “Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems”. In: *Computing* 56.3 (1996), pp. 179–196.

[125] K. Vemaganti and W. E. Lawrence. “Parallel methods for optimality criteria-based topology optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 194.34-35 (2005), pp. 3637–3667.

[126] F. Wang, B. S. Lazarov, and O. Sigmund. “On projection methods, convergence and robust formulations in topology optimization”. In: *Structural and Multidisciplinary Optimization* 43.6 (2011), pp. 767–784.

[127] M. Wang, X. Wang, and D. Guo. “A level set method for structural topology optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 192 (2003), pp. 227–246.

[128] S. Wang, E. de Sturler, and G. H. Paulino. “Large-scale topology optimization using preconditioned Krylov subspace methods with recycling”. In: *Interna-*

tional Journal for Numerical Methods in Engineering 69.12 (2007), pp. 2441–2468.

[129] Y. Wang, Z. Kang, and Q. He. “An adaptive refinement approach for topology optimization based on separated density field description”. In: *Computers & Structures* 117 (2013), pp. 10–22.

[130] A. J. Wathen. “Preconditioning”. In: *Acta Numerica* 24 (May 2015), pp. 329–376.

[131] H. Wendland. *Numerical Linear Algebra: An Introduction*. Cambridge University Press, 2017.

[132] S. J. Wright. *Primal-dual interior-point methods*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[133] J. Xu and L. Zikatanov. “Algebraic multigrid methods”. In: *Acta Numerica* 26 (2017), pp. 591–721.

[134] T. Zegard and G. H. Paulino. “Toward GPU accelerated topology optimization on unstructured meshes”. In: *Structural and Multidisciplinary Optimization* 48.3 (2013), pp. 473–485.

[135] F. Zhang, ed. *The Schur complement and its applications*. Vol. 4. Numerical Methods and Algorithms. Springer, Boston, MA, 2005.

[136] M. Zhou and G. I. N. Rozvany. “The COC algorithm, part II: topological, geometrical and generalized shape optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 89.1-3 (1991), pp. 309–336.

[137] J. Zhu, W. Zhang, and L. Xia. “Topology optimization in aircraft and aerospace structures design”. In: *Archives of Computational Methods in Engineering* 23 (2016), pp. 595–622.

[138] J. Zowe, M. Kočvara, and M. P. Bendsøe. “Free material optimization via mathematical programming”. In: *Mathematical Programming* 79.1 (Oct. 1997), pp. 445–466.

APPENDICES

A.1 Methods and Solvers: Additional Tables and Graphs

Tables A.1 to A.4 contain a detailed comparison of the CG and MINRES solver when used in the IP and PBM method for optimization scenarios with $L = 5, 6$ number of mesh levels. Figures A.1 and A.2 show the development of the duality gap over the course of the optimization for the same scenarios used in Section 3.2.1 to compare the IP, PBM and OC method. It is plotted against the total number of Newton iterations and solver iterations, respectively.

Table A.1: Comparison of CG and MINRES in the IP method for various scenarios with $L = 5$, in terms of Newton iterations, solver iterations and CPU time (in seconds). Values are generally rounded to the nearest integer.

	CG			MINRES		
scenario	Nwt it.	solver it.	time (s)	Nwt it.	solver it.	time (s)
Cantilever 2-2-2-5	51	230	127	51	131	111
Cantilever 4-2-2-5	44	227	234	43	150	213
Cantilever 6-2-2-5	54	391	407	53	225	448
Cantilever 8-2-2-5	41	258	457	40	150	376
Bridge 2-2-2-5	31	139	70	35	110	80
Bridge 4-2-2-5	33	155	152	34	95	154
Bridge 6-2-2-5	30	173	217	25	76	186
Bridge 8-2-2-5	28	198	282	29	134	331
MBB 2-2-2-5	23	130	65	26	97	66
MBB 4-2-2-5	22	124	104	31	136	180
MBB 6-2-2-5	23	122	185	34	150	249
MBB 8-2-2-5	19	81	178	40	202	403
average	33	186	207	37	138	233

Table A.2: Comparison of CG and MINRES in the IP method for various scenarios with $L = 6$, in terms of Newton iterations, solver iterations and CPU time (in minutes). Values are generally rounded to the nearest integer. Note that 500 Newton iterations indicate the maximum iteration number was reached and the optimization terminated before convergence.

	CG			MINRES		
scenario	Nwt it.	solver it.	time (m)	Nwt it.	solver it.	time (m)
Cantilever 2-2-2-6	500	1809	145	500	987	143
Cantilever 4-2-2-6	463	2177	265	448	1273	303
Cantilever 6-2-2-6	344	1982	335	500	1778	459
Cantilever 8-2-2-6	357	2328	539	405	1624	539
Bridge 2-2-2-6	64	320	20	81	264	25
Bridge 4-2-2-6	77	422	46	79	257	59
Bridge 6-2-2-6	73	466	77	81	294	79
Bridge 8-2-2-6	70	530	100	53	213	72
MBB 2-2-2-6	40	262	12	41	156	13
MBB 4-2-2-6	41	394	32	42	217	28
MBB 6-2-2-6	46	550	65	47	310	53
MBB 8-2-2-6	37	418	64	39	218	54
average	176	971	141	193	632	152

Table A.3: Comparison of CG and MINRES in the PBM method for various scenarios with $L = 5$, in terms of Newton iterations, solver iterations and CPU time (in seconds). Values are generally rounded to the nearest integer.

	CG			MINRES		
scenario	Nwt it.	solver it.	time (s)	Nwt it.	solver it.	time (s)
Cantilever 2-2-2-5	71	244	172	82	213	226
Cantilever 4-2-2-5	58	218	301	57	131	304
Cantilever 6-2-2-5	60	297	517	61	170	612
Cantilever 8-2-2-5	57	274	671	57	165	656
Bridge 2-2-2-5	66	352	186	68	286	207
Bridge 4-2-2-5	66	369	354	71	295	417
Bridge 6-2-2-5	73	457	621	71	353	728
Bridge 8-2-2-5	66	374	716	71	280	816
MBB 2-2-2-5	53	286	140	60	263	184
MBB 4-2-2-5	51	287	291	56	252	336
MBB 6-2-2-5	43	226	414	42	144	388
MBB 8-2-2-5	40	228	469	39	107	489
average	59	301	404	61	222	447

Figure A.1: Scaled duality gap vs. Newton iterations for the PBM, IP and OC method. A black diamond marks the termination of the algorithm, which can be outside the plot range.

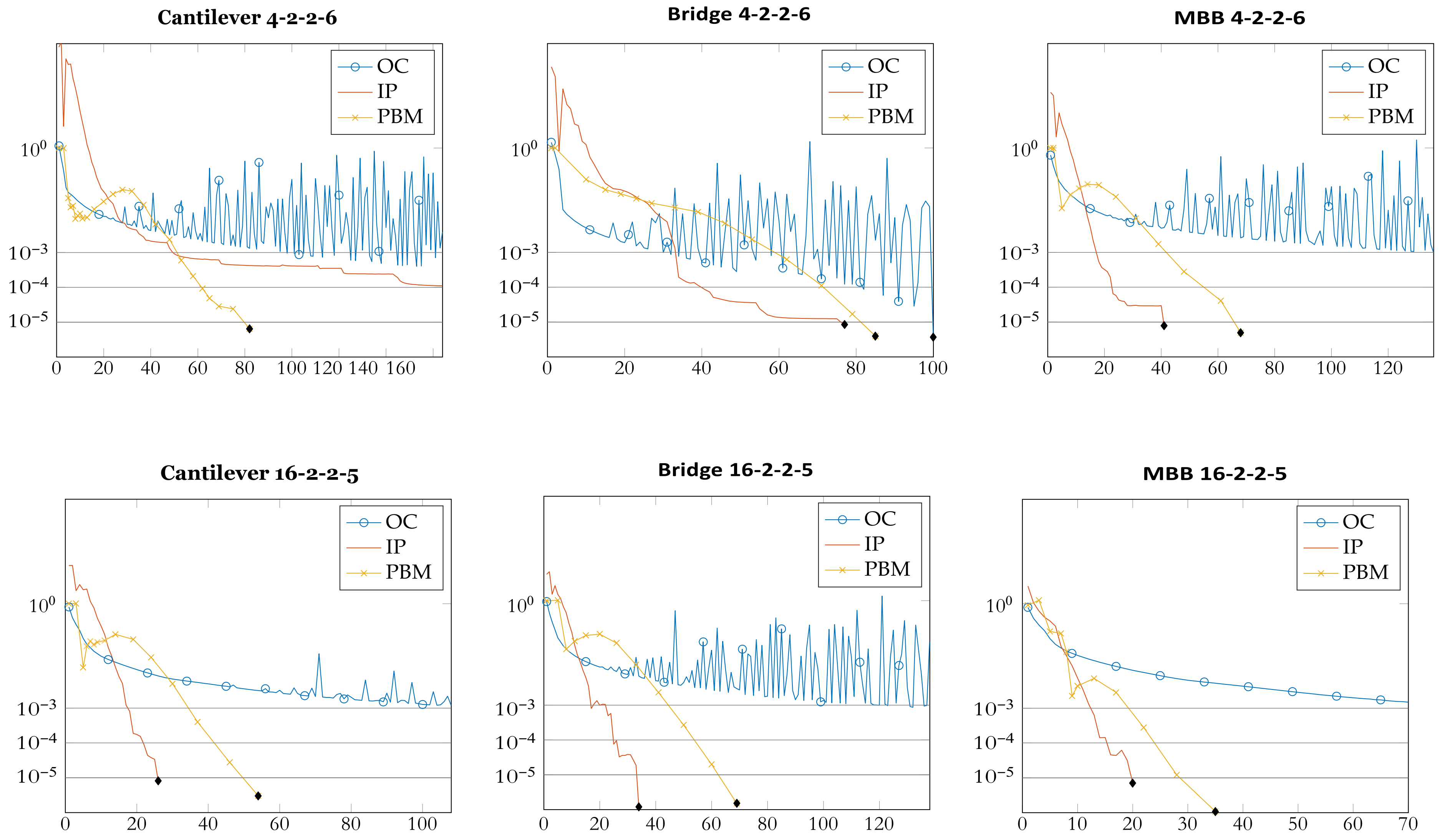


Figure A.2: Scaled duality gap vs. accumulated solver iterations for the PBM, IP and OC method. A black diamond marks the termination of the algorithm, which can be outside the plot range.

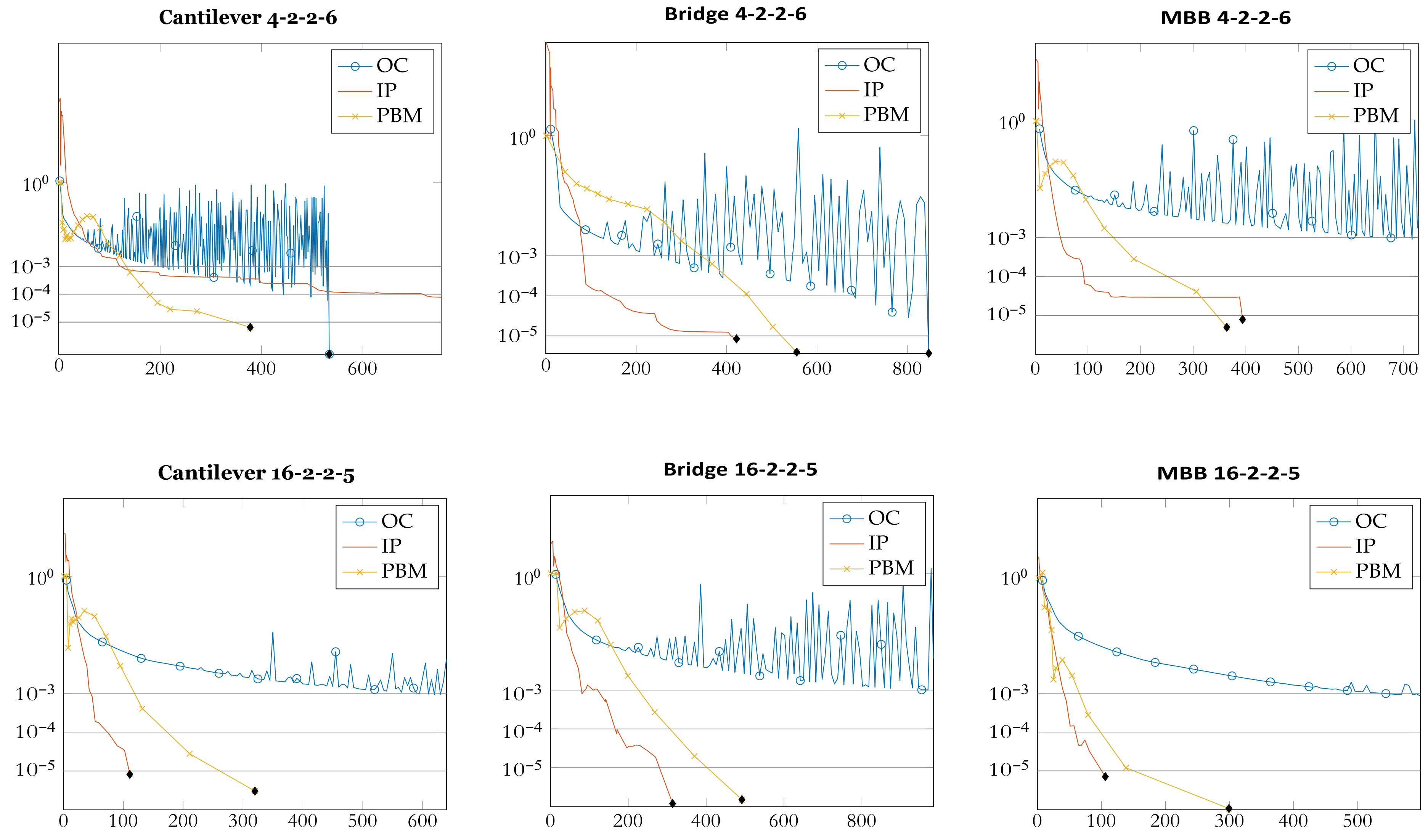


Table A.4: Comparison of CG and MINRES in the PBM method for various scenarios with $L = 6$, in terms of Newton iterations, solver iterations and CPU time (in minutes). Values are generally rounded to the nearest integer.

	CG			MINRES		
scenario	Nwt it.	solver it.	time (m)	Nwt it.	solver it.	time (m)
Cantilever 2-2-2-6	118	411	42	98	178	38
Cantilever 4-2-2-6	82	378	69	82	222	67
Cantilever 6-2-2-6	74	297	86	72	172	89
Cantilever 8-2-2-6	72	314	122	71	180	128
Bridge 2-2-2-6	76	451	29	89	438	39
Bridge 4-2-2-6	85	555	68	95	467	88
Bridge 6-2-2-6	83	517	96	93	484	109
Bridge 8-2-2-6	84	575	132	103	660	240
MBB 2-2-2-6	63	323	24	68	233	28
MBB 4-2-2-6	68	364	55	67	259	55
MBB 6-2-2-6	59	347	71	64	252	83
MBB 8-2-2-6	63	415	105	63	278	112
average	77	412	75	80	319	90

A.2 Aggregation Strategy Statistics

Table A.5: Accumulated solver time (in minutes) for different aggregation strategies and number of extra candidates. Values do not include the time needed for construction of coarse-grid operators. Rows: a) Knee 2-9, b) Knee 3-6, c) Lug 2-8, d) Lug 3-5, e) Crack 2-8, f) Crack 3-5, g) average.

	K1				K+				K++			
	0	1	2	3	0	1	2	3	0	1	2	3
a)	4	3	7	7	4	6	12	10	5	7	5	5
b)	14	13	11	14	20	18	11	19	17	32	34	32
c)	31	22	22	23	32	26	27	21	32	25	27	27
d)	22	31	36	43	27	29	34	44	27	27	31	46
e)	15	22	26	26	18	22	22	23	15	12	19	18
f)	26	51	24	31	44	31	28	41	24	26	17	26
g)	19	24	21	24	24	22	22	26	20	22	22	26

Table A.6: Accumulated multigrid setup time (in minutes) for different aggregation strategies and number of extra candidates. Rows: a) Knee 2-9, b) Knee 3-6, c) Lug 2-8, d) Lug 3-5, e) Crack 2-8, f) Crack 3-5, g) average.

	K1				K+				K++			
	0	1	2	3	0	1	2	3	0	1	2	3
a)	4	11	21	21	14	42	133	99	4	11	15	16
b)	13	27	42	64	40	109	68	126	14	32	42	60
c)	20	49	84	85	289	587	1106	669	22	51	91	90
d)	22	47	79	115	48	204	476	486	25	41	62	122
e)	22	68	132	132	128	552	586	1217	22	58	104	92
f)	22	57	74	101	76	296	199	468	20	38	62	107
g)	17	43	72	86	99	298	428	511	18	38	63	81