RESTRICTION METHODS FOR SHAPE AND TOPOLOGY OPTIMIZATION

BY

CAMERON TALISCHI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Glaucio H. Paulino, Chair and Director of Research
Professor C. Armando Duarte
Professor Robert B. Haber
Professor Eduard-Wilhelm Kirr
Professor Oscar Lopez-Pamies
Professor Eric de Sturler, Virginia Tech

Abstract

This dissertation deals with problems of shape and topology optimization in which the goal is to find the most efficient shape of a physical system. The behavior of this system is captured by the solution to a boundary value problem that in turn depends on the given shape. As such, optimal shape design can be viewed as a form of optimal control in which the control is the shape or domain of the governing state equation. The resulting methodologies have found applications in many areas of engineering, ranging from conceptual layout of high-rise buildings to the design of patient-tailored craniofacial bone replacements.

Optimal shape problems and more generally PDE-constrained inverse problems, however, pose several fundamental challenges. For example, these problems are often ill-posed in that they do not admit solutions in the classical sense. The basic compliance minimization problem in structural design, wherein one aims to find the stiffest arrangement of a fixed volume of material, favors non-convergent sequences of shapes that exhibit progressively finer features. To address the ill-posedness, one either enlarges the admissible design space allowing for generalized micro-perforated shapes, an approach known as "relaxation," or alternatively places additional constraints to limit the complexity of the admissible shapes, a strategy commonly referred to as "restriction." We discuss the issue of existence of solutions in detail and outline the key elements of a well-posed restriction formulation for both density and implicit function parametrizations of the shapes. In the latter case, we demonstrate both mathematically and numerically that without an additional "transversality" condition, the usual smearing of the Heaviside map (which links the implicit functions to the governing state equation), no matter how small, will transform the problem into the so-called variable thickness problem, whose theoretical optimal solutions do not have a clearly-defined boundary. Within the restriction setting, we also analyze and provide a justification for the so-called Ersatz approximation in structural optimization where the void regions are filled by a compliant material in order to facilitate the numerical implementation.

Another critically important but challenging aspect of optimal shape design is dealing with the resulting large-scale non-convex optimization systems which contain many local minima and require expensive function evaluations and gradient calculations. As such, conventional nonlinear programming methods may not be adequately efficient or robust. We develop a simple and tailored optimization algorithm for solving structural topology

optimization problems with an additive regularization term and subject only to a set of box constraints. The proposed splitting algorithm matches the structure of the problem and allows for separate treatment of the cost function, the regularizer, and the constraints. Though our mathematical and numerical investigation is mainly focused on Tikhonov regularization, one important feature of the splitting framework is that it can accommodate nonsmooth regularization schemes such as total variation penalization.

We also investigate the use of isoparametric polygonal finite elements for the discretization of the design and response fields in two-dimensional topology optimization problems. We show that these elements, unlike their low-order Lagrangian counterparts, are not susceptible to certain grid-scale instabilities (e.g., checkerboard patterns) that may appear as a result of inaccurate analysis of the design response. The better performance of polygonal discretizations is attributed to the enhanced approximation characteristics of these elements, which also alleviate shear and volumetric locking phenomena. In regards to the latter property, we demonstrate that low-order finite element spaces obtained from polygonal discretizations satisfy the well-known Babuska-Brezzi condition required for stability of the mixed variational formulation of incompressible elasticity and Stokes flow problems. Conceptually, polygonal finite elements are the natural extension of commonly used linear triangles and bilinear quads to all convex n-gons. To facilitate their use, we present a simple but robust meshing algorithm that utilizes Voronoi diagrams to generate convex polygonal discretizations of implicit geometries. Finally, we provide a self-contained discretization and analysis Matlab code using polygonal elements, along with a general framework for topology optimization.

To my family

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Professor Glaucio Paulino for repeatedly providing me with challenging and exceptional opportunities to develop my professional skills, for the freedom he gave me to explore new research areas and for his continued encouragement and support of my pursuit of mathematics from my early days as an undergraduate student. His wife, Berta, has extended to me her warm hospitality many times during my years at the University of Illinois, and I am deeply grateful for their insight, support, and friendship.

To collaborate with Ivan Menezes and Anderson Pereira has been greatly fulfilling on both a professional and personal level. It has been gratifying to have many pleasant yet productive conversations with people who have become close friends. I owe a great deal of gratitude to Pete Lenzini for the profound impact he has had on my academic life. It was his support and advice that set me on the path of research. Many thanks go to Professors Duarte, Haber, Kirr, Lopez-Pamies, and de Sturler for taking part in my PhD defense committee. I would also like to thank Ilinca Stanciulescu, Emilio Silva, Marco Alfano and the former and current members of Professor Paulino's research group for their encouragement and the pleasure of their acquaintance. I gratefully acknowledge the financial support of the DOE Computational Science Graduate Fellowship Program (under the grant DE-FG02-97ER25308) as well as help and dedication of the staff at the Krell Institute.

I am very thankful for my friends, especially Lauren Stromberg, Alessandro Beghini, Tomas Zegard, Sofie Leon, Wylie Ahmed, George Symeonides, and Christopher Kantas. Without their compassion, humor and camaraderie, my life would lack its finest qualities. I would also like to thank Mary Kate Dunne for her confidence, care, and kindness over the years as well as her help proofreading this thesis. If any errors remain, she is to be blamed.

My deepest gratitude and love are for my father, Ahmad, my mother, Faranak, and my brother Koushyar, to whom this thesis is dedicated. My family's belief in the importance of education, scholarship, and rational thought has profoundly influenced my choices and shaped my character. Their unconditional encouragement and loving care instill in me the drive and enthusiasm to continue to seek new challenges. As in the words of Albert Camus, "An achievement is a bondage. It obliges one to a higher achievement."

Table of Contents

List of Tables

List of Figures

List of Symbols

Ω	an open bounded subset of $\mathbb{R}^d$ with smooth boundary $\partial\Omega$
χ_ω	characteristic function associated with the set ω
$\lvert\omega\rvert$	Lebesgue measure of the set ω
$\mathbf{u}_\omega$	solution to the governing state equation corresponding to ω
J	objective or cost function of the optimal design problem
$\mathcal{O}$	space of admissible shapes
$W^{m,p}(\Omega)$	standard Sobolev space over Ω with $1 \leq p \leq \infty$ and $m \geq 0$
$\lVert\cdot\rVert_{m,p,\Omega}$	norm associated with $W^{m,p}(\Omega)$
$\lvert\cdot\rvert_{m,p,\Omega}$	semi-norm associated with $W^{m,p}(\Omega)$
$L^p(\Omega)$	Lebesgue space $W^{0,p}(\Omega)$ for $1 \leq p \leq \infty$
$H^m(\Omega)$	Sobolev space $W^{m,2}(\Omega)$ for $m \geq 0$
$BV(\Omega)$	space of functions of bounded variation over Ω
$L^p(\Omega; K)$	set of functions $f \in L^p(\Omega)$ with $f(\mathbf{x}) \in K$ almost everywhere in Ω
$\langle f \rangle$	average of function f over its domain Ω defined by $\lvert\Omega\rvert^{-1} \int_\Omega f \mathrm{d}\mathbf{x}$
$\mathcal{A}$	space of admissible designs (e.g. characteristic or density functions)
$\mathbb{P}_C$, $\mathbb{P}_R$	classical and restricted two-phase material distribution problems
$\mathbb{P}_S$	single-phase shape optimization problem
m_E, m_V	material interpolation functions for stiffness and volume
$\mathcal{P}_F$	filtering map with kernel F
$\wedge, \vee$	pointwise max and min operators
$\Pi_{\mathcal{A}}$	projection map onto space $\mathcal{A}$
$\mathcal{I}, \mathbf{I}$	identity map and identity matrix

H	Heaviside map defined by $H(\varphi) = \chi_{\{\varphi>0\}}$
H_γ	smeared or approximate Heaviside map with width γ
d_Ω	signed distance function associated with Ω
R_Ω	reflection operator for domain Ω
$\mathcal{T}(\mathbf{P};\Omega)$	Voronoi tessellation associated with point set $\mathbf{P}$ and domain Ω
$\mathcal{E}(\mathbf{P};\Omega)$	energy functional associated with Voronoi tessellation $\mathcal{T}(\mathbf{P};\Omega)$
$\mathcal{M}_\Omega(\mathbf{P})$	Voronoi mesh of domain Ω using the point set $\mathbf{P}$

Chapter 1

Introduction

This thesis deals with the theoretical and computational issues associated with problems of shape and topology optimization. Due to the inherent difficulties of parametrization of shapes in sufficient generality, the classical shape optimization problem suffers from several pathologies and current methodologies consist of concepts that span several fields in applied and computational mathematics. In this introductory chapter, we describe examples of structural shape design and review the various steps taken from the classical problem one sets out to solve to the final discrete "sizing" problem that is passed to a suitably designed optimization algorithm. The main purpose is to provide the context for the theoretical and computational results presented throughout the thesis as well as the general numerical framework implemented in the educational code provided in the appendix.

Although this chapter serves as a review for the field, at certain points, we provide a perspective that departs from the usual narrative in order to address certain common misconceptions, in particular those related to the Ersatz approach, filtering methods and their finite element discretization. For example, we note that the Ersatz approach, which consists of filling the void region with a compliant material, has to do with the approximation of the governing state equation and not the particular "sizing" formulation adopted. We emphasize that the purpose of filtering is to implicitly ensure smoothness of the design field. It is evident from formulations in which the positive lower bound on the design variables (used as a means to implement the Ersatz approximation) and filtering parameters are related that these separate concepts are sometimes mixed up. Moreover, in some papers, the use of filtering in the compliance problem is inconsistent as it appears only in evaluating the stiffness terms and not the volume constraint. Similarly, we show that the so-called "nonlinear" filtering formulations essentially amount to a modification of the material interpolation functions rather than a change in the filtering operation. We will also discuss the rationale behind filtering at the continuum level, and show the approximation steps (often not explicitly presented) that lead to the discrete operation commonly used. Within this narrative, we will partially address the important but often neglected question of "what optimization problem is solved after the various parametrization and the restriction steps are taken."

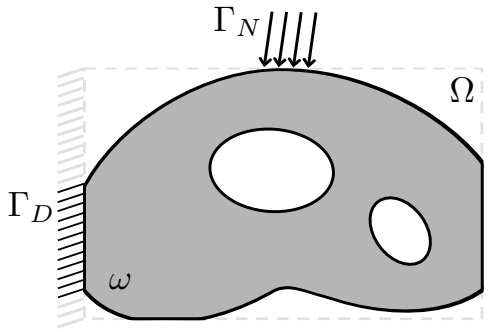


Figure 1.1: Extended design domain and boundary conditions for the governing state equation

1.1 Statement of the classical problem

The goal of topology optimization is to find the most efficient shape $\omega \subseteq \mathbb{R}^d, d = 2, 3$ of a physical system whose behavior is represented by the solution $\mathbf{u}_\omega$ to a boundary value problem. More specifically, one deals with problems of the form

$$\inf_{\omega \in \mathcal{O}} J(\omega, \mathbf{u}_\omega) \tag{1.1}$$

Here $\mathcal{O}$ denotes the set of admissible shapes, J is the cost functional that measures the performance of each candidate shape or design ω. The geometric restrictions on the admissible shapes such as bound on their volume may be prescribed in $\mathcal{O}$. As shown in Figure 1.1, one typically defines an extended design domain or a "hold-all" set Ω in which all the shapes lie, that is, $\omega \subseteq \Omega$ for all $\omega \in \mathcal{O}$. This working domain Ω also helps facilitate the description of the governing boundary value problem.

It is well-known that existence of optimal solutions to this problem is, in general, not guaranteed if $\mathcal{O}$ is defined to be the set of all open or measurable subsets of Ω (see, for example, [96, 3]). Therefore, we must impose constraints on the complexity of the admissible shape in $\mathcal{O}$ in order to ensure that the optimization problem admits a solution, an approach typically known as *restriction* in the literature [142, 28]. This is in contrast to *relaxation* which begins with $\mathcal{O}$ defined as the set of all measurable subsets of Ω and addresses the ill-posedness of the problem by further enlarging the space. We will further elaborate on the distinction between the two approaches and the regularity conditions imposed on $\mathcal{O}$ in chapter 2.

For most of the examples in this thesis, we consider linear elasticity as the governing state equation, which is typical in continuum structural optimization. The solution $\mathbf{u}_\omega \in \mathcal{V}_\omega$ satisfies the variational problem

$$\int_\omega \mathbf{C}\nabla\mathbf{u}_\omega : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{v}\mathrm{d}s, \quad \forall \mathbf{v} \in \mathcal{V}_\omega \tag{1.2}$$

where

$$\mathcal{V}_\omega = \left\{\mathbf{v} \in H^1(\omega)^d : \mathbf{v}|_{\partial\omega\cap\Gamma_D} = \mathbf{0}\right\} \tag{1.3}$$

is the space of admissible displacements, $\mathbf{C}$ is the stiffness tensor of the material of which the shape ω is made, Γ_D and Γ_N form a partition of $\partial\Omega$ and $\tilde{\Gamma}_N \subseteq \Gamma_N$ is where the non-zero tractions are specified. For optimal design problem (1.1) to be non-trivial, we shall assume that for each admissible shape ω, $\partial\omega \cap \Gamma_D$ has non-zero surface measure and $\tilde{\Gamma}_N \subseteq \partial\omega$. This reflects the desire that the admissible shapes are supported on Γ_D and subjected to the design loads defined on $\tilde{\Gamma}_N$. Note that in (1.2), the free boundary of ω, i.e., $\partial\omega\backslash\partial\Omega$, is traction-free. As we can see, the loading is assumed to be independent of the design. We refer to [32, 37] for formulation and analysis of problems with design-dependent loads such as pressure loading and self-weight.

A benchmark model problem in structural optimization is that of minimizing the compliance of the unknown shape subject to a penalty on its volume. In particular, the cost functional is given by

$$J(\omega, \mathbf{u}_\omega) = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{u}_\omega \mathrm{d}s + \lambda \left|\omega\right| \tag{1.4}$$

where the penalty parameter $\lambda > 0$ determines the trade-off between the stiffness provided by the material and the amount that is used (which presumably is proportional to the cost of the design).

The optimal design problem (1.1) with the boundary value problem constraint, as stated in (1.2), is not amenable to typical discretization and optimization strategies without an appropriate parametrization of the space of admissible shapes. For example, the implicit constraints on the space of admissible shapes $\mathcal{O}$ often cannot be directly enforced in the discrete setting. Note also that the internal virtual work term and the space of admissible displacements $\mathcal{V}_\omega$ change with the shape ω further complicating the prospects of discretization. Therefore it is useful to recast the boundary value problem on Ω by using the characteristic function χ_ω associated with ω. That is, we replace (1.2) with

$$\int_\Omega \chi_\omega \mathbf{C}\nabla\mathbf{u}_\omega : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{v}\mathrm{d}s, \quad \forall \mathbf{v} \in \mathcal{V} \tag{1.5}$$

where now the space of admissible displacement,

$$\mathcal{V} = \left\{\mathbf{v} \in H^1(\Omega)^d : \mathbf{v}|_{\Gamma_D} = \mathbf{0}\right\}, \tag{1.6}$$

unlike (1.3), is independent of ω. Accordingly, for the optimal design problem (1.1), we define the admissible space $\mathcal{A}_\mathcal{O} = \{\chi_\omega : \omega \in \mathcal{O}\}$. Now the geometric attributes of shapes ω can be described in terms of the associated characteristic functions in $\mathcal{A}_\mathcal{O}$. In fact, such distributed parametrization simplifies definition and imposition of such constraints. For

example, the volume and perimeter of a shape ω can written as

$$V(\omega) = \int_\Omega \chi_\omega \mathrm{d}\mathbf{x}, \qquad P(\omega) = \int_\Omega |\nabla \chi_\omega| \, \mathrm{d}\mathbf{x} \tag{1.7}$$

where the integral in the second expression is understood as the total variation of function χ_ω. We refer the reader to the monograph by Delfour and Zolesio [61] for a more detailed discussion of representation of shapes by characteristic functions and the associated concepts of geometric measure theory.

The parametrization of the set of admissible shapes via characteristic functions, although a very useful starting point, does not address all the theoretical and practical issues. First, we note that existence and uniqueness of solutions to the extended boundary value problem (1.5) are not guaranteed as the energy bilinear form is no longer coercive (where χ_ω is zero, there is no contribution to the internal virtual work from the displacement; this loss of coercivity corresponds to the singularity of the stiffness matrix in the finite element discretization of this variational equation). The common approach in topology optimization, sometimes referred to the *Ersatz* material model in the level set literature, consists of filling these void regions with compliant material of stiffness $\varepsilon\mathbf{C}$. This amounts to replacing χ with $\varepsilon + (1-\varepsilon)\,\chi$ in the bilinear form of (1.5). The transmission conditions on the boundary $\partial\omega \backslash \partial\Omega$ approximate the traction-free state in (1.2). The validity of this approximation for the optimal design problem (1.1) hinges not only on the convergence of the solutions to the state equation but also on the convergence of the optimal shapes with the modified state equation as $\varepsilon \to 0$. It is clear that the ill-posedness of the optimal shape problem further complicates the analysis of the Ersatz approximation. We refer the reader to [5] for a partial result on degeneracy of compliance minimization in the homogenization framework and to [32] for compliance minimization with design-dependent loads in a restriction framework. In chapter 2, we discuss a sufficient set of regularity conditions that can be imposed on $\mathcal{O}$ and a general proof for consistency of the Ersatz approximation for this space of admissible shapes.

1.2 Continuous parametrization

A major drawback of parametrization of domains with characteristic functions from a practical perspective is that such parametrization does not lend itself to the use of typical nonlinear programming techniques. In particular, note that $\mathcal{A}_\mathcal{O}$ is not a vector space and natural discretization of characteristic functions leads to integer programming problems that are too large in practice to be tractable. For this reason, a "continuous" parametrization of the shape is often adopted by the topology optimization community. This essentially means that the control in the optimization problem is a function that can take values in some

continuous interval. Henceforth we will call this a *sizing function*. This reformulation must be accompanied by a modification of the state equation or addition of a new constraint to the problem so as to recover binary nature of the design.

For example, in density methods, the sizing function is a volume fraction or density function, ρ, that takes values in $[0,1]$ and replaces the characteristic function in the description of the state equation and the objective and constraint functions. To recover "binary" fields in the optimal regime, one can add an explicit penalty term of the form

$$\alpha \int_\Omega \rho(1-\rho)\mathrm{d}\mathbf{x}, \qquad \alpha > 0 \tag{1.8}$$

to the objective function of the problem. For sufficiently large α, the intermediate values of the densities are penalized by virtue of the integrand being strictly positive. Moreover, this penalization is consistent in that if ρ is a characteristic function, the penalty functional vanishes. In practice, however, this approach does not work very well since it adds to nonconvexity and nonlinearity of the problem.

A better alternative is to use one's *a priori* knowledge of the problem and *implicitly* penalize the intermediate densities by augmenting the state equation via an suitable *material interpolation function.* For instance, in the so-called SIMP formulation (in the restriction setting) the space of admissible designs $\mathcal{A}$ is a sufficiently regular subset of $L^\infty(\Omega;[0,1])$[1], and the state equation is given by:

$$\int_\Omega \left[\varepsilon + (1-\varepsilon)\,\rho^p\right] \mathbf{C}\nabla\mathbf{u} : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{v}ds, \quad \forall \mathbf{v}\in\mathcal{V} \tag{1.9}$$

where $p > 1$ is a penalization exponent [20, 136, 135]. Meanwhile, ρ enters linearly in the geometric constraints such as the volume or perimeter of the design:

$$V(\rho) = \int_\Omega \rho\mathrm{d}\mathbf{x}, \qquad P(\rho) = \int_\Omega |\nabla\rho|\,\mathrm{d}\mathbf{x} \tag{1.10}$$

Intuitively, the intermediate volume fractions are penalized as they are assigned a smaller stiffness compared to their contribution to volume. If one can establish that the optimal solution to the SIMP problem is a characteristic function, it follows from the fact that $\rho^p = \rho$ for all $\rho \in \mathcal{A} \cap L^\infty(\Omega;\{0,1\})$, that enlarging the design space from characteristic functions to volume fraction functions is justified. Unfortunately, this is typically not the case when one is operating in a restriction setting since the admissible densities are smooth and the variation between the extreme values occurs over a transition region. Such results are, however, available for certain discrete formulations of SIMP [145, 133, 112]. As

[1]Here $L^\infty(\Omega;K)$ denotes the space of measurable functions defined on Ω that take values in $K \subseteq \mathbb{R}$. For example, $L^\infty(\Omega;\{0,1\})$ and $L^\infty(\Omega;[0,1])$ denote the space of measurable functions that take values in $\{0,1\}$ and interval $[0,1]$, respectively.

discussed in chapter 3, there is also numerical evidence that one can obtain near binary optimal densities for a well-posed compliance minimization problem based on total variation regularization.

We note that there is an alternative rationale for density methods, such as SIMP, that has its roots in the *relaxation* of the optimal design problem (1.1) [96, 97, 98, 165, 48, 3]. This so-called material distribution perspective inherently involves the concept of homogenization. The intuition is that severe oscillations of near optimal solutions (the root-cause of the ill-posedness of the classical problem) is taken into account by enlarging the space of admissible designs to include the corresponding generalized shapes. Since these oscillations take place at a fine scale, their effect in such generalized designs is captured through the homogenized properties of the corresponding (composite) material in the enlarged space. A density function that measures the volume fraction of these oscillations at the macroscopic scale characterizes each generalized design (see chapter 2).

Here, we are adopting a different perspective—that of an approximation of a characteristic function—since we are operating in the restriction setting which is natural for most practical applications of topology optimization where layout or manufacturing constraints are present. It is difficult to extend the relaxation philosophy to the restriction framework since it amounts to distributing material freely at the microscale but subject to a restricted variation at the macroscale. In a sense, this undermines the notion of relaxation from both mathematical and physical perspectives. Nevertheless, one can explicitly construct composite materials that follow the SIMP model [23], i.e., ones whose stiffness and volume fraction coincide with SIMP, and so, in principle, the designs can be realized. For yet another physical justification of SIMP based on fictitious fabrication cost, we refer the reader to [135, 183]. We note, however, that the SIMP problem without additional regularity conditions imposed on the admissible density functions is still ill-posed.

At this point a natural question arises: which space of admissible shapes $\mathcal{O}$ corresponds to the "continuous" and restricted design space $\mathcal{A}$? Considering the fact that the optimal density function ρ^* is typically not a characteristic function, one usually interprets the result to obtain a classical shape $\omega^* \subseteq \Omega$ via a post-processing routine. The question is: *in what sense is ω^* optimal?*

If the performance of the density functions, in the optimal regime, is well-approximated by that of the corresponding classical shapes, we can then argue that ω^* is "nearly" optimal *when the space of admissible shapes $\mathcal{O}$ consists of shapes corresponding to the approximate characteristic functions in $\mathcal{A}$.* Note that in such a scenario, the set of admissible shapes $\mathcal{O}$ is reasonably well-defined and perhaps captures the intention of the restriction scheme while the continuous parametrization setting is justified.

To further illustrate the notion of "post-processing" and what is meant by the "optimal regime," consider the three smooth density functions ρ taking values in $[0, 1]$ shown in

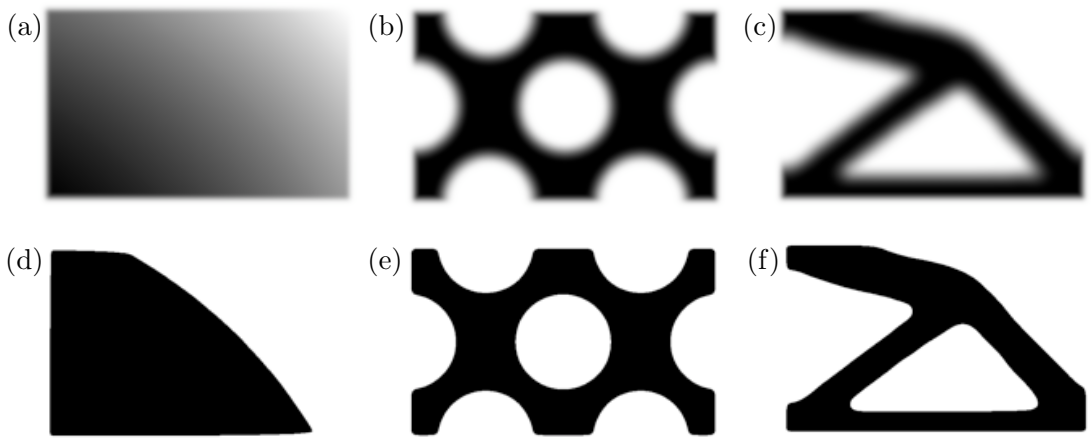


Figure 1.2: (a)(b)(c) greyscale plot of three "smooth" density functions ρ taking values in $[0, 1]$; (d)(e)(f) the corresponding interpreted shapes defined by $\chi_{\{\rho \geq 0.5\}}$

Figure 1.2. The last two functions approximate a characteristic function: the bottom row shows the associated characteristic functions defined by $\chi_{\{\rho \geq 0.5\}}$[2]. It is clear that only in the last two cases, ρ and $\chi_{\{\rho \geq 0.5\}}$ are "close." Furthermore, the continuous density formulations of SIMP is set up so that in the optimal regime, only designs of this kind appear, i.e., ones for which $\rho \approx \chi_{\{\rho \geq 0.5\}}$. The continuous parametrization is acceptable if, in addition, the values of objective and constraints functions are well approximated, that is, $J(\rho) \approx J\left(\chi_{\{\rho \geq 0.5\}}\right)$[3]. These conditions are evidently true for compliance minimization with SIMP though a mathematical proof is not available in the continuum setting.

So far in this discussion, the notion of a sizing function may seem synonymous with densities[4]. However, we note that the continuous parametrization is not limited to density methods, and level set formulations [8, 173, 19, 59, 168] can be also placed in the same framework: the level set or implicit function φ can assume both positive and negative values and can be required to belong to the bounded interval $[-\alpha, \alpha]$ for some $\alpha > 0$ [19]. What enters in the state equation and constraint functions, in place of χ_ω, is $H(\varphi)$, where H is the Heaviside function (or a smooth approximation to it). Note that we are not commenting on the choice of the optimization algorithm that is ultimately used in the discrete setting (the evolution of the optimal shape in level set methods in some of the above-mentioned references is based on shape sensitivity analysis and motion of the shape boundary). Similar to the density formulations, some constraints must be imposed

[2]This function takes value of 1 at point $\mathbf{x} \in \Omega$ if $\rho(\mathbf{x}) \geq 0.5$ and is zero otherwise – this is a simple choice of post-processing

[3]Throughout this thesis, with a slight abuse of notation, we will use J to denote the objective as a function of the underlying design parametrization, which should be evident from the context. For example, within a density framework, J is a function of the density field ρ with the dependence of J on $\mathbf{u}_\rho$ implied.

[4]Variants of density methods involve different material interpolation functions, but are similar in spirit [144, 36].

on the variation of the implicit functions so that the resulting optimization problem is well-posed. Also some mechanism must ensure that the implicit function in the optimal regime is sufficiently steep near the boundary so that the stiffness is near binary throughout the domain. The precise nature of this so-called transversality condition and its implication are discussed in detail in the next chapter.

1.3 Restriction and regularity of sizing functions

We now discuss the issue of the regularity of the space of admissible sizing functions. As mentioned before, this is relevant from a theoretical perspective since it is related to the well-posedness of the problem. We emphasize that continuous parametrization (by replacing characteristic functions with sizing functions) by itself does not address this issue. Moreover, restriction provides an appropriate setting for justifying the commonly used Ersatz approach. From a practical point of view, restricting the variations of the sizing functions is related to manufacturing constraints on the admissible geometries considered.

One set of restriction formulations imposes local or global regularity constraints explicitly with the aid of the "continuous" parametrization. For example, in the perimeter constraint formulation, the addition of the following constraint [11, 88, 127] to $\mathcal{A}$

$$\int_{\Omega} |\nabla \rho| \, \mathrm{d}\mathbf{x} \leq \overline{P} \tag{1.11}$$

ensures that the admissible densities do not oscillate too much by requiring that the total variation of the design field be bounded by the prescribed perimeter $\overline{P}$. A Tikhonov-type scheme introducing the additional constraint

$$\int_{\Omega} |\nabla \rho|^2 \, \mathrm{d}\mathbf{x} \leq \overline{M} \tag{1.12}$$

is similar in that it also restricts large variations (gradients) of the density field. These two formulations are considered in chapters 3 and 4 in alternative but equivalent forms wherein regularizing terms are added to the cost functional to enforce regularity only in the optimal regime rather than restricting the entire design space.

In the slope constraint formulation [128], $\mathcal{A} \subseteq W^{1,\infty}(\Omega)$ (admissible functions are weakly differentiable with essentially bounded derivatives) and

$$\operatorname*{ess\,sup}_{\mathbf{x}\in\Omega} |\nabla \rho(\mathbf{x})|_{\infty} \leq \overline{G} \tag{1.13}$$

which means that the gradient of $\rho \in \mathcal{A}$ cannot be too large (i.e., exceed the specified value $\overline{G}$) throughout the extended domain Ω. In contrast to the "global" total variation

and Tikhonov constraints, this is an example of a local constraint on density variation. The discretization of this local constraint leads to a large number of linear constraints for the optimization problem and can be prohibitively expensive.

The alternative approach, emphasized in this chapter, is to impose regularity on $\mathcal{A}$ *implicitly* with the aid of a "regularization" map $\mathcal{P}$. For example, in the popular filtering formulation [31, 29], $\mathcal{A}$ consists of density functions that are produced via convolution with a smooth filter function F, that is,

$$\mathcal{A} = \{\mathcal{P}_F(\eta) : \eta \in L^\infty(\Omega; [0,1])\} \tag{1.14}$$

where $\mathcal{P}_F$ is the integral operator defined as

$$\mathcal{P}_F(\eta)(\mathbf{x}) := \int_\Omega F(\mathbf{x}, \mathbf{y})\eta(\mathbf{y})\mathrm{d}\mathbf{y} \tag{1.15}$$

The expression (1.14) states that for each $\rho \in \mathcal{A}$, there exists a measurable function η such that $\rho = \mathcal{P}_F(\eta)$. By the virtue of the properties of the map $\mathcal{P}_F$, the sizing function ρ inherits the smoothness characteristics of the kernel F. Therefore, even if η is rough, ρ is guaranteed to be smooth (see Figure 1.3(a) and (b)). As we shall discuss in the next section, it is the discretization of η that produces the set of design variables for the optimization problem. Thus, we can see that the use of $\mathcal{P}_F$ in defining the sizing functions eliminates the need to explicitly impose regularity on ρ.

The linear "hat" kernel of radius R, which is the common choice for filtering, is given by

$$F(\mathbf{x}, \mathbf{y}) = c(\mathbf{x}) \max\left(1 - \frac{|\mathbf{x} - \mathbf{y}|}{R}, 0\right) \tag{1.16}$$

where $c(\mathbf{x})$ is defined as a normalizing coefficient such that

$$\int_\Omega F(\mathbf{x}, \mathbf{y})\mathrm{d}\mathbf{y} = 1 \tag{1.17}$$

for all $\mathbf{x} \in \Omega$. It is easy to see that the expression for this coefficient is

$$c(\mathbf{x}) = \left[\int_{B_R(\mathbf{x})\cap\Omega} \left(1 - \frac{|\mathbf{x} - \mathbf{y}|}{R}\right) \mathrm{d}\mathbf{y}\right]^{-1} \tag{1.18}$$

where $B_R(\mathbf{x})$ is the ball of radius R around $\mathbf{x}$. The condition (1.17) guarantees that the bounds for the filtered field $\mathcal{P}_F(\eta)$ are the same as those set for the design function η (e.g., 0 and 1 for design space in (1.14)). We emphasize, however, that in light of the previous discussion on definition of $\mathcal{A}$, the role of the map $\mathcal{P}_F$ is simply to restrict the space of admissible densities. This distinction has been made in [141] where the author refers to

$\rho \in \mathcal{A}$ as the "physical" density function to distinguish it from $\eta \in L^\infty(\Omega; [0,1])$.

We can prescribe other layout or manufacturing constraints on admissible shapes via implicit maps in the same manner that the filtering approach enforces smoothness. We illustrate this idea via an example for enforcing symmetry and remark that manufacturing constraints such as extrusion, pattern repetition and gradation can be imposed in a similar way [99, 10, 146]. Suppose $\Omega \subseteq \mathbb{R}^2$ is a symmetric domain with respect to the x_1-axis and let $\Omega^+ = \{(x_1, x_2) \in \Omega : x_2 \geq 0\}$. Rather than adding constraints of the form

$$\rho(x_1, x_2) = \rho(x_1, -x_2) \tag{1.19}$$

for all $\mathbf{x} = (x_1, x_2) \in \Omega$ (which is conceivable in the discrete setting), we can build symmetry into the space of admissible sizing functions by defining the operator $\mathcal{P}_s$ that maps function η defined on Ω^+ to the function $\mathcal{P}_s(\eta)$ defined on Ω with the property that

$$\mathcal{P}_s(\eta)(\mathbf{x}) = \eta(x_1, |x_2|) \tag{1.20}$$

for every $\mathbf{x} = (x_1, x_2) \in \Omega$ and setting[5]

$$\mathcal{A} = \left\{\mathcal{P}_s(\eta) : \eta \in L^\infty(\Omega^+; [0,1])\right\} \tag{1.21}$$

Again this means that $\rho \in \mathcal{A}$ if $\rho = \mathcal{P}_s(\eta)$ for some $\eta \in L^\infty(\Omega^+; [0,1])$ and so ρ automatically satisfies (1.19). Given this discussion, it is easy to combine the symmetry and filtering using composition of the two maps $\mathcal{P} = \mathcal{P}_F \circ \mathcal{P}_s$.

1.4 Discretization

Two main features of common well-posed topology optimization formulations are interpolation models and regularity of admissible designs. Many topology optimization formulations can thus be cast in the form of a sizing problem

$$\inf_{\rho \in \mathcal{A}} J(\rho, \mathbf{u}_\rho) \tag{1.22}$$

where the space of admissible sizing functions is

$$\mathcal{A} = \left\{\mathcal{P}(\eta) : \eta \in L^\infty\left(\Omega; \left[\underline{\rho}, \overline{\rho}\right]\right)\right\} \tag{1.23}$$

[5]The distinction between the η and the admissible function $\mathcal{P}_s(\eta)$ should be more apparent here: η is defined on half of the domain Ω^+ while $\mathcal{P}_s(\eta)$ is defined over all of Ω.

and $\mathbf{u}_\rho \in \mathcal{V}$ solves

$$\int_\Omega m_E(\rho)\mathbf{C}\nabla\mathbf{u}_\rho : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{v}\mathrm{d}s, \quad \forall\mathbf{v}\in\mathcal{V} \tag{1.24}$$

Here m_E is the material interpolation function that relates the value of ρ at a point to the stiffness at that point[6]. Similarly, interpolation functions for other geometric measures such as the perimeter may be needed for the formulation. For example, the compliance minimization problem

$$J(\rho,\mathbf{u}_\rho) = \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{u}_\rho\mathrm{d}s + \lambda\int_\Omega m_V(\rho)\mathrm{d}\mathbf{x} \tag{1.25}$$

requires one additional interpolation function, m_V, for the volume constraint. Aside from m_V and m_E, in this framework we need to provide bounds, $\underline{\rho}$ and $\overline{\rho}$, as well as the mapping $\mathcal{P}$.

The reformulation of the optimal design problem in the form of distributed sizing optimization (1.22) lends itself to a tractable discretization and optimization scheme. In particular, the finite element (FE) discretization of the design field $\mathcal{A}$ requires partitioning of the extended domain Ω without the need for remeshing as the design is evolving in the course of the optimization. Often the displacement field is discretized based on the same FE partition although this choice can lead to certain numerical artifacts such as checkerboard patterns [62, 91, 142]. For example, one often obtains spurious solutions when using lower-order Lagrangian elements such as linear triangles and bilinear quadrilateral in the absence of the restriction of the density fields. In contrast, as we will demonstrate in chapter 5, polygonal discretizations are not susceptible to such numerical instabilities. Note also that in the restriction setting, such coupled discretization strategies can be shown to be convergent under mesh refinement so the aforementioned numerical instabilities are expected to disappear when a sufficiently fine mesh is used (e.g., when the mesh size is smaller than the filtering radius). In chapter 4, we prove a finite element convergence result for Tikhonov regularization. Similar results are available for filtering formulations as well (see, for example, [128, 31, 29]).

We will next discuss the various steps in the discretization process based on one finite element mesh. The goal is to identify the design variables for the discrete optimization problem and how they are related to the parameters defining the state equations and subsequently the cost functional. As we shall see, this involves an approximation of the mapping $\mathcal{P}$ which is usually not discussed in the topology optimization literature.

Let $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$ be a partition of Ω, i.e., $\Omega_\ell \cap \Omega_k = \emptyset$ for $\ell \neq k$ and $\cup_\ell \overline{\Omega}_\ell = \overline{\Omega}$ with h denoting the characteristic mesh size. In the analysis of convergence of FE solutions to a

[6] m_E essentially determines the dependence of the state equation on the design

solution of (1.22), h is sent to zero. We intend to identify the discrete counterpart to (1.22) based on this partition and so in the remainder of this section $\mathcal{T}_h$ is fixed. The *piecewise constant* discretization of $\mathcal{A}$ in (1.23) is defined as:

$$\mathcal{A}_h = \left\{ \mathcal{P}(\eta_h) : \eta_h \in L^\infty \left(\Omega; \left[\underline{\rho}, \overline{\rho} \right] \right), \eta_h|_{\Omega_\ell} = \text{const } \forall \ell \right\} \tag{1.26}$$

In other words, each $\rho \in \mathcal{A}_h$ is the image of map $\mathcal{P}$ acting on a design function η_h that takes a constant value over each element Ω_ℓ. Note that η_h belongs to a finite dimensional space of functions of the form:

$$\eta_h(\mathbf{x}) = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}(\mathbf{x}) \tag{1.27}$$

where $\chi_{\Omega_\ell}(\mathbf{x})$ is the characteristic function associated with element Ω_ℓ and z_ℓ is the constant value that η_h assumes over Ω_ℓ. Furthermore, (1.26) is defined such that each $\rho_h \in \mathcal{A}_h$ can be written as $\rho_h = \mathcal{P}(\eta_h)$ for some η_h in the form of (1.27). Thus each candidate design in $\mathcal{A}_h$ can be defined by a set of design of variables $\mathbf{z} := [z_\ell]_{\ell=1}^N$, which is provided to the optimization algorithm for this partitioning. The only difference between $\mathcal{A}$ and $\mathcal{A}_h$ with the above definition is the restriction placed on η_h. It is precisely the discretization of the functions η that produces the design variables $\mathbf{z}$ that on the one hand, completely characterize the discrete space of admissible designs $\mathcal{A}_h$ and on the other hand become the parameters passed to the optimization algorithm for sizing.

As mentioned before, it is convenient that the state equation, more specifically the displacement field $\mathcal{V}$, is discretized on the same partition $\mathcal{T}_h$. Let $\mathcal{V}_h$ be this finite dimensional subspace and suppose each $\mathbf{u}_h \in \mathcal{V}_h$ has the expansion

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{M} U_i \mathbf{N}_i(\mathbf{x}) \tag{1.28}$$

that is, $\{\mathbf{N}_i\}_{i=1}^M$ is the basis for $\mathcal{V}_h$ and M is the number of displacement degrees of freedom. The Galerkin approximation to the state equation (1.24) with $\rho = \rho_h \in \mathcal{A}_h$ can be written as

$$\mathbf{K}\mathbf{U} = \mathbf{F} \tag{1.29}$$

where $\mathbf{U} = [U_i]_{i=1}^M$ is the vector of nodal displacements,

$$[\mathbf{F}]_i = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{N}_i \mathrm{d}s \tag{1.30}$$

are the nodal loads, and

$$\begin{aligned}[\mathbf{K}]_{ij} &= \int_\Omega m_E(\rho_h)\mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j \mathrm{d}\mathbf{x} \\ &= \sum_{\ell=1}^{N} \int_{\Omega_\ell} m_E(\rho_h)\mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j \mathrm{d}\mathbf{x} \end{aligned} \tag{1.31}$$

The integral inside the summation is the (i, j)-th entry of the stiffness matrix for element Ω_ℓ in the global node numbering. We note that $\rho_h = \mathcal{P}(\eta_h)$ may not be constant over Ω_ℓ and therefore cannot be pulled outside of this integral. Similarly, quantities like the volume of the design cannot be readily computed given the finite element partition. This highlights the fact that in common algorithms for topology optimization an additional approximation step is used beyond what is explicitly stated in the definition of $\mathcal{A}_h$ in (1.26). This approximation in solving the state equation in fact amounts to the discretization of the operator $\mathcal{P}$ and therefore could be explicitly expressed in the definition of the space of admissible designs $\mathcal{A}_h$.

Typically, in practice, the sizing function ρ_h is replaced by a function $\tilde{\rho}_h$ that is constant over each finite element, that is,

$$\tilde{\rho}_h(\mathbf{x}) = \sum_{\ell=1}^{N} y_\ell \chi_{\Omega_\ell}(\mathbf{x}) \tag{1.32}$$

One choice for the elemental values y_ℓ is to sample the value of ρ_h at the centroid of element ℓ,

$$y_\ell = \rho_h(\mathbf{x}_\ell^*) \tag{1.33}$$

where we have denoted by $\mathbf{x}_\ell^*$ the location of the centroid of the element Ω_ℓ. For example, in the case of filtering, we have

$$\begin{aligned} y_\ell &= \rho_h(\mathbf{x}_\ell^*) \\ &= \mathcal{P}_F(\eta_h)(\mathbf{x}_\ell^*) \\ &= \int_\Omega F(\mathbf{x}_\ell^*, \overline{\mathbf{x}})\eta_h(\overline{\mathbf{x}})\mathrm{d}\overline{\mathbf{x}} \\ &= \sum_{k=1}^{N} z_k \underbrace{\int_{\Omega_k} F(\mathbf{x}_\ell^*, \overline{\mathbf{x}})\mathrm{d}\overline{\mathbf{x}}}_{:=w_{\ell k}} \end{aligned} \tag{1.34}$$

Collecting the weights computed in a matrix $[\mathbf{P}]_{\ell k} = w_{\ell k}$, we can relate the elemental values of $\tilde{\rho}_h$ to those of η_h by

$$\mathbf{y} = \mathbf{P}\mathbf{z} \tag{1.35}$$

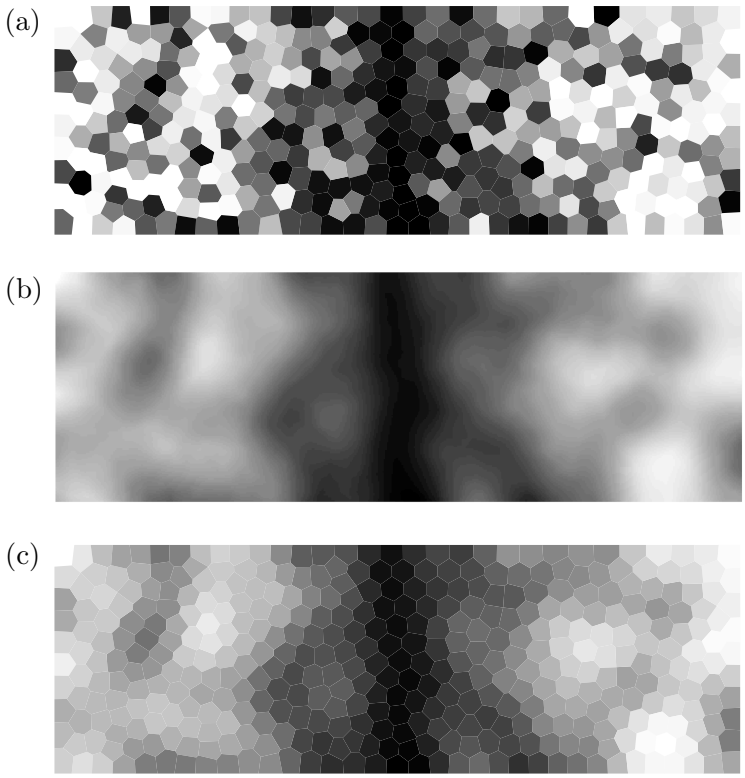


Figure 1.3: Illustration of the effects of filtering mapping and its discretization: (a) $\eta_h = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}$ for a random vector of design variables $\mathbf{z} = [z_\ell]_{\ell=1}^{N}$; (b) $\mathcal{P}_F(\eta_h)$ for the design function η_h shown in (a) and linear kernel F. Note that despite the severe oscillations of η_h, $\mathcal{P}_F(\eta_h)$ has smooth variation dictated by F; (c) $\mathcal{P}_F^h(\eta_h)$ which is the element-wise constant approximation to $\mathcal{P}_F(\eta_h)$ on the same finite element partition based on which η_h is defined

Note that the weights $w_{\ell k}$ are independent of the design variables $\mathbf{z}$ and can be computed once at the beginning of the algorithm. Moreover, many of the weights are zero since $\mathbf{x}_\ell^*$ is not in the support of $\mathcal{P}_F(\chi_k)$ when Ω_ℓ and Ω_k are far apart in the mesh. Therefore, $\mathbf{P}$ can be efficiently stored in a sparse matrix. In the appendix, it is shown that these weights correspond to the usual discrete filtering formulas used for the linear hat function (1.16).

The matrix $\mathbf{P}$ must be viewed as the discrete counterpart to the mapping $\mathcal{P}_F$. In fact, such discretization of any *linear* map[7] $\mathcal{P}$ in the definition of $\mathcal{A}$ produces a constant matrix. Let us define the "discretized" map[8]

$$\mathcal{P}_h : \eta \to \sum_{\ell=1}^{N} \mathcal{P}(\eta)(\mathbf{x}_\ell^*)\chi_{\Omega_\ell} \tag{1.36}$$

Note that for each piecewise constant design function η_h and linear map $\mathcal{P}$,

$$\mathcal{P}(\eta_h)(\mathbf{x}_\ell^*) = \mathcal{P}\left(\sum_{k=1}^{N} z_k \chi_{\Omega_k}\right)(\mathbf{x}_\ell^*) = \sum_{k=1}^{N} z_k \mathcal{P}(\chi_{\Omega_k})(\mathbf{x}_\ell^*) = \mathbf{P}\mathbf{z} \tag{1.37}$$

where

$$[\mathbf{P}]_{\ell k} = \mathcal{P}(\chi_{\Omega_k})(\mathbf{x}_\ell^*) \tag{1.38}$$

This illustrates the relationship between $\mathcal{P}_h$ and matrix $\mathbf{P}$. Just as the vector $\mathbf{z}$ represents the elemental values of design function η_h, vector $\mathbf{P}\mathbf{z}$ gives the elemental values of the piecewise constant function $\mathcal{P}_h(\eta_h)$.

A more precise description of the space of admissible designs $\mathcal{A}_h$ associated with $\mathcal{T}_h$ that accounts for this approximation is thus given by

$$\mathcal{A}_h = \left\{\mathcal{P}_h(\eta_h) : \eta_h \in L^\infty\left(\Omega; \left[\underline{\rho}, \overline{\rho}\right]\right), \eta_h|_{\Omega_\ell} = \text{const } \forall \ell\right\} \tag{1.39}$$

Figure 1.3 illustrates the effects of the filtering mapping $\mathcal{P}_F$ with a linear hat kernel F and its discretization $\mathcal{P}_F^h$.

With $\tilde{\rho}_h$ replacing ρ_h in expression (1.31), which amounts to replacing $\mathcal{P}(\eta_h)$ with $\mathcal{P}_h(\eta_h)$, the stiffness matrix is simplified to:

$$\mathbf{K} = \sum_{\ell=1}^{N} m_E(y_\ell)\mathbf{k}_\ell \tag{1.40}$$

where $[\mathbf{k}_\ell]_{ij} = \int_{\Omega_\ell} \mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j d\mathbf{x}$ is the ℓth element stiffness matrix. As mentioned before, it is not just the calculation of the stiffness matrix that benefits from the approximation $\mathcal{P}$.

[7] Filtering, symmetry, pattern repetition, and extrusion constraints can all be implemented by means of such linear maps

[8] Alternatively we can view $\mathcal{P}_h = \mathcal{I}_h \circ \mathcal{P}$ where $\mathcal{I}_h$ maps any ρ_h to $\tilde{\rho}_h$, that is, $\mathcal{I}_h(\rho) = \sum_{\ell=1}^{N} \rho(\mathbf{x}_\ell^*)\chi_{\Omega_\ell}$

Evaluation of objective and constraint functions that involve volume and surface integrals are also now greatly simplified. For example, the volume term can be written as

$$\int_{\Omega} m_V(\tilde{\rho}_h)\mathrm{d}\mathbf{x} = \sum_{\ell=1}^{N} m_V(y_\ell)\,|\Omega_\ell| \tag{1.41}$$

We note that this additional approximation is often not explicitly considered in the convergence proofs (e.g. in [29]) but is justifiable since its effect disappears as the mesh size h goes to zero. However, accuracy considerations become relevant. Since there is only one mesh used, the variation of η_h and accuracy of $\tilde{\rho}_h$ and $\mathbf{u}_h$ are tied to the same FE mesh.

We conclude this section by stating the discrete compliance minimization problem defined on $\mathcal{T}_h$

$$\inf_{\rho_h \in \mathcal{A}_h} \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{u}_h \mathrm{d}s + \lambda \int_{\Omega} m_V(\rho_h)\mathrm{d}\mathbf{x} \tag{1.42}$$

where the space of admissible functions $\mathcal{A}_h$ is given in (1.39), and $\mathbf{u}_h \in \mathcal{V}_h = \operatorname{span}\{\mathbf{N}_i\}_{i=1}^{M}$ solves the discretized state equation

$$\int_{\Omega} m_E(\rho_h)\mathbf{C}\nabla\mathbf{u}_h : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{v}\mathrm{d}s, \quad \forall\mathbf{v}\in\mathcal{V}_h \tag{1.43}$$

The only difference between (1.42) and the continuum problem (1.22) is that spaces $\mathcal{A}$ and $\mathcal{V}$ are replaced by their finite dimensional counterparts $\mathcal{A}_h$ and $\mathcal{V}_h$.

This problem is completely equivalent to the familiar discrete form[9]

$$\min_{\mathbf{z}\in\left[\underline{\rho},\overline{\rho}\right]^N} J(\mathbf{z}) = \mathbf{F}^T\mathbf{U}(\mathbf{z}) + \lambda\mathbf{A}^T m_V(\mathbf{P}\mathbf{z}) \tag{1.44}$$

where $\mathbf{F}$ given by (1.30) is independent of the design variables $\mathbf{z}$, $\mathbf{U}(\mathbf{z})$ solves (1.29) in which $\mathbf{K}$ depends "linearly" on $m_E(\mathbf{P}\mathbf{z})$ as stated in (1.40), $[\mathbf{A}]_\ell = |\Omega_\ell|$ is the vector of element volumes, and $\mathbf{P}$ is defined in (1.38).

1.5 Optimization algorithm

Next we discuss the main features of the algorithms commonly used in structural optimization for solving the discrete problem (1.44). The basic approach consists of replacing the objective and constraint functions with some suitable approximations in the neighborhood of the current design point. That is, one solves the following approximate problem in each iteration:

$$\min_{\mathbf{z}^{\mathsf{L}}\le\mathbf{z}\le\mathbf{z}^{\mathsf{U}}} J_{\mathsf{app}}(\mathbf{z}) \tag{1.45}$$

[9] In the remainder of the thesis, we understand $m_E(\mathbf{y})$ and $m_V(\mathbf{y})$ to be vectors with entries $m_E(y_\ell)$ and $m_V(y_\ell)$

where J_{app} is an approximation to the objective function and $\mathbf{z}^{\mathsf{L}}$ and $\mathbf{z}^{\mathsf{U}}$ specify the lower and upper bounds for the search region where the approximation is valid. For example, if m denotes the admissible *move limit* and $\mathbf{z}_0$ is the design at the current iteration, then

$$z_\ell^{\mathsf{L}} = \max\left(\underline{\rho}, z_\ell^0 - m\right), \quad z_\ell^{\mathsf{U}} = \min\left(\overline{\rho}, z_\ell^0 + m\right), \quad \ell = 1, \ldots, N \tag{1.46}$$

The key to the success of such an algorithm is the choice of the approximation functions in terms of their accuracy, the cost of computation, and the effort needed to solve the resulting optimization algorithm. Here we will discuss the scheme that leads to the so-called Optimality Criteria (OC) algorithm commonly used in topology optimization. In chapters 3 and 4, a new class of algorithms is developed that separate the treatment of the structural cost functional and the regularization terms.

First note that the objective function (1.44) can be written as $J(\mathbf{z}) = f(\mathbf{z}) + \lambda g(\mathbf{z})$ where g is a linear function of $\mathbf{z}$ and so it can be written as

$$g(\mathbf{z}) = g\left(\mathbf{z}^0\right) + \sum_{\ell=1}^{N} \left(z_\ell - z_\ell^0\right) \partial_\ell g\left(\mathbf{z}^0\right) \tag{1.47}$$

To obtain the approximate function $J_{\mathsf{app}}(\mathbf{z})$, the first term $f(\mathbf{z}) = \mathbf{F}^T\mathbf{U}(\mathbf{z})$ is linearized in the exponential intermediate variables[10]

$$\left(\frac{z_\ell - \underline{\rho}}{\overline{\rho} - \underline{\rho}}\right)^a \tag{1.48}$$

with $a \leq 1$ around the current value of the design variables $\mathbf{z} = \mathbf{z}^0$. The first order Taylor expansion in these intermediate variables yields

$$f_{\text{app}}(\mathbf{z}) = f\left(\mathbf{z}^0\right) + \sum_{\ell=1}^{N} \frac{1}{a}\left(z_\ell^0 - \underline{\rho}\right)\left[\left(\frac{z_\ell - \underline{\rho}}{z_\ell^0 - \underline{\rho}}\right)^a - 1\right] \partial_\ell f\left(\mathbf{z}^0\right) \tag{1.49}$$

Observe that $f_{\mathsf{app}}(\mathbf{z})$ is a separable convex function if $\partial_\ell f(\mathbf{z}^0) \leq 0$ for all ℓ. This is the case for the compliance function as can be seen from the expression for its gradient given in equation (A.9) of the appendix. The approximate cost function $J_{\mathsf{app}}(\mathbf{z}) = f_{\mathsf{app}}(\mathbf{z}) + \lambda g(\mathbf{z})$ is also separable and so the minimization over cube $\left[\mathbf{z}^{\mathsf{L}}, \mathbf{z}^{\mathsf{U}}\right]$ reduces to N one-dimensional minimization problems for each component of $\mathbf{z}$. Using the convexity of J_{app}, the minimizer

[10] The significance of approximating response dependent cost functions in "reciprocal" variables, i.e., when $a = -1$, are discussed in [82] and references therein. For $a = 1$, one recovers the usual Taylor linearization.

$\mathbf{z}^{\mathsf{new}}$ is given by

$$z_\ell^{\mathsf{new}} = \begin{cases} z_\ell^{\mathsf{U}}, & z_\ell^* \geq z_\ell^{\mathsf{U}} \\ z_\ell^{\mathsf{L}}, & z_\ell^* \leq z_\ell^{\mathsf{L}} \\ z_\ell^*, & \text{otherwise} \end{cases}, \qquad \ell = 1, \ldots, N \tag{1.50}$$

where $\mathbf{z}^* = [z_\ell^*]_{\ell=1}^N$ is the stationary point of J_{app}, i.e., $\nabla J_{\mathsf{app}}(\mathbf{z}^*) = \mathbf{0}$. Upon substitution of (1.49) and (1.47), we can find an explicit expression for z_ℓ^*,

$$z_\ell^* = \underline{\rho} + \left[\frac{-\partial_\ell f(\mathbf{z}^0)}{\lambda \partial_\ell g(\mathbf{z}^0)}\right]^{\frac{1}{1-a}} \left(z_\ell^0 - \underline{\rho}\right), \qquad \ell = 1, \ldots, N \tag{1.51}$$

The update scheme consisting of (1.51) and (1.50) is commonly known as the Optimality Criteria (OC) method. The quantity $\zeta = 1/(1-a)$ is sometimes referred to as the damping coefficient. For the so-called reciprocal approximation, $a = -1$ and so $\zeta = 1/2$. This is the value usually used for compliance minimization since, on physical grounds, the reciprocal approximation is known to be accurate for compliance. For more information on this derivation, the reader is referred to [82].

1.6 Numerical results

We now present some numerical results for compliance minimization problems based on the density filtering formulation. All the results are obtained using the Matlab code `PolyTop` presented in the appendix. This code is based on the sizing formulation and optimization algorithm discussed in this chapter. In all cases, the Ersatz parameter ε was set to 10^{-4} and the Young's modulus and Poisson's ratio of the solid phase $\mathbf{C}$ were taken to be $E = 1$ and $\nu = 0.3$, respectively. The maximum tolerance for the change in design variables was taken to be 1%. Volume penalty parameter λ was adjusted in each iteration of the optimization algorithm to fix the volume fraction $|\Omega|^{-1} \int_\Omega \rho \mathrm{d}\mathbf{x}$ at the prescribed level $\overline{v}$.

The first example is the MBB beam problem [119] whose domain, loading and support conditions are shown in Figure 1.4(a). A mesh of 5,000 polygonal elements was generated using `PolyMesher` (see chapter 6 and the appendix). The final result shown in Figure 1.4(b) was obtained using a linear filter of radius 0.04 (the rectangular domain has unit height) and the SIMP model with continuation performed on the penalty parameter p as follows: the value of p was increased from 1 to 4 using increments of size 0.5 and for each value of p, a maximum of 150 iterations was allowed. Similarly, the RAMP functions [144, 24] defined by

$$m_E(\rho) = \varepsilon + (1-\varepsilon)\frac{\rho}{1+q(1-\rho)}, \quad m_V(\rho) = \rho \tag{1.52}$$

were used to generate the result shown in Figure 1.4(c). The parameter q was initially set

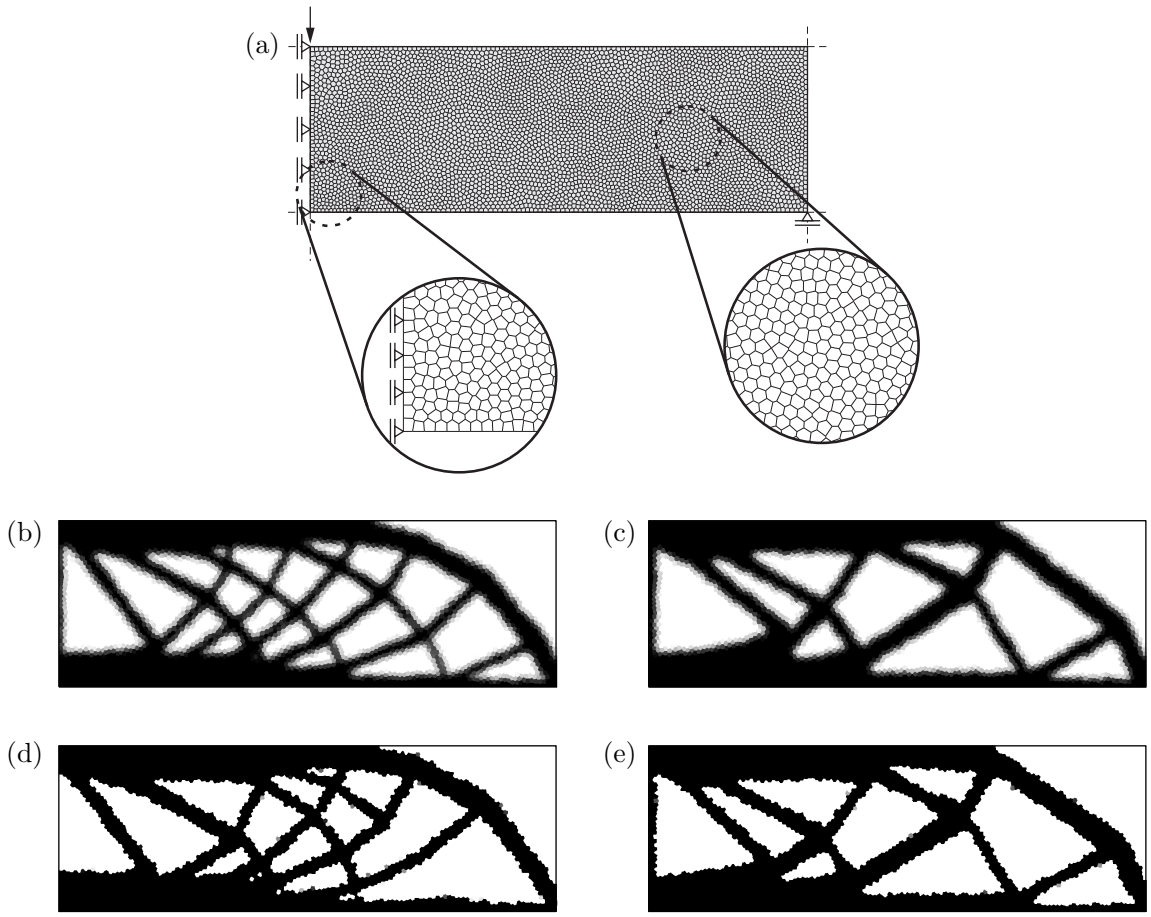


Figure 1.4: MBB beam problem (a) domain geometry, loading and boundary conditions; The mesh is composed of 5,000 elements (27 4-gons, 1,028 5-gons, 3,325 6-gons, 618 7-gons, and two 8-gons) and 9,922 nodes. Final topologies using (b) SIMP (c) RAMP (d) SIMP with Heaviside filtering (e) RAMP with Heaviside filtering for prescribed volume fraction $\overline{v} = 0.5$

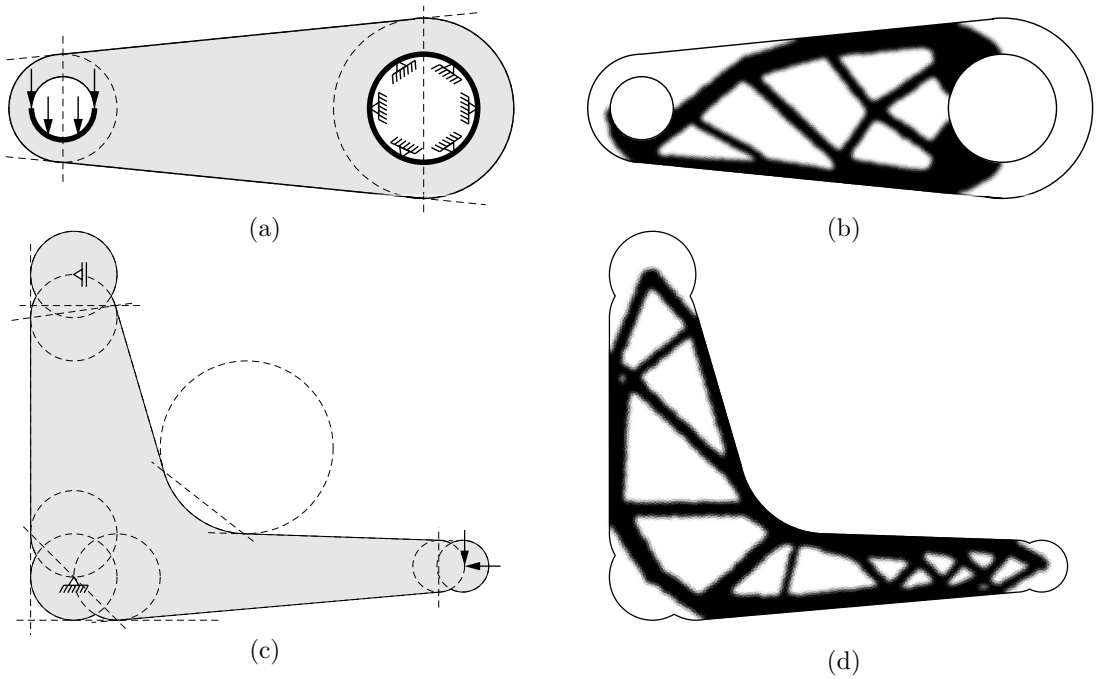


Figure 1.5: Compliance minimization problems with non-trivial domain geometries: (a) domain of wrench problem, $R = 0.03$, $\overline{v} = 0.4$; (b) final topology for wrench problem with RAMP functions; (c) domain of suspension triangle problem, $R = 0.25$, $\overline{v} = 0.45$, magnitude of horizontal load is eight times larger than the vertical load; (d) final topology for suspension problem with RAMP functions

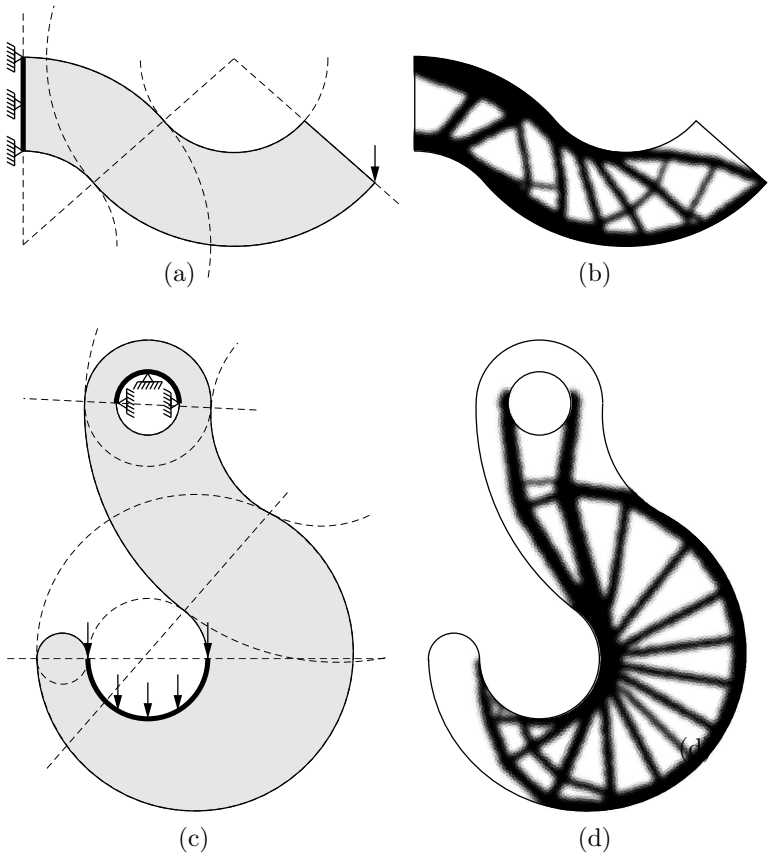


Figure 1.6: Compliance minimization problems with non-trivial domain geometries: (a) domain of serpentine beam problem, $R = 0.25$, $\overline{v} = 0.55$; (b) final topology for serpentine beam problem with SIMP functions; (c) domain of hook problem, $R = 2.0$, $\overline{v} = 0.40$; (d) final topology for hook beam problem with SIMP functions

to zero and continuation was subsequently carried out by doubling its value from 1 to 128. As the same filtering radius was used, the difference between the two results highlights the difference between the performance of these interpolation functions.

Yet another example of a material interpolation model is the Heaviside projection [85]. Even though it is typically considered a nonlinear filtering approach, it is easy to see that it can be cast in the present framework where the regularization map is linear. More specifically, if it is used in conjunction with SIMP, the material interpolation functions are given as

$$m_E(\rho) = \varepsilon + (1-\varepsilon)\,[h(\rho)]^p \tag{1.53}$$

$$m_V(\rho) = h(\rho) \tag{1.54}$$

where

$$h(x) = 1 - \exp(-\beta x) + x \exp(-\beta) \tag{1.55}$$

is the approximate Heaviside function and ρ belongs to the space of admissible designs defined in (1.14), that is, ρ is obtained from the usual linear filtering. This shows that *the Heaviside filtering essentially amounts to a modification of the material interpolation functions.* The additional parameter β controls the amount of grey scales that appears in the optimal solutions. Note, however, that SIMP penalization plays a crucial role since with $p = 1$, we have $m_E(\rho) \approx m_V(\rho)$ for any ρ and so the optimal solution will consist mostly of intermediate densities no matter how large β is. One can similarly define material interpolation functions for the Heaviside scheme based on the RAMP functions. Figures 1.4(d) and (e) show the Heaviside filtering results with SIMP and RAMP with the same penalty parameters and radius of filtering as in the previous results. Moreover the continuation of p and q were interleaved with the continuation on value of β as follows: the value of β was initially set to 1 and for each increment of p or q, it was doubled. As expected, the results depend on the manner in which the continuation of the penalty parameter and β is carried out.

The next set of results are for problems with non-trivial domain geometries or loading and support conditions. The design domains with the boundary conditions are shown in the left column of Figures 1.5 and 1.6. The wrench and suspension triangle were solved with RAMP functions while the hook and serpentine beam problems were solved with the SIMP interpolation function, both using the same continuation schemes as before. Even though the location of holes can be prescribed by means of passive elements in a uniform mesh (similar to what is done in the 99-line Maltab code [140]), it is evident that describing arbitrary geometries such as those shown here on a regular grid becomes cumbersome if not intractable. Moreover, the use of an unstructured mesh is necessary if one is to resolve the domain geometry, accurately specify the design loads and compute the structure's response. Problems of this sort typically arise in the practical applications, for example, the suspension triangle (cf. Figure 1.6(c)) is an industrial application of topology optimization presented in [7].

Next, we show how symmetry and similar layout constraints can be enforced in the present framework. We consider the wrench problem and wish to enforce symmetry along the horizontal axis. An unstructured but symmetric polygonal mesh was generated using `PolyMesher` (cf. chapter 6 and the appendix). For $\ell = 1, \ldots, N/2$, element $\Omega_{\ell+N/2}$ is the reflection of element Ω_ℓ about the horizontal axis. As discussed in section 1.3, symmetry can be enforced via the mapping $\mathcal{P}_s$ given in (1.20). For this mesh, the matrix associated with the discretization of $\mathcal{P}_s$ (cf. equation (1.38)) is defined by

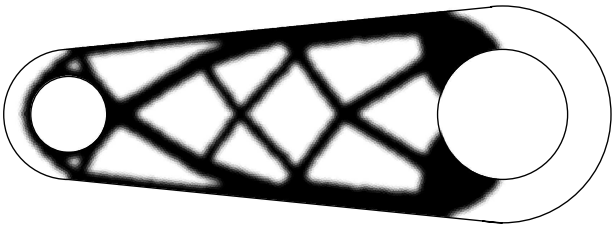


Figure 1.7: Symmetric solution to the wrench problem

$$[\mathbf{P}_s]_{\ell k} = \begin{cases} 1, & \text{if } \ell = k \text{ or } \ell = k + N/2 \\ 0, & \text{otherwise} \end{cases} \tag{1.56}$$

where $\ell = 1, \dots N$ and $k = 1, \dots, N/2$. To apply filtering as well, we set:

$$\mathbf{P} = \mathbf{P}_F \mathbf{P}_s \tag{1.57}$$

where $\mathbf{P}_F$ is the linear filtering matrix. Figure 1.7 shows the optimal wrench topology with symmetry enforced along the horizontal axis. This problem has asymmetric loading and imposing symmetry amounts to requiring the optimal design to also withstand identical upward loads. One can show that this symmetric topology is identical to the solution for the multiple load problem consisting of mirrored loading cases with equal weights.

Compliant mechanism design

The discussion so far and the above numerical examples have been limited to the minimum compliance problems. We next consider the design of compliant mechanisms, a problem whose cost functional is not self-adjoint and therefore, unlike compliance, the gradient field may take both negative and positive values in the domain. We begin with a brief description of the formulation of such problems within the framework of linear elasticity[11].

The objective of mechanism design is to identify a structure that maximizes the force exerted on a workpiece under the action of an external actuator. We assume in this setting that both the workpiece and the actuator are elastic and their stiffness are represented by vector fields $\mathbf{k}_1 \in L^\infty(\Gamma_{S_1})$ and $\mathbf{k}_2 \in L^\infty(\Gamma_{S_2})$, respectively. Here $\Gamma_{S_1}, \Gamma_{S_2}$ are segments of the traction boundary $\Gamma_N \subseteq \partial\Omega$ where the structure is interacting with these elastic bodies. The tractions experienced by the structure through this interaction for a displacement field $\mathbf{u}$ can be written as

[11] Since compliant mechanisms often undergo large deformations, linear elasticity may not be an accurate model of the response of the structure. Moreover, accounting for large deformation often leads to a different optimal design [24].

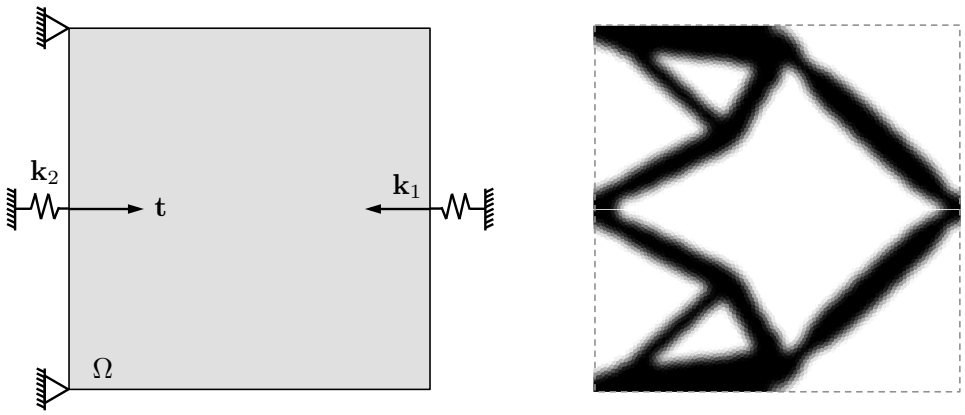


Figure 1.8: Force inverter mechanism: design domain and boundary conditions (left) and final solution (right)

$$\mathbf{t}_{S_r}(\mathbf{u}) = -\left(\mathbf{k}_r \cdot \mathbf{u}\right) \frac{\mathbf{k}_r}{|\mathbf{k}_r|} \qquad \text{on } \Gamma_{S_r} \text{ for } r = 1, 2 \tag{1.58}$$

Accordingly, the displacement $\mathbf{u}_\rho$ for a given distribution of material ρ in Ω is the solution to the following boundary problem

$$\int_\Omega m_E(\rho)\mathbf{C}\nabla\mathbf{u}_\rho : \nabla\mathbf{v}\mathrm{d}\mathbf{x} + \sum_{r=1,2}\int_{\Gamma_{S_r}} \frac{\left(\mathbf{k}_r \cdot \mathbf{u}_\rho\right)\left(\mathbf{k}_r \cdot \mathbf{v}\right)}{|\mathbf{k}_r|}\mathrm{d}s = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{v}\mathrm{d}s, \quad \forall \mathbf{v} \in \mathcal{V} \tag{1.59}$$

The cost functional for the mechanism design problem is defined as

$$J(\rho, \mathbf{u}_\rho) = -\int_{\Gamma_{S_1}} \mathbf{k}_1 \cdot \mathbf{u}_\rho \mathrm{d}s + \lambda \int_\Omega m_V(\rho)\mathrm{d}\mathbf{x} \tag{1.60}$$

where the second term again represents a constraint on the volume of the design. The first term of this objective is a measure of the (negative of) force applied to the workpiece in the direction of $\mathbf{k}_1$ which can be seen from the following relation:

$$\int_{\Gamma_{S_1}} \mathbf{k}_1 \cdot \mathbf{u}_\rho \mathrm{d}s = \int_{\Gamma_{S_1}} \left[-\mathbf{t}_{S_1}(\mathbf{u}_\rho)\right] \cdot \frac{\mathbf{k}_1}{|\mathbf{k}_1|}\mathrm{d}s \tag{1.61}$$

Viewed another way, the minimization of the first term of (1.60) amounts to maximizing the displacement of the structure at the location of the workpiece in the direction of $\mathbf{k}_1$.

The discrete counterpart to minimization of $J(\rho, \mathbf{u}_\rho)$ over the admissible space $\mathcal{A}$ defined by (1.23), obtained through the discretization procedure of section 1.4, can be written as

$$\min_{\mathbf{z} \in \left[\underline{\rho}, \overline{\rho}\right]^N} J(\mathbf{z}) = -\mathbf{L}^T\mathbf{U}(\mathbf{z}) + \lambda \mathbf{A}^T m_V(\mathbf{P}\mathbf{z}) \tag{1.62}$$

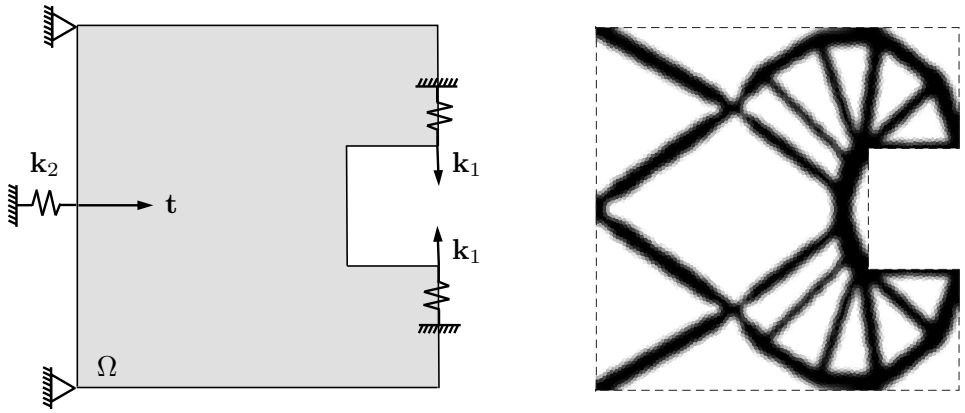


Figure 1.9: Gripper mechanism: design domain and boundary conditions (left) and final solution (right)

where $[\mathbf{L}]_i = \int_{\Gamma_{S_1}} \mathbf{k}_1 \cdot \mathbf{N}_i \mathrm{d}s$ and $\mathbf{U}(\mathbf{z})$ solves

$$[\mathbf{K}(\mathbf{z}) + \mathbf{K}_s]\,\mathbf{U} = \mathbf{F} \tag{1.63}$$

Here $\mathbf{K}(\mathbf{z})$ and $\mathbf{F}$ as defined before in (1.31) and (1.30) and

$$[\mathbf{K}_s]_{ij} = \sum_{r=1,2} \int_{\Gamma_{S_r}} \frac{(\mathbf{k}_r \cdot \mathbf{N}_i)\,(\mathbf{k}_r \cdot \mathbf{N}_j)}{|\mathbf{k}_r|} \mathrm{d}s \tag{1.64}$$

which, we note, is independent of the design $\mathbf{z}$.

It can be seen from the expression of the gradient of $f(\mathbf{z}) = -\mathbf{L}^T\mathbf{U}(\mathbf{z})$ that it can be both positive and negative (see eq. (A.10) in the appendix) and therefore the exponential approximation (1.49) may not be convex. One alternative is to carry out the linearization in *mixed* intermediate variables. For example, the CONLIN method [77] uses reciprocal intermediate variables when $\partial_\ell f < 0$ and a linear expansion for the remaining terms. As in (1.49), this strategy again leads to a convex separable approximation. A similar approach is adopted in the Method of Moving Asymptotes [153], which we used to generate the numerical results for the mechanism design examples shown in Figures 1.8 and 1.9. In both examples, $|\mathbf{k}_1| = |\mathbf{k}_2| = 1$, $R = 0.02$ and the width of the design domain is one.

1.7 Outline of the thesis

The remainder of this thesis is organized as follows. In the next chapter, we present some theoretical results establishing the well-posedness of a large class of restriction formulations. In particular, we discuss a sufficient compactness condition for existence of solutions and investigate its theoretical and practical consequences for density and implicit function parametrizations. We also provide a proof for the validity of the Ersatz approximation

within the restriction setting. In chapter 3, we focus on Tikhonov and total variation regularization of the density-based compliance minimization problem and develop an optimization algorithm using the concept of operator splitting. The algorithm is further analyzed and refined in chapter 4 and a connection is made with the heuristic sensitivity filtering method. We also provide a convergence proof for a standard finite element approximation of the Tikhonov-regularized problem. In chapter 5, we shift our attention to the choice of discretization spaces and in particular investigate the use of unstructured polygonal meshes with Wachspress-type interpolation. These discretizations are not susceptible to numerical instabilities such as checkerboard patterns that can appear for compliance minimization problem and mixed variational formulation of the incompressible Stokes equation. In the latter context, we demonstrate that polygonal discretizations satisfy the well-known Babuska-Brezzi conditions. We also discuss and present results for topology optimization problems for incompressible Stokes flow. Next, in chapter 6, we develop a general-purpose meshing algorithm that can generate unstructured convex polygonal meshes using the concept of Voronoi diagrams. Concluding remarks and potential extensions of the work are given in chapter 7. The appendix includes educational Matlab codes that feature self-contained discretization and analysis routines using polygonal finite elements. Also, an implementation of general topology optimization framework based on the material in this introductory chapter is presented.

The work in this dissertation has resulted in publications [162, 163, 164, 160, 159]. Additional manuscripts based on the material in chapter 2 and 5 are currently under preparation.

Chapter 2

Theoretical Basis for Restriction Formulations

In this chapter, we examine the ill-posedness of the continuum topology optimization problem and discuss elements of a well-posed restriction formulation. We limit our attention mainly to the two-phase material distribution problem since, as discussed before, the Ersatz approximation of the single-phase shape optimization leads to such a problem. However, a result is presented at the end of the chapter regarding the theoretical justification of the Ersatz model and the connection between the two classes of problems in the degenerate limits of the compliant phase. To illustrate the inherent tendency of the classical problem to produce rapidly oscillating designs, a simple counterexample for the compliance design is discussed. This counterexample is used later to elaborate on the significance of the regularity conditions that can be imposed on the implicit functions to ensure existence of solutions. We present and prove a sufficient compactness condition that guarantees existence of solutions for the two-phase problems, a result that serves as the basis for the analysis of various restriction formulations presented in this thesis.

The latter part of the chapter is devoted to a formulation based on implicit functions that satisfies this compactness condition. Recently, there has been great interest in implicit function formulations for solving topology optimization problems, but often the ill-posedness of the continuum problem is neglected in the construction of the algorithms. Since the inherent pathology of the topology optimization problem is its tendency to produce highly oscillatory shapes, a uniform smoothness condition for the implicit functions is naturally expected. What is perhaps less obvious is a "transversality" condition that requires implicit functions to be steep around the zero level set. We will demonstrate numerically that one obtains smeared designs (i.e., optimal implicit functions having values close to zero throughout most of the design domain) when a smooth Heaviside function maps the implicit function to the coefficients of the state equation. In fact, we will show that the use of the smeared Heaviside transforms the problem into the variable thickness problem. In light of these theoretical considerations, we will discuss several existing formulations and provide possible explanations for the some of the heuristics employed.

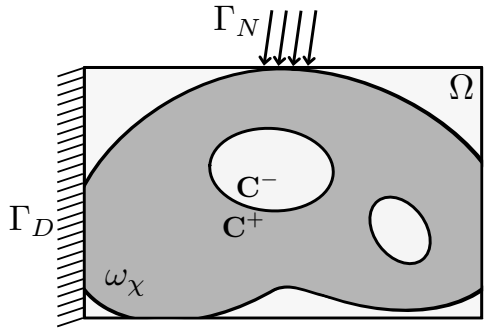


Figure 2.1: Illustration of the boundary value problem for the two-phase material distribution problem

2.1 Two-phase material distribution problem

Consider two linear elastic isotropic materials with elasticity tensors $\mathbf{C}^+$ and $\mathbf{C}^-$ such that $\mathbf{C}^- \leq \mathbf{C}^+$ in the sense of quadratic forms. We refer to $\mathbf{C}^+$ and $\mathbf{C}^-$ as the *stiff* and *compliant* phases, respectively. In the two-phase material distribution problem, the goal is to fill a given design domain Ω by an optimal arrangement of these material. In particular, we identify the optimal region $\omega \subseteq \Omega$ occupied by the phase $\mathbf{C}^+$ that provides the best performance of the mixture. The remainder, $\Omega \backslash \omega$, will be filled by $\mathbf{C}^-$(see Figure 2.1). Since *a priori* there are no restrictions on regularity of ω, we only require that it is measurable. Thus, we define the *classical space of admissible shapes* based on the characteristic functions associated with each set as

$$\mathcal{A}_C = L^\infty(\Omega; \{0, 1\}) \tag{2.1}$$

Given $\chi \in \mathcal{A}_C$, the overall elasticity tensor in Ω, corresponding to the distribution of phases defined by χ, can be written as

$$\mathbf{C}_\chi = \chi \mathbf{C}^+ + (1 - \chi)\, \mathbf{C}^- \tag{2.2}$$

The *response* of such a mixture is characterized by the solution to the following boundary value problem defined over Ω

$$\begin{aligned} \operatorname{div}\left[\mathbf{C}_\chi \boldsymbol{\epsilon}(\mathbf{u})\right] &= \mathbf{0} \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \Gamma_D \\ \left[\mathbf{C}_\chi \boldsymbol{\epsilon}(\mathbf{u})\right] \cdot \mathbf{n} &= \mathbf{t} \quad \text{on } \Gamma_N \end{aligned} \tag{2.3}$$

Here $\mathbf{u}$ is the displacement field; $\boldsymbol{\epsilon}(\mathbf{u}) = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$ is the linearized strain tensor; Γ_D and Γ_N form a partition of $\partial\Omega$ (i.e., $\Gamma_D \cap \Gamma_N = \emptyset$ and $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N$) and unless otherwise

stated, it is assumed that Γ_D has non-zero surface measure; $\mathbf{n}$ is the unit normal vector to $\partial\Omega$; and finally $\mathbf{g} \in H^{1/2}(\Gamma_D)^d$ and $\mathbf{t} \in L^2(\Gamma_N)^d$ are the displacement and traction boundary data, respectively. This boundary value problem has a unique weak solution $\mathbf{u}_\chi \in \mathcal{U}$, which solves the variational problem

$$a(\mathbf{u}_\chi, \mathbf{v}; \chi) = \ell(\mathbf{v}), \ \forall \mathbf{v} \in \mathcal{V} \tag{2.4}$$

Here $\mathcal{V} = \{\mathbf{u} \in H^1(\Omega)^d : \mathbf{u}|_{\Gamma_D} = \mathbf{0}\}$ is the space of admissible kinematic variations, $\mathcal{U} = \mathbf{g}' + \mathcal{V}$ where $\mathbf{g}' \in H^1(\Omega)^d$ such that $\mathbf{g}' = \mathbf{g}$ on Γ_D (in the sense of trace), and

$$a(\mathbf{u}, \mathbf{v}; \chi) = \int_\Omega \mathbf{C}_\chi \boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x}, \quad \ell(\mathbf{v}) = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{u} \mathrm{d}s \tag{2.5}$$

are the energy bilinear and load linear forms. The linear and bilinear forms are continuous and the $\mathcal{V}$-ellipticity of the bilinear form follows from Korn's inequality and positive-definiteness of the elasticity tensors (cf. [49]). Thus there exist strictly positive constants M and c such that for all $\chi \in L^\infty(\Omega; [0,1])$,

$$|a(\mathbf{u}, \mathbf{v}; \chi)| \leq M \|\mathbf{u}\|_{1,2,\Omega} \|\mathbf{v}\|_{1,2,\Omega}, \ \forall \mathbf{u}, \mathbf{v} \in H^1(\Omega)^d \tag{2.6}$$

and

$$a(\mathbf{v}, \mathbf{v}; \chi) \geq c \|\mathbf{v}\|^2_{1,2,\Omega}, \ \forall \mathbf{v} \in \mathcal{V}. \tag{2.7}$$

That the linear form $\ell : \mathcal{V} \to \mathbb{R}$ is bounded follows from the regularity assumptions on the applied tractions $\mathbf{t}$. Not only do these properties show the existence of a weak solution to (2.3), they can also be used to establish boundedness of a sequence of solutions to the state equation. Indeed for any $\chi \in L^\infty(\Omega; [0,1])$ and the corresponding state solution $\mathbf{u}_\chi$, we have

$$\begin{aligned} c \|\mathbf{u}_\chi - \mathbf{g}'\|^2_{1,2,\Omega} &\leq a(\mathbf{u}_\chi - \mathbf{g}', \mathbf{u}_\chi - \mathbf{g}'; \chi) \\ &= a(\mathbf{u}_\chi, \mathbf{u}_\chi - \mathbf{g}'; \chi) - a(\mathbf{g}', \mathbf{u}_\chi - \mathbf{g}'; \chi) \\ &\leq \ell(\mathbf{u}_\chi - \mathbf{g}') + M \|\mathbf{g}'\|_{1,2,\Omega} \|\mathbf{u}_\chi - \mathbf{g}'\|_{1,2,\Omega} \\ &\leq \left(\|\ell\| + M \|\mathbf{g}'\|_{1,2,\Omega} \right) \|\mathbf{u}_\chi - \mathbf{g}'\|_{1,2,\Omega} \end{aligned} \tag{2.8}$$

which implies

$$\|\mathbf{u}_\chi\|_{1,2,\Omega} \leq \|\mathbf{u}_\chi - \mathbf{g}'\|_{1,2,\Omega} + \|\mathbf{g}'\|_{1,2,\Omega} \leq \frac{1}{c} \|\ell\| + \left(1 + \frac{M}{c}\right) \|\mathbf{g}'\|_{1,2,\Omega} \tag{2.9}$$

Observe that the right hand side is independent of χ.

The performance of each candidate design is measured by an *objective or cost function*

$J : \mathcal{A}_C \times \mathcal{U} \to \mathbb{R}$, which we assume satisfies the following continuity requirements:

$$J(\cdot, \mathbf{u}) : L^\infty(\Omega; [0,1]) \to \mathbb{R} \qquad \text{is strongly continuous in } L^1(\Omega) \tag{2.10}$$

$$J(\chi, \cdot) : \mathcal{U} \to \mathbb{R} \qquad \text{is strongly continuous in } H^1(\Omega)^d \tag{2.11}$$

Note that since Ω has finite measure and all the functions in $L^\infty(\Omega; [0,1])$ are bounded, strong convergence in $L^1(\Omega)$ is equivalent to strong convergence in $L^p(\Omega)$ for any $1 \leq p < \infty$. We also remark that these continuity requirements are relatively weak and are met by most objective functions of interest. For example, the objective function for the minimum compliance problem is of the form

$$J(\chi, \mathbf{u}_\chi) = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{u}_\chi \mathrm{d}s - \int_{\Gamma_D} \mathbf{g} \cdot \left[\mathbf{C}_\chi \boldsymbol{\epsilon}\left(\mathbf{u}_\chi\right)\right] \mathrm{d}s + \lambda \int_\Omega \chi \mathrm{d}\mathbf{x} \tag{2.12}$$

The last term in J represents a penalty on the volume of the stiff material used. Minimizing this objective amounts to finding the stiffest arrangement of the two phases while using the least amount of the solid phase. Note that the first term in (2.12) minimizes the displacement under the applied traction $\mathbf{t}$ and the second term maximizes the reactions under applied displacement $\mathbf{g}$. The parameter λ determines the trade-off between the stiffness provided by the stiff material and the amount that is used.

Similarly, given a target (measured) displacement $\mathbf{z} \in L^2(E)^d$ with $E \subseteq \Omega$, and knowledge of the boundary conditions imposed on Ω, one may wish to recover the distribution of material inside the domain. A suitable objective function for this problem is

$$J(\chi, \mathbf{u}_\chi) = \int_E \left|\mathbf{u}_\chi - \mathbf{z}\right|^2 \mathrm{d}\mathbf{x} \tag{2.13}$$

Finally, one can minimize the average effective stress in a structure made of limited amount of stiff phase by choosing the objective function as

$$J(\chi, \mathbf{u}_\chi) = \int_\Omega \left(\left|\sigma_{\mathsf{e}}(\chi, \mathbf{u}_\chi)\right|^2 + \lambda\chi\right) \mathrm{d}\mathbf{x} \tag{2.14}$$

Here $\sigma_{\mathsf{e}}(\chi, \mathbf{u})$ is some effective measure of stress, e.g., the von Mises stress given by

$$\sigma_{\mathsf{e}} = \sqrt{\frac{3}{2}\mathbf{s} : \mathbf{s}}, \quad \mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I}, \quad \boldsymbol{\sigma} = \mathbf{C}_\chi \boldsymbol{\epsilon}(\mathbf{u}_\chi) \tag{2.15}$$

The classical two-phase optimal shape problem $\mathbb{P}_C$ is defined by

$$\inf_{\chi \in \mathcal{A}_C} J(\chi, \mathbf{u}_\chi) \tag{2.16}$$

As mentioned before, this problem, in general, suffers from non-existence of solutions.

2.2 Implicit function description

The space of characteristic functions $\mathcal{A}_C$ for the material distribution problem can be defined in terms of implicit or embedding functions with the aid of the Heaviside map H:

$$H : \varphi \longmapsto \chi_{\{\varphi>0\}} \tag{2.17}$$

In particular, we can define the function space

$$\mathcal{F}_C = L^\infty(\Omega; [-\alpha, \alpha]) \tag{2.18}$$

where $\alpha > 0$ is a fixed constant. Then any $\chi \in \mathcal{A}_C$ can be written as $H(\varphi)$ for some $\varphi \in \mathcal{F}_C$ (in fact, one can choose $\varphi = \chi$ itself). It is obvious from the definition of H that $H(\varphi)$ is a characteristic function for any $\varphi \in \mathcal{F}_C$. In short, we can write the classical space as

$$\mathcal{A}_C = H(\mathcal{F}_C) \tag{2.19}$$

The motivation for formulating the problem in the language of implicit functions is twofold: a class of well-posed optimal design problems can be obtained by placing constraints on the function space $\mathcal{F}_C$. One such restriction approach is outlined later in this chapter in section 2.7. Moreover, this continuous parameterization is amenable to usual gradient-based optimization algorithms provided that the Heaviside operator is approximated by a smooth function. In addition to implicit function formulations, level set methods are also based on such characterization of shapes.

2.3 Counterexample to existence of solutions

We now describe a simple counterexample, borrowed from [3], for the minimum compliance problem in order to highlight the inherent pathology of the optimal shape problems, in particular, the tendency of the minimizing sequences to rapidly oscillate. We will later use this counterexample to illustrate the significance of the constraints in the restriction framework of section 2.7. A class of counterexamples for minimization of the Kohn-Strang functional with affine boundary conditions is analyzed in [5, 57]. For the sake of simplicity, it is assumed that both phases have zero Poisson's ratio.

Consider the minimum compliance problem with pure uniaxial tractions given by

$$J(\chi, \mathbf{u}_\chi) = \ell(\mathbf{u}_\chi) + \lambda_{\frac{1}{2}} \int_\Omega \chi \mathrm{d}\mathbf{x}, \quad \Gamma_D = \emptyset, \quad \mathbf{t} = \boldsymbol{\sigma}_0 \cdot \mathbf{n}, \quad \boldsymbol{\sigma}_0 = t_0(\mathbf{e}_d \otimes \mathbf{e}_d) \tag{2.20}$$

where $\mathbf{e}_d$ is the dth unit basis vector, t_0 is a positive constant, and the penalty parameter

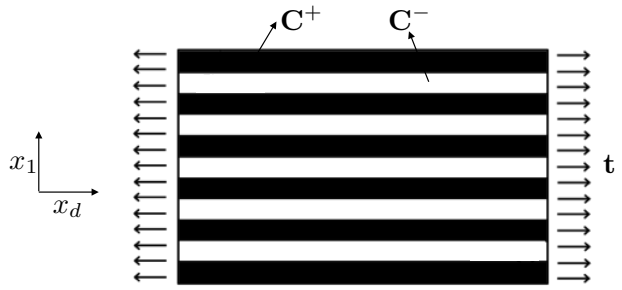


Figure 2.2: Illustration of the arrangement of the two phases in the minimizing sequence (2.33)

has value:

$$\lambda_{\frac{1}{2}} = \frac{2t_0^2\left(\mu^+ - \mu^-\right)}{\left(\mu^+ + \mu^-\right)^2} \tag{2.21}$$

Here μ^+ and μ^- denote the shear moduli of $\mathbf{C}^+$ and $\mathbf{C}^-$, respectively. As we shall see in a moment, with this value of penalty parameter, the optimal behavior of the system is such that it requires equal volume of the two phases. Note further that prescribed tractions $\mathbf{t}$ satisfy the compatibility condition $\int_{\partial\Omega} \mathbf{t}\cdot\mathbf{w}\mathrm{d}s = 0$ where $\mathbf{w}$ is rigid body motion $\mathbf{w}$. It follows that, even though $\Gamma_D = \emptyset$, (2.3) admits a solution unique up to rigid body motions.

We prove the lack of existence of solutions in $\mathcal{A}_C$ in three steps. First we establish a lower bound for $J(\chi, \mathbf{u}_\chi)$ over $\mathcal{A}_C$. We then show that this lower bound *cannot* be achieved by any $\chi \in \mathcal{A}_C$. Finally, with the aid of homogenization theory, we construct a sequence of classical shapes in $\mathcal{A}_C$ "converging" to a composite design that is optimal (i.e., has performance equal to this lower bound). The intuition is that distributing the stiff phase in strips along the direction of the applied tractions yields stiffer designs (see Figure 2.2). The thinner these strips, the more efficient the arrangement. The limit of this process is a composite of the two phases with a laminated microstructure. Mathematically speaking, the minimizing sequence corresponding to this construction is not convergent in the space $\mathcal{A}_C$. The sequence of shapes associated with this minimizing sequence does not converge to a classical shape but rather to a generalized composite.

Using the principle of minimum complementary energy, we can write the compliance functional as

$$\ell(\mathbf{u}_\chi) = \inf_{\boldsymbol{\tau}\in H_0} \int \mathbf{C}_\chi^{-1}\boldsymbol{\tau} : \boldsymbol{\tau}\mathrm{d}\mathbf{x} \tag{2.22}$$

where the space of admissible stresses H_0 consists of stress fields that are divergent-free and satisfy the boundary conditions, i.e.,

$$H_0 = \left\{\boldsymbol{\tau} \in L^2(\Omega)^{d\times d} : \mathrm{div}\boldsymbol{\tau} = \mathbf{0} \text{ in } \Omega, \boldsymbol{\tau}\cdot\mathbf{n} = \mathbf{t} \text{ on } \Gamma_N\right\} \tag{2.23}$$

The "quadratic" integrand of (2.22) is convex at each $\mathbf{x} \in \Omega$ and so its value is larger than

the tangent hyperplane at $(\langle \mathbf{C}_\chi \rangle, \boldsymbol{\sigma}_0)$. In particular, one can write for any $\boldsymbol{\tau} \in H_0$:

$$\begin{aligned} \mathbf{C}_\chi(\mathbf{x})^{-1}\boldsymbol{\tau}(\mathbf{x}) : \boldsymbol{\tau}(\mathbf{x}) \;\geq\; & \langle \mathbf{C}_\chi \rangle^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 - \langle \mathbf{C}_\chi \rangle^{-1} (\mathbf{C}_\chi(\mathbf{x}) - \langle \mathbf{C}_\chi \rangle) \langle \mathbf{C}_\chi \rangle^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 \\ & + 2 \langle \mathbf{C}_\chi \rangle^{-1} \boldsymbol{\sigma}_0 : (\boldsymbol{\tau}(\mathbf{x}) - \boldsymbol{\sigma}_0) \end{aligned} \tag{2.24}$$

Integrating both sides over Ω and noting $\int_\Omega \boldsymbol{\tau} \mathrm{d}\mathbf{x} = |\Omega| \, \boldsymbol{\sigma}_0$, we get

$$\int \mathbf{C}_\chi^{-1} \boldsymbol{\tau} : \boldsymbol{\tau} \mathrm{d}\mathbf{x} \geq |\Omega| \langle \mathbf{C}_\chi \rangle^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 \tag{2.25}$$

In turn, this implies

$$\inf_{\chi \in \mathcal{A}_C} \inf_{\boldsymbol{\tau} \in H_0} \int \left(\mathbf{C}_\chi^{-1} \boldsymbol{\tau} : \boldsymbol{\tau} + \lambda \chi \right) \mathrm{d}\mathbf{x} \geq |\Omega| \inf_{\chi \in \mathcal{A}_C} \left(\langle \mathbf{C}_\chi \rangle^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 + \lambda_{\frac{1}{2}} \langle \chi \rangle \right) \tag{2.26}$$

Noting that $\langle \mathbf{C}_\chi \rangle = \mathbf{C}^+ \langle \chi \rangle + \mathbf{C}^{-1} (1 - \langle \chi \rangle)$, we can see that the right hand side is only a function of $\langle \chi \rangle$. Moreover, its minimum value over the range $\langle \chi \rangle \in [0, 1]$ is given by

$$L = \frac{2\mu^+ t_0^2 |\Omega|}{(\mu^+ + \mu^-)^2} \tag{2.27}$$

We next show that no classical shape can realize this lower bound L. Suppose, to the contrary, that $\overline{\chi} \in \mathcal{A}_C$ has performance equal to this value. Then the remainder in the equation (2.24) should vanish at almost every point in the domain. So for almost $\mathbf{x} \in \Omega$,

$$\mathbf{C}_{\overline{\chi}}(\mathbf{x})^{-1} \left(\boldsymbol{\sigma}_{\overline{\chi}}(\mathbf{x}) - \mathbf{C}_{\overline{\chi}}(\mathbf{x}) \langle \mathbf{C}_{\overline{\chi}} \rangle^{-1} \boldsymbol{\sigma}_0 \right) : \left(\boldsymbol{\sigma}_{\overline{\chi}}(\mathbf{x}) - \mathbf{C}_{\overline{\chi}}(\mathbf{x}) \langle \mathbf{C}_{\overline{\chi}} \rangle^{-1} \boldsymbol{\sigma}_0 \right) = 0 \tag{2.28}$$

Here $\boldsymbol{\sigma}_{\overline{\chi}}$ denotes the stress field corresponding to $\overline{\chi}$, i.e., $\boldsymbol{\sigma}_{\overline{\chi}} = \mathbf{C}_{\overline{\chi}} \boldsymbol{\epsilon}(\mathbf{u}_{\overline{\chi}})$. Then we must have

$$\boldsymbol{\sigma}_{\overline{\chi}}(\mathbf{x}) = \mathbf{C}_{\overline{\chi}}(\mathbf{x}) \langle \mathbf{C}_{\overline{\chi}} \rangle^{-1} \boldsymbol{\sigma}_0 \tag{2.29}$$

and so the stress field can only take values of $\mathbf{C}^+ \langle \mathbf{C}_{\overline{\chi}} \rangle^{-1} \boldsymbol{\sigma}_0$ or $\mathbf{C}^- \langle \mathbf{C}_{\overline{\chi}} \rangle^{-1} \boldsymbol{\sigma}_0$, which are both different from the boundary value $\boldsymbol{\sigma}_0$[1].

Now suppose Ω is filled by a rank-1 laminate made up of $\mathbf{C}^+$ and $\mathbf{C}^-$ in *equal* amounts and with the laminations in direction of $\mathbf{e}_1$. The (constant) stiffness tensor $\mathbf{C}^*$ satisfies[2]

$$(\mathbf{C}^*)^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 = \frac{t_0^2}{\mu^+ + \mu^-} \tag{2.30}$$

[1] Unless, of course, $\overline{\chi}$ is a constant function. But it is easy to verify that for $\overline{\chi} \equiv 1$ or $\overline{\chi} \equiv 0$, we have $J(\overline{\chi}) > L$.

[2] For any symmetric second order tensor $\boldsymbol{\xi}$ such that $\boldsymbol{\xi}\mathbf{e}_1 = \mathbf{0}$, we have $\mathbf{C}^* \boldsymbol{\xi} = \frac{1}{2} \left(\mathbf{C}^+ + \mathbf{C}^- \right) \boldsymbol{\xi} = \left(\mu^+ + \mu^- \right) \boldsymbol{\xi}$

since $\boldsymbol{\sigma}_0 \cdot \mathbf{e}_1 = \mathbf{0}$. Moreover, it is not difficult to show that

$$\boldsymbol{\sigma}_0 = \text{argmin}_{\boldsymbol{\tau} \in H_0} \int_\Omega \left(\mathbf{C}^*\right)^{-1} \boldsymbol{\tau} : \boldsymbol{\tau} \mathrm{d}\mathbf{x} \tag{2.31}$$

and so value of the objective function for this design is equal to the lower bound

$$J^* = |\Omega| \left(\mathbf{C}^*\right)^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 + \frac{1}{2} |\Omega| \lambda_{\frac{1}{2}} = \frac{|\Omega| t_0^2}{\mu^+ + \mu^-} + \frac{1}{2} |\Omega| \frac{2t_0^2 \left(\mu^+ - \mu^-\right)}{\left(\mu^+ + \mu^-\right)^2} = L \tag{2.32}$$

Now consider the following sequence of characteristic functions

$$\chi_n = H(\varphi_n), \quad \varphi_n(\mathbf{x}) = \alpha \sin(n x_1) \tag{2.33}$$

The weak* limit of this sequence in $L^\infty(\Omega)$ is the constant function $\rho \equiv 1/2$, and so $\int_\Omega \chi_n \mathrm{d}\mathbf{x} \to |\Omega|/2$. Moreover, $\mathbf{C}^*$ is the so-called $G-$limit of the sequence of elasticities $\mathbf{C}_{\chi_n}$ (the definition is given in the next section), which implies the convergence of energies

$$\mathbf{C}_{\chi_n}^{-1} \boldsymbol{\sigma}_{\chi_n} : \boldsymbol{\sigma}_{\chi_n} \to \left(\mathbf{C}^*\right)^{-1} \boldsymbol{\sigma}_0 : \boldsymbol{\sigma}_0 \tag{2.34}$$

Therefore, we have $J(\chi_n, \mathbf{u}_{\chi_n}) \to L$, and so L is indeed the infimum of J over $\mathcal{A}_C$ even though no element of $\mathcal{A}_C$ can achieve it.

2.4 Relaxation

The details of the proof of convergence of energy in (2.34) is outside the scope of this document, but it is important to note how the mathematical theory of homogenization appears in discussion of the optimal shape problem. We recall the definition of $G-$convergence, which provides the theoretical basis for the *relaxation* of optimal shape problem $\mathbb{P}_C$. The relaxation of an ill-posed variational problem consists of finding a well-posed problem (one that admits a solution) that is "equivalent" to the original problem in the sense that the two problems have the same infima, and moreover, the solutions of the relaxed problem are precisely the (weak) limits of the minimizing sequences of the original problem [96].

A sequence of symmetric elasticity tensors $\mathbf{C}_n$ $G-$converges to tensor $\mathbf{C}^*$ provided that

$$\mathbf{u}_n \rightharpoonup \mathbf{u}^* \qquad \text{weakly in } H^1(\Omega)^d \tag{2.35}$$

where $\mathbf{u}_n$ and $\mathbf{u}^*$ are solutions to the boundary value problem (2.3) with $\mathbf{C}_n$ and $\mathbf{C}^*$ as the elasticity tensor. The relaxation of $\mathbb{P}_C$ consists of enlarging the space of admissible domains $\mathcal{A}_C$ to include all the weak limits of sequence χ_n and the $G-$limits of the associated elasticity tensors $\mathbf{C}_{\chi_n}$ in the state equation. In short, this relaxation approach amounts

to replacing $\mathcal{A}_C$ by its $G-$closure (and accordingly modifying the state equation and the objective function). The composite designs in this larger space encode the behavior of the minimizing sequences of the classical problem.

We refer the reader to the extensive literature on theory and numerics of relaxation, for example, [96, 4, 92, 107, 22]. Despite significant advances in this framework, an explicit characterization of the $G-$closure of $\mathcal{A}_C$ is not yet known, and so "true" relaxation of $\mathbb{P}_C$ is limited to a certain class of objective functions such as compliance and eigenvalue minimization problems. For these problems, restricting the space of composites to the sequential laminates is known to be sufficient to obtain the full relaxation.

Another noteworthy connection is with density formulations discussed in chapter 1 of this thesis. As mentioned there, a density function can be viewed as measuring the volume fraction of the two phases at each point in the given composite design. The material models such as SIMP describe the relationship between the volume fraction and the stiffness at each point. Effectively this amounts to solving the problem over a restricted class of composites that obey the material model (the microstructures of the two-phase composites that follow the SIMP power relation have been numerically computed in [23]). For this reason, these formulations are sometimes referred to as *partial relaxation*. However, note that the partially relaxed problem is again ill-posed unless some restrictions are placed on the variation of the density fields. For this reason, in chapter 1, we provided an alternative explanation of such density formulations based on approximation of characteristic functions that make more sense both from theoretical and practical perspective.

2.5 Compactness condition

The key idea of *restriction* is to replace the space of admissible designs by a strictly smaller space that does not allow for oscillations of the minimizing sequence favored by the optimization problem. In particular, one must identify a sufficiently regular subset $\mathcal{A}_R \subseteq \mathcal{A}_C$ in order to guarantee existence of solutions. Accordingly, the "restricted" optimal design problem $\mathbb{P}_R$ is defined as

$$\inf_{\chi \in \mathcal{A}_R} J(\chi, \mathbf{u}_\chi) \tag{2.36}$$

Before proceeding to the central result, few remarks are in order about the philosophical justification of the restriction approaches. For many inverse problems that arise in the context of shape optimization (e.g., problem (2.13)), one has some *a priori* knowledge of the unknown shape and its regularity. So it may not make sense, as is the case with relaxation, to allow rapid oscillations only to retain the infimum of the objective function. Similarly, in some engineering applications such as structural design, the manufacturing requirements place some constraints on the space of admissible geometries that can be

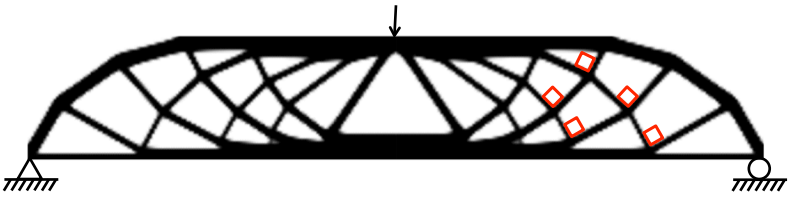


Figure 2.3: Low volume fraction solution to the MBB beam problem based on Tikhonov regularization. Notice that this solution exhibits the orthogonality of tension and compression members characteristic of Michell optimal frame layouts (cf. [21, 104, 103, 143])

built. Such practical considerations as well as one's understanding of the nature of the unknown solutions are often enough to sufficiently restrict the space of admissible shapes. Furthermore, in some cases, one can demonstrate that despite the artificial effects of the restriction schemes, the characteristics of the original ill-posed problem (e.g., orthogonality of Michell networks—see Figure 2.3) can still appear in the solutions in the restricted space. Finally, restriction provides a natural context for the validity of the Ersatz approximation, i.e., relating the one-phase shape optimization problem to the material distribution problem in the limit of vanishing weak phase. Relaxation faces some technical difficulties since the theory of homogenization is inoperative in the degenerate regime. We should mention, however, the work by Allaire et al. [4] on the relaxation of single-phase minimum compliance problem, which relies on a generalized notion of compliance defined over $\mathcal{A}_C$. Even in such a limited case, only partial results connecting the two classes of problems are obtained. In summary, while the restriction approach may seem like an arbitrary modification of the original problem one is set out to solve, it does have some justification from both theoretical and practical points of view.

The main issue in the analysis of existence of minimizers is that the cost function J depends implicitly on χ through the solution of the state equation. The key result in this chapter, given by the following proposition, relates the convergence of shapes and that of the state solutions *in the appropriate topologies*. An immediate consequence is given in Theorem 2.2 which gives a sufficient compactness condition for $\mathcal{A}_R$ such that $\mathbb{P}_R$ has a solution. As we shall see, all the restriction formulations discussed in this thesis for the material distribution problem satisfy this condition, a fact that may not be evident by glancing at the seemingly dissimilar proofs in the literature. In some instances (see section 3.2 and comments preceding Proposition 3.1), it allows us to improve the existing results.

Proposition 2.1. *Consider* $\chi_n, \chi \in L^\infty(\Omega; [0,1])$ *such that* $\chi_n \to \chi$ *strongly in* $L^1(\Omega)$. *Then* $\mathbf{u}_{\chi_n} \to \mathbf{u}_\chi$ *strongly in* $H^1(\Omega)^d$.

Proof. By (2.9), the sequence $\mathbf{u}_{\chi_n}$ is bounded and so there is a subsequence, again denoted by $\mathbf{u}_{\chi_n}$, that converges weakly to some $\mathbf{u} \in \mathcal{U}$. Since Ω is bounded, we shall assume that by going to another subsequence $\chi_n \to \chi$ a.e. in Ω. We will show that $\mathbf{u}$ is the solution to

the state equation corresponding to χ, i.e., $\mathbf{u} = \mathbf{u}_\chi$. For any $\mathbf{v} \in \mathcal{V}$,

$$\begin{aligned} |a(\mathbf{u}_{\chi_n}, \mathbf{v}; \chi_n) - a(\mathbf{u}, \mathbf{v}; \chi)| &\leq |a(\mathbf{u}_{\chi_n}, \mathbf{v}; \chi_n) - a(\mathbf{u}_{\chi_n}, \mathbf{v}; \chi)| \\ &\quad + |a(\mathbf{u}_{\chi_n}, \mathbf{v}; \chi) - a(\mathbf{u}, \mathbf{v}; \chi)| \qquad (2.37) \\ &\leq \left| \int_\Omega (\mathbf{C}_{\chi_n} - \mathbf{C}_\chi)\, \boldsymbol{\epsilon}(\mathbf{u}_{\chi_n}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \right| \\ &\quad + |a(\mathbf{u}_{\chi_n} - \mathbf{u}, \mathbf{v}; \chi)| \qquad (2.38) \end{aligned}$$

The second term vanishes since $\nabla \mathbf{u}_{\chi_n} \rightharpoonup \nabla \mathbf{u}$ weakly in $L^2(\Omega)^{d\times d}$. Regarding the first term, we appeal to Egoroff's theorem [134]: given $\delta > 0$, there exists a measurable subset $\Omega_\delta \subseteq \Omega$ such that $|\Omega \backslash \Omega_\delta| \leq \delta$ and $\chi_n \to \chi$ uniformly on Ω_δ. Moreover, $\|\nabla \mathbf{u}_{\chi_n}\|_{0,2,\Omega}$ is bounded, so for some constant k,

$$\begin{aligned} \left| \int_\Omega (\mathbf{C}_{\chi_n} - \mathbf{C}_\chi)\, \boldsymbol{\epsilon}(\mathbf{u}_{\chi_n}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \right| &= \left| \int_{\Omega_\delta} (\chi_n - \chi) \left(\mathbf{C}^+ - \mathbf{C}^-\right) \boldsymbol{\epsilon}(\mathbf{u}_{\chi_n}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \right| + k\delta \\ &\leq k \left(\|\chi_n - \chi\|_{0,\infty,\Omega} + \delta \right) \qquad (2.39) \end{aligned}$$

Since δ is arbitrary, this quantity vanishes as $n \to \infty$. Hence $\ell(\mathbf{v}) = a(\mathbf{u}_{\chi_n}, \mathbf{v}; \chi_n) \to a(\mathbf{u}, \mathbf{v}; \chi)$ for each $\mathbf{v} \in \mathcal{V}$, which implies that $\mathbf{u}$ solves the state equation corresponding to χ[3]. Since $\mathbf{u}_\chi$ is unique, it follows that every convergent subsequence of the entire sequence $\mathbf{u}_{\chi_n}$ converges to $\mathbf{u} = \mathbf{u}_\chi$. Therefore, the convergence holds for the entire sequence.

We next show that the convergence of state solutions is in fact strong. Since $\mathbf{u}_{\chi_n} - \mathbf{u}_\chi \in \mathcal{V}$, we have:

$$\begin{aligned} c \|\mathbf{u}_{\chi_n} - \mathbf{u}_\chi\|_{1,2,\Omega}^2 &\leq a(\mathbf{u}_{\chi_n} - \mathbf{u}_\chi, \mathbf{u}_{\chi_n} - \mathbf{u}_\chi; \chi_n) \\ &= a(\mathbf{u}_{\chi_n}, \mathbf{u}_{\chi_n}; \chi_n) - a(\mathbf{u}_{\chi_n}, \mathbf{u}_\chi; \chi_n) - a(\mathbf{u}, \mathbf{u}_{\chi_n} - \mathbf{u}_\chi; \chi_n) \qquad (2.40) \\ &= \ell(\mathbf{u}_{\chi_n}) - \ell(\mathbf{u}_\chi) - a(\mathbf{u}_\chi, \mathbf{u}_{\chi_n} - \mathbf{u}_\chi; \chi_n) \end{aligned}$$

As $\mathbf{u}_{\chi_n} \rightharpoonup \mathbf{u}_\chi$ weakly, $\ell(\mathbf{u}_{\chi_n}) \to \ell(\mathbf{u}_\chi)$ and $a(\mathbf{u}_\chi, \mathbf{u}_{\chi_n} - \mathbf{u}_\chi; \chi_n) \to 0$. Hence the right-hand side of (2.40) vanishes and $\mathbf{u}_{\chi_n} \to \mathbf{u}_\chi$ in norm. □

We note that a weaker form of the above proposition, establishing only weak convergence of the state solutions, follows from homogenization theory since the convergence $\chi_n \to \chi$ in $L^1(\Omega)$ implies that $\mathbf{C}_{\chi_n}$ G-converges to $\mathbf{C}_\chi$ (see Lemma 1.2.22 of [3] for the analogous result in conductivity). The following theorem gives a sufficient condition for existence of solutions.

[3] An alternative way to show that the first term in (2.38) vanishes is to note that $\mathbf{C}_{\chi_n} - \mathbf{C}_\chi$ converges a.e. to the zero tensor and $(\mathbf{C}_{\chi_n} - \mathbf{C}_\chi)\, \boldsymbol{\epsilon}(\mathbf{v})$ converges in $L^2(\Omega)^{d\times d}$ to the zero tensor. Since $\boldsymbol{\epsilon}(\mathbf{u}_{\chi_n})$ is bounded, it follows that the term vanishes as $n \to \infty$.

Theorem 2.2. *Suppose $\mathcal{A}_R$ is compact in the strong topology of $L^1(\Omega)$[4]. Then there exists a solution to $\mathbb{P}_R$.*

Proof. By compactness of $\mathcal{A}_R$, any *minimizing* sequence $\chi_n \in \mathcal{A}_R$ admits a subsequence that converges to some $\chi \in \mathcal{A}_R$ in $L^1(\Omega)$. By the previous proposition, the associated solutions to the state equation also converge strongly to the solution $\mathbf{u}_\chi$. From the continuity of the objective function on $L^1(\Omega) \times H^1(\Omega)^d$, we conclude that $J(\chi_n, \mathbf{u}_{\chi_n}) \to J(\chi, \mathbf{u}_\chi)$. □

Perhaps the most well-known example of a restriction approach is the perimeter constraint formulation by Ambrosio and Buttazzo [11] and implemented by, among others, Haber and co-workers [88]. The restricted space of admissible domains consists of those domains with bounded perimeter. More specifically,

$$\mathcal{A}_R = \left\{\chi \in BV(\Omega; \{0,1\}) : \textstyle\int_\Omega |\nabla \chi| \, \mathrm{d}\mathbf{x} \leq \overline{P}\right\} \tag{2.41}$$

where $\overline{P}$ is the specified upper bound for the total variation of the characteristic functions. Observe that the total variation functional has the following nontrivial definition

$$\int_\Omega |\nabla \chi| \, \mathrm{d}\mathbf{x} := \sup \left\{ \int_\Omega \chi \mathrm{div} \boldsymbol{\psi} \mathrm{d}\mathbf{x} : \boldsymbol{\psi} \in C_0^1(\Omega)^d, \|\boldsymbol{\psi}\|_{0,\infty,\Omega} \leq 1 \right\} \tag{2.42}$$

which for sufficiently regular characteristic function χ measures the total length (area) of the interface between the set $\{\chi = 0\}$ and $\{\chi = 1\}$. The compactness of this space follows from the compact embedding of $BV(\Omega)$ into $L^1(\Omega)$ (cf. [75]).

Returning to the counterexample of section 2.3, we can see the total perimeter between the two phases in the minimizing sequence (2.33) grows without bound (see Figure 2.4). Given the finite upper $\overline{P}$ on the perimeter, only the initial segment of sequence belongs to the space (2.41). Therefore, the perimeter constraint restricts the rapid oscillations in the phase boundary.

Under certain growth assumption on the objective function, one can show that the optimal solution to the perimeter constraint problem in fact forms an open set [11]. Also, for two-dimensional compliance minimization problem, Chambolle and Larsen [44] proved that the optimal solution has C^∞ boundary.

Within the restriction framework outlined here, existence of solutions for a larger class of problems can be established. For example, inclusion of stress constraints will not pose any difficulties since the strong convergence of the state solutions guarantees that, in the limit, the pointwise stress criteria is satisfied. More specifically, we can consider the admissible space

$$\mathcal{A}_R^{\overline{\sigma}} = \{\chi \in \mathcal{A}_R : \sigma_{\mathsf{e}}(\chi, \mathbf{u}_\chi) \leq \overline{\sigma} \text{ a.e. in } \Omega\} \tag{2.43}$$

[4] Again note that, since Ω has finite measure and all the characteristic functions are bounded, by Vitali's theorem, convergence in $L^p(\Omega)$ implies convergence in $L^q(\Omega), 1 \leq q < \infty$, i.e., they all induce the same topology.

Figure 2.4: Three terms in the minimizing sequence (2.33). The total variation $\int_\Omega |\nabla \chi| \, \mathrm{d}\mathbf{x}$ for these arrangements from left to right is $4L$, $6L$, and $10L$, respectively

where $\overline{\sigma} > 0$ is the given maximum allowable effective stress. If sequence $\chi_n \in \mathcal{A}_R^{\overline{\sigma}}$ converges to χ, then by Proposition 2.1, the associated stresses also converge strongly, that is, $\boldsymbol{\sigma}_{\chi_n} \to \boldsymbol{\sigma}_\chi$ strongly in $L^2(\Omega)^{d\times d}$. Thus, up to a subsequence, we have $\sigma_{\mathtt{e}}(\chi_n, \mathbf{u}_{\chi_n}) \to \sigma_{\mathtt{e}}(\chi, \mathbf{u}_\chi)$ almost everywhere. To obtain the same result, we can also appeal to a more general result by Lipton [108] on the stress-constrained $G-$closure, since by Proposition 2.1, strong convergence of χ_n to χ implies $G-$convergence of the elasticity tensors $\mathbf{C}_{\chi_n}$ to $\mathbf{C}_\chi$.

2.6 Restriction of space of density functions

In the previous section, we remained within the space of characteristic functions by requiring $\mathcal{A}_R \subseteq \mathcal{A}_C$ which may suggest that the above compactness condition excludes many restriction-type methods that have been proposed in the context of density methods. Examples include the filtering method [31], the slope constraint formulation [128], regularized intermediate volume approach [28, 29] as well as the Tikhonov and total variation regularization formulations presented later in this thesis.

However, the main ingredient in the proof of existence for these formulation is Proposition 2.1 as it relates the convergence of design field to the convergence of the displacement field. Note that the proposition is stated and proved in the larger space $L^\infty(\Omega; [0,1])$ of density functions and is therefore applicable in the larger context. For example, consider the density formulation based on the SIMP model. Provided that the space of admissible densities is compact in $L^1(\Omega)$, given a minimizing sequence ρ_n, one can extract a subsequence, again denoted by ρ_n, such that $\rho_n \to \rho$ in $L^1(\Omega)$ for some admissible density function ρ. It follows that $\rho_n^p \to \rho^p$ as well (since the density functions are bounded) and by Proposition 2.1, the associated solutions to the state equation also converge. Given the continuity requirements on the cost functional J, we again conclude that compactness is $L^1(\Omega)$ is a sufficient condition for existence of solutions.

All the above-mentioned density methods (see also section 1.3) indeed satisfy this condition. For example, consider the filtering method in which the space of admissible densities

is defined as

$$\mathcal{A}_R = \{\rho : \rho = \mathcal{P}_F(\eta) \text{ for some } \eta \in L^\infty(\Omega; [0,1])\} \tag{2.44}$$

where F is the filtering kernel. The next proposition shows that this space is compact in $L^1(\Omega)$ under some mild assumptions on the kernel F (see also Theorem 3.3 of [29]). It is easy to see that the linear hat kernel defined by (1.16) satisfies these assumptions.

Proposition 2.3. *Let $F : \Omega \times \Omega \to \mathbb{R}$ be a measurable and non-negative function. Moreover, assume that for some constant $c > 0$,*

$$\|F(\mathbf{x}, \cdot)\|_{0,1,\Omega} \leq c \quad \textit{for a.e. } \mathbf{x} \in \Omega \tag{2.45}$$

Then the space $\mathcal{A}_R$ defined by (2.44) is compact in $L^1(\Omega)$.

Proof. Given a sequence $\rho_n \in \mathcal{A}_R$, by definition, there exists a sequence $\eta_n \in L^\infty(\Omega; [0,1])$ such that $\rho_n = \mathcal{P}_F(\eta_n)$. By boundedness of η_n, we have $\eta \in L^\infty(\Omega)$ such that $\eta_n \rightharpoonup \eta$ in weak* topology of $L^\infty(\Omega)$. We next show that $\eta \geq 0$ almost everywhere. Suppose to the contrary $\eta < 0$ over a measurable subset $B \subseteq \Omega$ with $|B| > 0$. Let $\varepsilon = -\int_B \eta \mathrm{d}\mathbf{x}$. Since $\chi_B \in L^1(\Omega)$ and $\varepsilon > 0$, for sufficiently large n, we have $\int_\Omega \chi_B (\eta_n - \eta) \, \mathrm{d}\mathbf{x} < \varepsilon/2$ and so

$$-\int_B \eta \mathrm{d}\mathbf{x} = \int_\Omega \chi_B (\eta_n - \eta) \, \mathrm{d}\mathbf{x} - \int_\Omega \chi_B \eta_n \mathrm{d}\mathbf{x} < \varepsilon/2 \tag{2.46}$$

which is a contradiction. A similar proof show that $\eta \leq 1$ a.e. and so $\eta \in L^\infty(\Omega; [0,1])$.

The fact that $F(\mathbf{x}, \cdot) \in L^1(\Omega)$ implies that for almost every $\mathbf{x} \in \Omega$,

$$\rho_n(\mathbf{x}) = \int_\Omega F(\mathbf{x}, \mathbf{y}) \eta_n(\mathbf{y}) \mathrm{d}\mathbf{x} \to \int_\Omega F(\mathbf{x}, \mathbf{y}) \eta(\mathbf{y}) \mathrm{d}\mathbf{x} \equiv \rho(\mathbf{x}) \tag{2.47}$$

Thus ρ_n converges almost everywhere to $\rho = \mathcal{P}_F(\eta)$. Moreover, ρ_n is uniformly bounded since

$$|\rho_n(\mathbf{x})| \leq \|F(\mathbf{x}, \cdot)\|_{0,1,\Omega} \, \|\eta_n\|_{0,\infty,\Omega} = c \tag{2.48}$$

Since Ω has finite measure, by Lebesgue dominated convergence, $\rho_n \to \rho$ in $L^1(\Omega)$. This shows that space (2.44) is compact in $L^1(\Omega)$. □

The intuition behind the filtering formulation is that while the auxiliary field η may be rough (there are no restrictions placed on its regularity), the corresponding density function is necessarily smooth by virtue of its construction via the convolution operation. Note that in the above proof, the sequence η_n was only convergent in the weak (average) sense. However, passing this weakly convergence sequence through the filtering operation produces a strongly convergent sequence of density functions[5]. The compactness condition

[5]This property of the filtering map is also emphasized by Borrvall and Petersson [29] and is at the heart of their

requires convergence in the strong topology to ensure that the corresponding solutions to the state equation also converge. Similar to filtering, the regularity conditions such as bound on total variation or peak gradient (cf. (1.11) and (1.13)) can be viewed as "compactifying" conditions since they guarantee that the space is compact in the strong topology.

2.7 Restriction of space of implicit functions

We next focus our attention to the requirements for a well-posed implicit function formulation for the two-phase topology optimization problem. In particular, we discuss an appropriate set of conditions that need to be imposed on the implicit functions parameterizing the shapes to ensure existence of solutions.

The following formulation is due to Liu et al. [109] (also pursued in [158]) and its particular relevance to the implicit function formulation is that it is essentially obtained by imposing additional constraints on $\mathcal{F}_C$. In particular, the classical space of characteristic functions $\mathcal{A}_C$ is replaced by

$$\mathcal{A}_R = H(\mathcal{F}_R) \tag{2.49}$$

where H is the Heaviside map and the implicit functions $\varphi \in \mathcal{F}_R \subseteq W^{1+\theta,p} \cap \mathcal{F}_C$ satisfy

$$\text{(R1)}: \quad \|\varphi\|_{1+\theta,p,\Omega} \leq K \tag{2.50}$$

$$\text{(R2)}: \quad |\varphi(\mathbf{x})| + |\nabla\varphi(\mathbf{x})| \geq \nu \quad \text{a.e. in } \Omega \tag{2.51}$$

for some positive constants θ, K and ν[6].

Before giving the proof for compactness of $\mathcal{A}_R$, it is instructive to discuss the significance of each constraint in relation to the counterexample described in the section 2.3. (R1) excludes the possibility of rapid oscillations of a minimizing sequence φ_n. Note that for the sequence defined by (2.33) for the counterexample,

$$\|\varphi\|_{1+\theta,p,\Omega} \to \infty \tag{2.52}$$

The condition (R2) ensures that the set $\{\varphi = 0\}$ has zero measure. This set corresponds to the "boundary" of the domain defined by $H(\varphi)$ and is precisely where the Heaviside function is discontinuous. To show this, we appeal to a classical result (see [75, 105]), which states that for every $\varphi \in W^{1,p}_{\text{loc}}(\Omega), 1 \leq p \leq \infty$, we have $\chi_{\{\varphi=0\}}\nabla\varphi = \mathbf{0}$ a.e. in Ω. If φ satisfies (R2), then $|\nabla\varphi| \geq \nu$ a.e. on $\{\varphi = 0\}$ and so $\chi_{\{\varphi=0\}} = 0$ a.e. on Ω. Without (R2), the

proposed regularized intermediate density method. However, their existence proof in [29], as stated, is limited to the compliance minimization problem.

[6]The first condition, (R1), can be weakened to allow certain local oscillations of associated domains. In particular, we only need to require $\varphi \in W^{1,p}_{\text{loc}}(\Omega)$ to satisfy $\|\varphi\|_{1+\theta_\Sigma,p,\Sigma} \leq K_\Sigma$ for all $\Sigma \subset\subset \Omega$ and $\theta_\Sigma > 0$ (cf. [109]).

sequence,

$$\varphi_n(\mathbf{x}) = \frac{\alpha}{n^{2+\theta}} \sin(nx_1) \tag{2.53}$$

is a non-convergent minimizing sequence for the counterexample even though it satisfies (R1). Observe that $H(\varphi_n)$ produces the same laminated designs as (2.33) even though the gradient of φ_n in this case is bounded. In some sense, the Heaviside magnifies the small amplitude oscillations in φ_n around zero. In [105], the "transversality" condition[7] (R2) is examined in relation to the issue of level set "fattening," which concerns the propagation of boundary $\{\varphi = 0\}$ by Hamilton-Jacobi equations (for example, in mean curvature type flows). Since the level set methods are based on the motion of the level set front, these relatively weak regularity conditions clearly delineate the conditions that the level set function must meet. These conditions *together* introduce a length scale into a problem that otherwise has no characteristic length.

Proposition 2.4. *The space of admissible designs $\mathcal{A}_R$ defined by (2.49) is compact in strong topology of $L^1(\Omega)$.*

Proof. Let $\chi_n = H(\varphi_n)$ be a sequence in $\mathcal{A}_R$. By Sobolev embedding theorems [1], there exists $\varphi^* \in W^{1,p}(\Omega)$ such that, up to a subsequence, $\varphi_n \to \varphi^*$ strongly in $W^{1,p}(\Omega)$. This in turn implies that, for possibly another subsequence, $\varphi_n \to \varphi^*$ and $\nabla\varphi_n \to \nabla\varphi^*$ a.e. in Ω. It follows that $\varphi^* \in \mathcal{F}_R$. It remains to show that $\chi_n \to \chi^* := H(\varphi^*)$. If $\varphi^*(\mathbf{x}) > 0$, then there exists $n_{\mathbf{x}}$ such that $\varphi_n(\mathbf{x}) > 0$ for all $n \geq n_{\mathbf{x}}$. Thus $\chi^*(\mathbf{x}) = \chi_n(\mathbf{x}) = 1$ for all $n \geq n_{\mathbf{x}}$. A similar argument can be made for the case $\varphi^*(\mathbf{x}) < 0$. Since $\varphi^*(\mathbf{x}) = 0$ only over a set of measure zero, it follows that $\chi_n \to \chi^*$ a.e. and by Lebesgue dominated convergence theorem in $L^1(\Omega)$ norm. □

2.8 Approximation of the Heaviside map

Most implicit function or level set formulations for the topology optimization in the literature replace the Heaviside map in the numerical calculation by a smeared approximation of it. This is done with little attention to the ill-posedness of the continuum problem which, as we shall now demonstrate, completely transforms the problem. It is precisely the transversality condition (R2) that guarantees that this approximation is valid.

The typical approximation H_γ of Heaviside map is of the form

$$H_\gamma(\varphi)(\mathbf{x}) = \begin{cases} 0, & \varphi(\mathbf{x}) < -\gamma \\ \zeta\left(\gamma^{-1}\varphi(\mathbf{x})\right), & -\gamma \leq \varphi(\mathbf{x}) \leq \gamma \\ 1, & \varphi(\mathbf{x}) > \gamma \end{cases} \tag{2.54}$$

[7] This designation comes from the fact that this condition requires the graph of φ to cut the zero hyperplane "transversally"

where $0 < \gamma \ll \alpha$ is the smearing width and ζ is a smooth function that satisfies $\zeta(-1) = 0$ and $\zeta(1) = 1$. An example of such function is $\zeta(y) = \frac{1}{2}\left[1 + \sin\left(\frac{\pi y}{2}\right)\right]$.

When $\mathcal{A}_C$ is replaced by $H_\gamma(\mathcal{F}_C)$, *regardless of the width of the approximate Heaviside*, the classical optimal design problem is transformed into the so-called variable thickness problem. The variable thickness problem is defined by $\mathbb{P}_C$ when $L^\infty(\Omega; [0,1])$ is used in place of $\mathcal{A}_C$. In the two-dimensional setting and when $\mathbf{C}^- \propto \mathbf{C}^+$, each $\rho \in L^\infty(\Omega; [0,1])$ has the interpretation of the thickness of a plate occupying Ω. Indeed, for any $\rho \in L^\infty(\Omega; [0,1])$, there exists $\varphi \in L^\infty(\Omega; [-\alpha, \alpha])$ such that $\rho = H_\gamma(\varphi)$. Conversely, $H_\gamma(\varphi)$ is a "thickness" function for any implicit function φ.

It is straightforward to show the existence of solutions for this problem (see, for example [126]). Also the optimal solutions of the variable thickness problem generally take on intermediate values (i.e., values not equal to 0 or 1) over large areas of Ω. Therefore, regardless of how small γ is, the solutions to the problem with approximate Heaviside will consist of large areas with an intermediate phase. In a sense, this is a manifestation of the ill-posedness of $\mathbb{P}_C$.

In the present restriction framework, the transversality condition (R2) prevents the implicit functions from remaining in the smeared range of the Heaviside far away from the boundary. Without such a condition imposed, it is expected that the optimization would favor implicit functions that are flat in order to achieve composite designs. We also note that the first order condition of optimality for the compliance design problem with approximate Heaviside states that for each point in Ω,

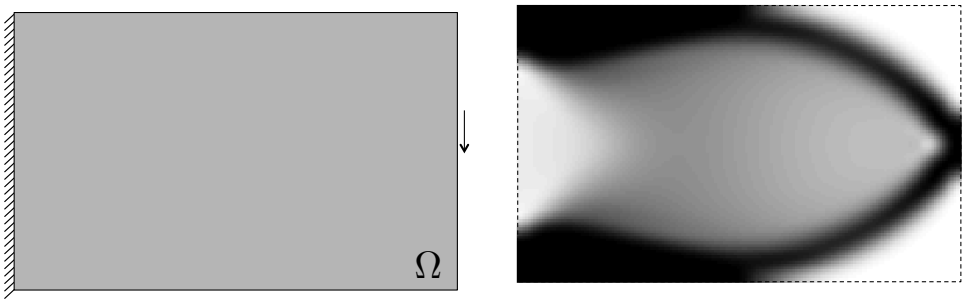
$$\delta_\gamma(\varphi)\left[\left(\mathbf{C}^+ - \mathbf{C}^-\right)\boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) : \boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) - \lambda\right] = 0 \tag{2.55}$$

where δ_γ is the smooth Dirac delta function, defined by

$$\delta_\gamma(\varphi)(\mathbf{x}) = \begin{cases} \gamma^{-1}\zeta'\left(\gamma^{-1}\varphi(\mathbf{x})\right), & -\gamma < \varphi(\mathbf{x}) < \gamma \\ 0, & \text{otherwise} \end{cases} \tag{2.56}$$

Without the transversality requirement, we can have $\delta_\gamma(\varphi) \neq 0$ over large parts of the domain, in which case, (2.55) would be identical to the necessary and sufficient condition of optimality for the variable thickness problem (cf. [126]).

We demonstrate this phenomenon numerically using the computational framework presented in the chapter 1 (with its implementation in the appendix). The sizing function in this case is φ, which in order to guarantee its smoothness, is defined using the filtering operator as $\varphi = \mathcal{P}_F(\eta)$. The auxiliary field η belongs to the space $L^\infty(\Omega; [-\alpha, \alpha])$. Observe that by choosing a sufficiently smooth filtering kernel F, we can ensure that φ satisfies the (R1) smoothness condition. For example, in the case of the linear hat kernel (cf. (1.16)),



(a) **(b)**

Figure 2.5: (a) The extended design domain (with height $h = 1$ and width $w = 1.6$) and the boundary conditions for the cantilever problem (b) The "variable thickness" solution to the cantilever problem

we can choose the radius R to be sufficiently large.

In order to observe the effects of Heaviside approximation, we choose the following material interpolation functions for the compliance problem

$$m_E(\varphi) = H_\gamma(\varphi) + \varepsilon \left[1 - H_\gamma(\varphi)\right], \qquad m_V(\varphi) = H_\gamma(\varphi) \tag{2.57}$$

We consider the design of a cantilever problem whose extended domain and boundary conditions shown in Figure 2.5(a). The initial guess is shown in Figure 2.6(a) and the following parameters are used in the calculations

$$\alpha = 0.1, \qquad R = 0.06, \qquad \gamma = 0.03 \tag{2.58}$$

The volume parameter λ is updated in each iteration so that $\int_\Omega m_V(\varphi)\mathrm{d}\mathbf{x}$ is fixed at $|\Omega|/2$ and a move limit of $m = 0.01$ is used (cf. section 1.5).

Despite starting from an implicit function that satisfies the transversality condition (and therefore the has two phases separated), the final arrangement contains large regions of intermediate values. Notice from Figure 2.6 that as the design evolves and implicit function becomes flat, the shape loses its definition. For comparison, Figure 2.5(b) shows the solution to variable thickness problem for the same cantilever configuration starting from a uniform initial thickness of $1/2$[8].

With (R1) and (R2) imposed on the implicit functions, we can show that replacing H by H_γ is justified in the sense that the optimal solutions to the approximate problem converge to a minimizer of J over $\mathcal{A}_R$ as $\gamma \to 0$. Let us define the space $\mathcal{A}_R^\gamma = H_\gamma(\mathcal{F}_R)$ for each $\gamma > 0$. Observe that $\mathcal{A}_R^\gamma \nsubseteq \mathcal{A}_R$ since the smeared Heaviside can produce design fields with intermediate values in $(0, 1)$. It is straightforward to show that $\mathcal{A}_R^\gamma$ is compact

[8]This was computed using SIMP model with $p = 1$ which in fact produces a convex problem

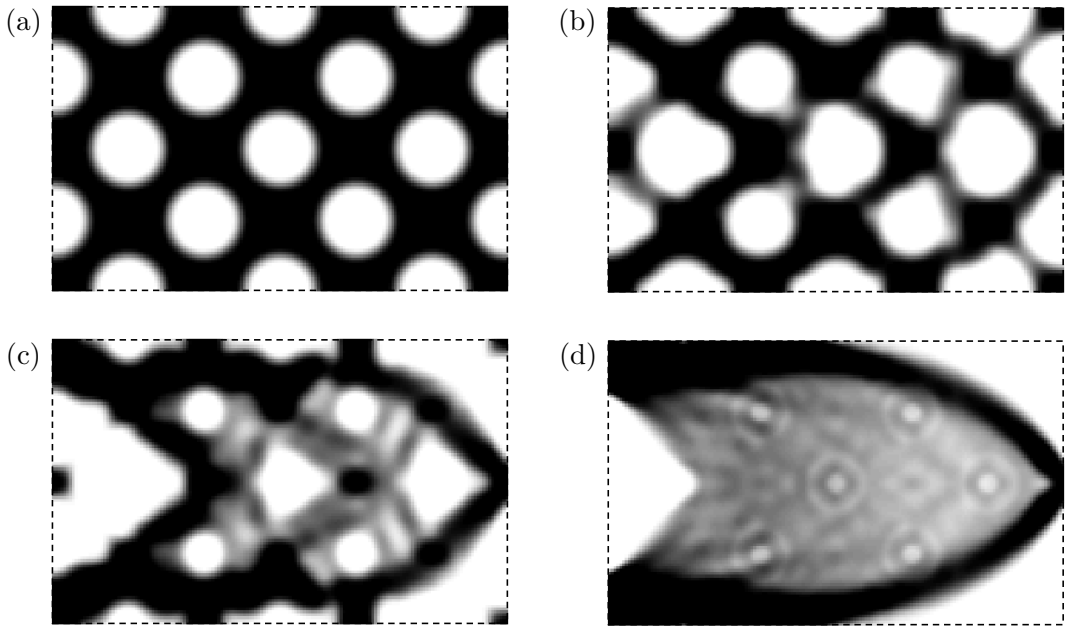


Figure 2.6: The evolution of the design field $\chi = H_\gamma(\varphi)$ associated with interpolation functions (2.57) plotted in greyscale (a) initial guess (b) iteration 10 (c) iteration 20 (d) iteration 150. Since the value of φ in the grey regions is in the range $(-\gamma, \gamma)$, the implicit function field is becoming flat near zero as the design evolves

in $L^1(\Omega)$ and so $J(\chi, \mathbf{u}_\chi)$ has a minimizer χ^*_γ is $\mathcal{A}^\gamma_R$. We next prove that the sequence χ^*_γ has a subsequence that converges to a minimizer χ^* of J in $\mathcal{A}_R$.

Proposition 2.5. *For each $0 < \gamma \leq \alpha$, let χ^*_γ* denote a minimizer of J in the space $\mathcal{A}^\gamma_R = H_\gamma(\mathcal{F}_R)$. There exists a minimizer χ^* of J in $H(\mathcal{F}_R)$ such that, up to a subsequence, $\chi^*_\gamma \to \chi^*$ *strongly in $L^1(\Omega)$ as $\gamma \to 0$.*

Proof. Let $\varphi_\gamma \in \mathcal{F}_R$ such that $\chi^*_\gamma = H_\gamma(\varphi^*_\gamma)$. As in the proof of Proposition 2.4, we can show that there exists $\varphi^* \in \mathcal{F}_R$ such that $\varphi^*_\gamma \to \varphi^*$ a.e. in Ω. Define $\chi^* = H(\varphi^*)$. As before $\{\varphi^* = 0\}$ is a set of measure zero. Suppose $\varphi^*(\mathbf{x}) > 0$ for some $\mathbf{x} \in \Omega$. Then, there exists $\gamma_\mathbf{x}$ such that for all $\gamma < \gamma_\mathbf{x}$,

$$\chi^*_\gamma(\mathbf{x}) = H_\gamma(\varphi^*_\gamma)(\mathbf{x}) = 1 = H(\varphi^*)(\mathbf{x}) = \chi^*(\mathbf{x}) \tag{2.59}$$

A similar argument can be made for the case $\varphi^*(\mathbf{x}) < 0$ and so $\chi^*_\gamma \to \chi^*$ pointwise and subsequently, by Lebesgue dominated convergence theorem, in $L^1(\Omega)$ norm. It remains to show that χ^* is a minimizer of J in $\mathcal{A}_R$. To this end, let $\varphi \in \mathcal{F}_R$ and define $\chi = H(\varphi)$ and $\chi_\gamma = H_\gamma(\varphi)$. Observe that $\chi_\gamma \to \chi$ strongly in $L^1(\Omega)$ as $\gamma \to 0$. By optimality of φ^*_γ in $\mathcal{A}^\gamma_R$, we have

$$J(\chi^*_\gamma, \mathbf{u}_{\chi^*_\gamma}) \leq J(\chi_\gamma, \mathbf{u}_{\chi_\gamma}) \tag{2.60}$$

Passing to the limit $\gamma \to 0$, it follows from the continuity of J that

$$J(\chi^*, \mathbf{u}_{\chi^*}) \leq J(\chi, \mathbf{u}_\chi) \tag{2.61}$$

Hence χ^* is a minimizer of J over $\mathcal{A}_R$. □

2.9 Comments on some existing algorithms

As seen from the discussion in the previous section, the transversality condition is necessary in order to introduce a limiting length scale into an otherwise ill-posedness problem. Many level set and implicit function methods in the literature do not appropriately address the ill-posedness issue and therefore it is not surprising the resulting numerical algorithms often involve various heuristic measures which are sensitive to the choice of parameters. In this section will review and examine a handful of existing level set and implicit functions methods in light of the findings of the previous two sections, namely the need for two distinct regularity conditions. This is accompanied by some numerical investigations and illustrations.

One manifestation of the absence of a length-scale in level set methods based on the Hamilton-Jacobi evolution equation is that the complexity of the optimal solutions depend on the complexity of the initial guess used, which together with the mesh size[9] are the only length parameters that appear in the discrete problem. This phenomena is often attributed to the inability of the evolution equation to generate holes and so one approach to remedy it is to use topological derivatives to judiciously introduce holes in the course of the algorithm. However, such an approach is difficult to justify from a theoretical perspective since any complexity limit needed to guarantee existence of solutions is violated by the introduction of "infinitesimal" holes. From a practical perspective, such techniques are not very robust and very sensitive to the various parameter (e.g., how often a topological sensitivity analysis is carried out and holes are introduced), a fact that be easily demonstrated using the educational code by Challis [42].

Modification of sensitivities

With regards to the transversality condition, one mechanism in Hamilton-Jacobi-based level set methods that directly affects the gradient of implicit functions, especially at the boundary, is the so-called reinitialization step. The reason for reinitialization is that implicit function can become too steep near boundary or too flat in the interior regions in the course of the design evolution [8] which in turn can lead to accumulation of numerical errors in

[9] Incidentally, despite the fact that the issue of mesh-dependency is well-known in the topology optimization community in the context of density methods, it is rarely mentioned or examined in the level set literature

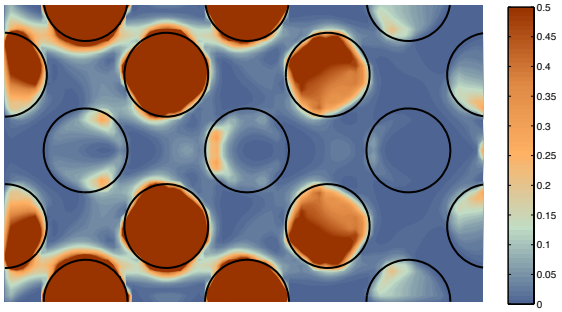


Figure 2.7: Plot of strain energy field $(\mathbf{C}^+ - \mathbf{C}^-)\,\boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) : \boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right)$ for φ corresponding to the material distribution shown in Figure 2.6(a). Note that the legend terminates at 0.5 for better illustration of the strain energy distrobution inside the stiff phase $\mathbf{C}^+$

solving the Hamilton-Jacobi equation. The reinitialization process creates a new implicit function φ with the same zero level set as before (so that boundary of the domain does not change) but with unit slope, i.e., $|\nabla\varphi| = 1$ everywhere in the extended domain. In other words, a signed-distance function is computed to replace the implicit function to represent the current shape. Observe that the reinitialization step is directly related to the transversality condition since it affects the slope of the implicit function near the boundary.

Since the reinitialization routine is computationally costly, it is usually performed as infrequently as possible[10]. In a number of level set formulations in order to avoid the difficulties associated with reinitialization, the velocity (sensitivity) field that drive the evolution of the boundary is modified in an inconsistent manner. For the compliance minimization problem, Wang et al. [175] propose a formulation based on radial basis functions[11] to eliminate the need for reinitialization and allow for the generation of holes. However, without much explanation, they also set the strain tensor $\boldsymbol{\epsilon}(\mathbf{u})$ in the $\mathbf{C}^-$ phase to zero when computing the velocity field. A similar approach is adopted in [178] where, interestingly, "too much" reinitialization is reported to adversely affect the evolution of the design. In this work, the authors use a geometrically exact reinitialization in *every* iteration in order to maintain the signed distance property of the implicit function at all times. In this manner, the Hamilton-Jacobi equation reduces to the basic gradient descent update, which for the compliance problem reads as,

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \left(\mathbf{C}^+ - \mathbf{C}^-\right)\boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) : \boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) - \lambda \tag{2.62}$$

However, they report "numerical instabilities" appearing in the form of oscillations in the

[10] Notice that determining the frequency of the reinitialization steps is not a trivial task itself and is often based on trial and error.

[11] The radial basis functions have also been in level set formulation of fluid flow problem in [129, 100]. As discussed in chapter 5, the mixing of the phases is naturally excluded for certain fluids problems.

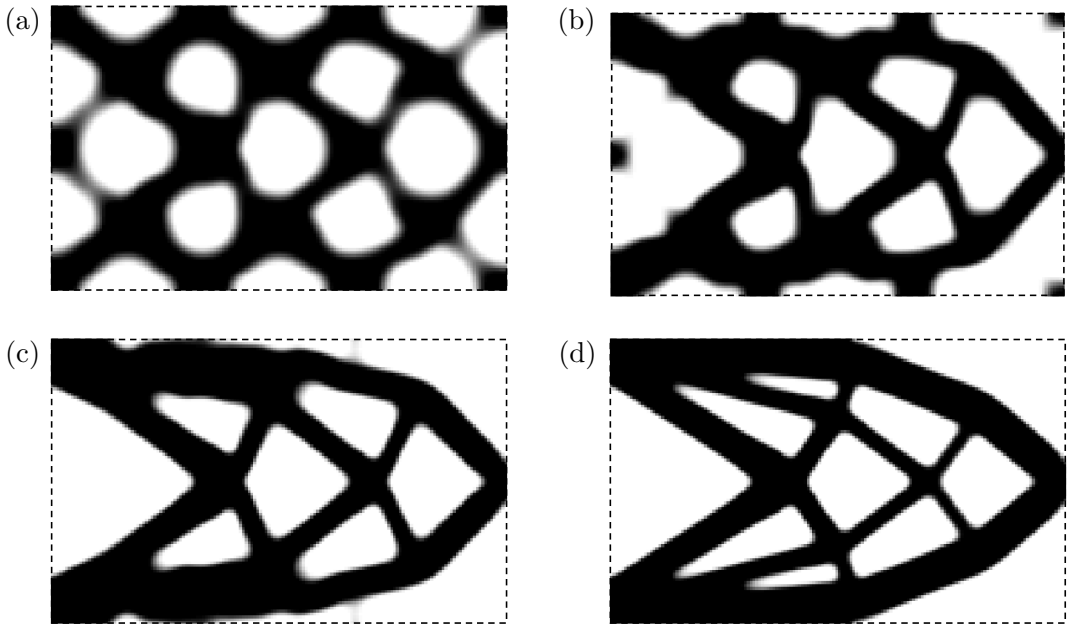


Figure 2.8: The evolution of the design field $\chi = H_\gamma(\varphi)$ using modified sensitivities (a) iteration 10 (b) iteration 20 (c) iteration 40 and (d) final design

boundary of final solutions. Again without much justification, they propose to remedy this problem by multiplying the first term in the sensitivity field by $H_\gamma(\varphi)$ and thus use the following evolution equation

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = H_\gamma(\varphi)\left(\mathbf{C}^+ - \mathbf{C}^-\right)\boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) : \boldsymbol{\epsilon}\left(\mathbf{u}_{H_\gamma(\varphi)}\right) - \lambda \tag{2.63}$$

A closer inspection of (2.62) reveals that as the deformation in the compliant phase is usually large (see Figure 2.7), the value of φ in the compliant phase is expected to increase in that phase while it would be lowered in the stiff phase, effectively flattening the level set function and producing a region of intermediate phases. By contrast, the $H_\gamma(\varphi)$ factor in (2.63) removes this "excessive" deformation in the $\mathbf{C}^-$ phase and prevents the mixing of the two phases.

As further evidence that such modifications of the sensitivity field may have a similar effect to the enforcement of transversality condition, we use the same formulation and algorithm that generated the result in Figure 2.6 but change the strain energy field as in (2.63). As shown in Figure 2.8, with this modification of sensitivities, the implicit function remains steep near the boundary and two phases are kept separate throughout the algorithm even though no transversality condition is imposed. Though this discussion can perhaps shed some light on the formulations in the above-mentioned references, the modification of sensitivities remains to be a heuristic and requires further justification and

Figure 2.9: Initial guess $H_\gamma(\mathcal{P}_F(\eta))$ plotted in greyscale and the associated auxiliary field η

examination before it can be used as the basis for an optimization algorithm. Notice that the present analysis is limited to the compliance problems and as usual the extension to the non-self-adjoint problems may not be straightforward.

It is worthwhile to mention a recent publication by van Dijk and co-workers [169] where the authors consider an explicit level set formulation based on consistent sensitivity information and the exact Heaviside map. In their formulation, the elements in the mesh that intersect the zero level set are assigned an intermediate stiffness and volume (the value is given by the normalized integral of $H(\varphi)$ over the element). Though at first glance it is expected that intermediate phase in this setting is only limited to one layer of elements around the boundary, they obtain final solutions that contain "grey" regions over several adjacent elements. In these regions, the level set function oscillates rapidly around zero so that the boundary $\{\varphi = 0\}$ intersects all the neighboring elements. The authors correctly conclude that "the presence of these gray areas with numerically superior performance demonstrates the consistency of the approach."

Use of multiple Heaviside approximations

There exists a handful of implicit function methods in the literature (not based on the Hamilton-Jacobi equation) that, without offering an explicit justification, use different approximate Heaviside functions for the compliance and volume terms. For example, Belytschko et al. [19] use two different smeared Heaviside functions for the compliance and volume terms for the standard compliance minimization problem. Their proposed Heaviside function for stiffness lies below the function used for the volume term. The stated reason for such choice is that the method is otherwise "unworkable." In another paper [177], Yamada et al. use a smooth interpolation function for the volume in place of an approximate Heaviside, which they claim allows for the appearance of intermediate regions and therefore contributes to "numerical stability" of the algorithm. Their argument is that "[e]limination of grayscales is important when using the equilibrium equations but is not important in the volume calculation." A similar approach is also effectively used in [87].

A simpler explanation in these cases is that the use of such distinct interpolation models

for volume and stiffness penalizes the appearance of intermediate phases in the same manner as the SIMP model in density formulations. Clearly, as demonstrated by the result in Figure 2.6, using the same smeared Heaviside function for both volume and stiffness leads to large regions of intermediate phases, contrary to the claim by Yamada and co-workers. The fact that both methods are capable of topological changes (e.g., introducing holes) during the optimization is also not surprising.

For this class of methods, penalization is the mechanism that enforces transversality. We explore this phenomenon numerically using the following interpolation functions:

$$m_E(\varphi) = [H_\gamma(\varphi)]^p + \varepsilon \left\{1 - [H_\gamma(\varphi)]^p\right\}, \qquad m_V(\varphi) = H_\gamma(\varphi) \tag{2.64}$$

where $p > 1$ is the penalization exponent. As before, the implicit function space is defined as $\{\mathcal{P}_F(\eta) : \eta \in L^\infty(\Omega; [-\alpha, \alpha])\}$ to ensure their smoothness and satisfaction of the (R1) condition. Notice that sensitivity calculations in this case are *consistent* with the above material interpolation functions. We present the numerical results for the MBB beam problem (cf. Figure 1.4) with the following parameters

$$\alpha = 1, \qquad p = 4, \qquad \gamma = \frac{0.4\alpha}{R} \tag{2.65}$$

Here R is the radius of the linear filtering kernel F.

This formulation leads to an optimization algorithm that evolves the design by variations only at the boundary since due to the presence of $\delta_\gamma(\varphi)$, the sensitivity field is zero away from the boundary provided that φ is sufficiently steep. We observed that the algorithm tends to stop at local minima when the auxiliary field η takes its extreme values throughout most of the domain. To continue the progress of the design, we reset η at such times in the algorithm to

$$\eta = \frac{\alpha}{3}\left(\chi_{\{\varphi>0\}} - \chi_{\{\varphi\leq 0\}}\right) \tag{2.66}$$

which preserves the topology of the current design. The algorithm is terminated when there is no change in the design after such resetting. Note that we are not advocating this as an efficient or well-designed algorithm and this "trick" is just used to get past certain local minima.

Figure 2.9 shows the initial guess that was used for the auxiliary field η and the corresponding design function $H_\gamma(\mathcal{P}_F(\eta))$. The final results for various values of the filtering radius R are shown in Figure 2.10. First observe the effects of the penalization. In all the optimal solutions, η takes extreme values of $-\alpha$ and α at the boundary similar to what is seen in density formulations. As a result, the implicit function field transitions between its extreme values near boundary over a distance of approximately $2R$ and so its slope at the

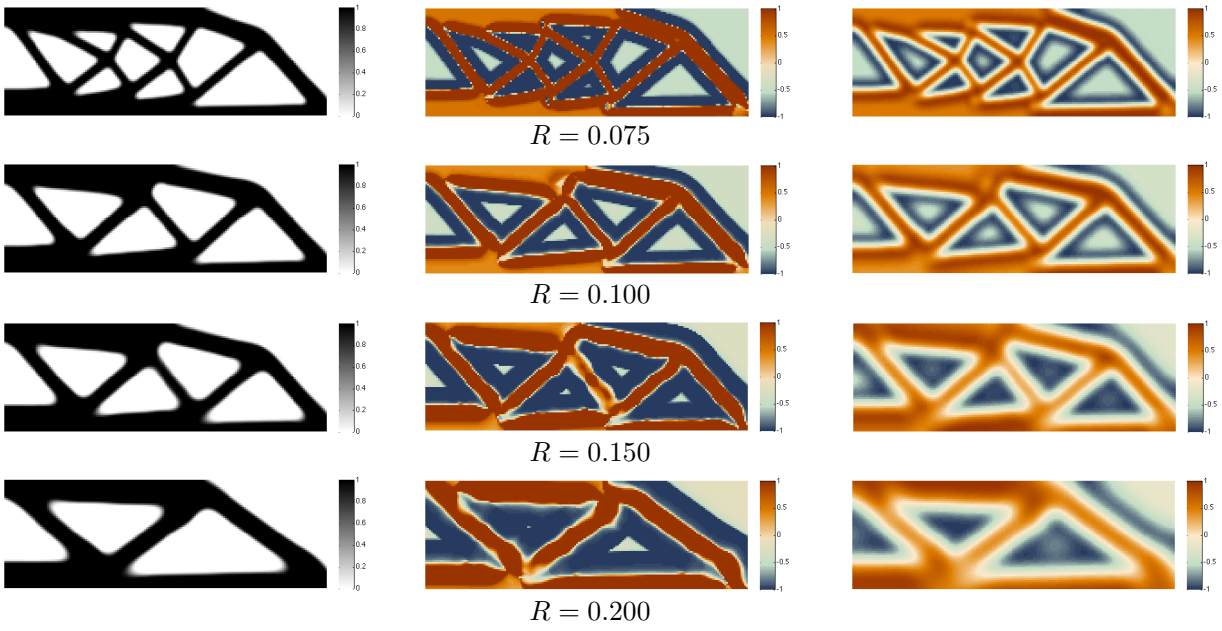


Figure 2.10: The final design using the material interpolation functions (2.64) for various filtering radii. The design field $H_\gamma(\varphi)$, the auxiliary function η, and the implicit function $\varphi = \mathcal{P}_F(\eta)$ are shown in the left, middle and right columns, respectively.

boundary is approximately α/R. These results indicate that, for the compliance problem, the transversality constant ν in (R2) can be related to the ratio α/R. Note also that the width of the Heaviside function γ was set to a fraction of this ratio (see eq. (2.65)).

With the transversality condition enforced, the complexity of the final solutions depends on the level of smooth imposed on the implicit functions, in this case determined by the filtering and its radius R. This can also been from the results presented in Figure 2.11 where different initial guesses were used for the same value of R. Despite the fact that the complexity of the initial is different, the final solutions, though different, have the same of degree of complexity as dictated by the filtering radius.

More direct approaches

In conclusion of this section, we mention and discuss some implicit function methods in the literature where the transversality condition is prescribed more explicitly in the formulation [55, 167, 166]. Note that these references do not directly discuss the existence issue and the use of additional "slope-penalty" terms is attributed to the desire for obtaining the "ideal" implicit function, namely a signed distance function, in the optimal regime.

To ensure smoothness of the implicit function field, these papers advocate the use of a Tikhonov regularizer of the form

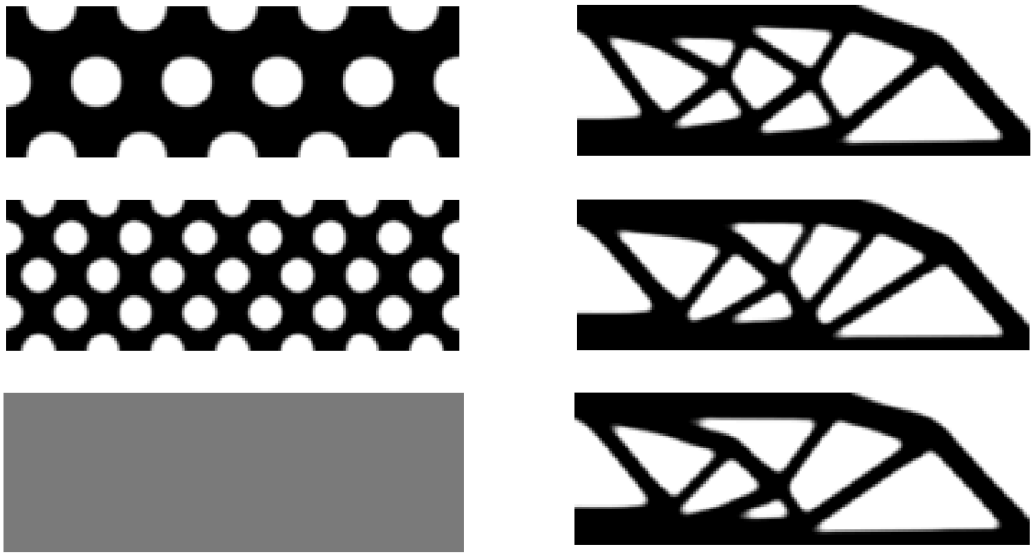


Figure 2.11: The initial configuration (left column) and final design (right column)

$$\beta_1 \int_\Omega |\nabla\varphi|^2 \, \mathrm{d}\mathbf{x} \tag{2.67}$$

which is added to the objective function. Of course, this term further encourage the flatness of the implicit function field unless other measures are taken. Moreover, in [167] and [166], an additional total variation regularization term

$$\beta_2 \int_\Omega |\nabla H_\gamma(\varphi)| \, \mathrm{d}\mathbf{x} \tag{2.68}$$

is included to penalize the total boundary of the shapes. For the class of inverse problem (cf. (2.13)) considered in these papers, the total variation regularization is often standard. However, noting that $|\nabla H_\gamma(\varphi)| = \delta_\gamma(\varphi)\,|\nabla\varphi|$, it is not clear why (2.68) would be needed along with the Tikhonov regularization term (2.67). Note that (2.68) only contains information about φ near the interface and does not control roughness of φ that may occur in the course of the optimization. This concentration further encourages the flatness of implicit functions near the boundary. Moreover, the presence of multiple regularization terms make the appropriate selection of prefactors β_1 and β_2 more complicated.

Regarding the transversality condition, Cunha [55] adds an additional term of the form

$$\hat{\beta} \int_\Omega \left(|\nabla\varphi|^2 - 1\right)^2 \mathrm{d}\mathbf{x} \tag{2.69}$$

to force the implicit function to approach a signed distance function with $|\nabla\varphi| \approx 1$. Note

that while the Tikhonov term (2.67) attempts to minimize the L^2-norm of $\nabla\varphi$, the new term (2.69) encourage $|\nabla\varphi|$ to remain close to 1 the throughout most of Ω. Therefore, these terms have opposing effects and the degree of influence of each term depends on the relative value of the coefficients β_1 and $\hat{\beta}$.

In a more recent work, van den Doel and Ascher [167] find this penalization strategy to be too stringent and propose a more forgiving global penalty term

$$\hat{\beta}\left[\int_\Omega \left(|\nabla\varphi|^2 - 1\right) \mathrm{d}\mathbf{x}\right]^2 \tag{2.70}$$

which penalizes flatness in φ in the average sense. Expanding this term, we can see it is equivalent to using the

$$\hat{\beta}\left[\int_\Omega |\nabla\varphi|^2 \,\mathrm{d}\mathbf{x}\right]^2 - 2\hat{\beta}\,|\Omega| \int_\Omega |\nabla\varphi|^2 \,\mathrm{d}\mathbf{x} \tag{2.71}$$

Notice that the second term directly opposes the effects of the Tikhonov regularizer. Just as in [55], the coefficients need to be carefully chosen and adapted in the course of the optimization in to obtain the right balance between these competing penalty terms.

Lastly, Terrel and Long [166] propose an alternative formulation that in fact closely resembles the (R2) condition in part due to the fact they approach the issue from a different perspective. Rather than obtaining a signed-distance function, their goal is to minimizer the measure of set where φ takes values in the smeared range $(-\gamma, \gamma)$ of the approximate Heaviside in order to ensure the discreteness of the design field $H_\gamma(\varphi)$. Recognizing that this quantity depends on both γ and $\nabla\varphi$, they add a pointwise inequality constraint of the form

$$\left(\frac{\varphi}{\gamma}\right)^2 + |\nabla\varphi|^2 \geq 1 - \delta \tag{2.72}$$

where δ is a small positive constant. It is evident that this constraint has the effect of regulating the slope of the implicit function near the boundary without affecting its variation in the interior. Though this constraint is the most direct enforcement of the transversality condition, there are several computational difficulties associated with imposing a pointwise constraint especially for large-scale problems defined on fine computational grids. In [166], the authors state that they impose this inequality constraint using a barrier method but provide very little information on the implementation details. Therefore, we did not include this formulation in our numerical investigations but we note that there is room for more work in this direction for topology optimization.

We numerically tested the first two formulations to assess their performance for the minimum compliance problem. In both cases, we were unable to find appropriate values for the coefficients of the two conflicting terms. When β_1 was chosen larger relative to $\hat{\beta}$, the

Tikhonov regularizer simply dominated the problem and as expected we obtained variable-thickness type solutions (similar to results in Figure 2.6). In the case where $\hat{\beta}$ was larger, the optimal solution consisted of rapid oscillations with small amplitude (ranging between γ and $-\gamma$) in large parts of the domain. In this way, the magnitude of the slope $|\nabla\varphi|$ remain close to one even though most of the domain, just as in the variable thickness, consisted of the intermediate phases. In relation to (R1) and (R2) conditions, the transversality condition was satisfied in this regime but the smoothness condition was violated. We observe both types of behavior even when starting from a near optimal solution (e.g., using φ shown in Figure 2.10 as the initial guess).

2.10 Single-phase problem and the Ersatz approximation

In this section, we examine the validity of the Ersatz approximation and the connection between the two-phase material distribution problem and the single-phase shape optimization in the limit of vanishing stiffness of the compliant phase.

Recall from chapter 1 that the unknown in the single-phase optimal shape problems is the domain on which the boundary value problem is defined. This domain is occupied by only one material, say the stiff phase $\mathbf{C} = \mathbf{C}^+$. As such the space of admissible shapes, denoted by $\mathcal{O}$, requires more regularity since measurability alone is not sufficient to ensure the well-posedness of the boundary value problem. For now, we shall assume the domains are open and contained in Ω, and that they are sufficiently smooth such that for each $\omega \in \mathcal{O}$, the governing state equation

$$\begin{aligned} \operatorname{div}\left[\mathbf{C}^+\boldsymbol{\epsilon}(\mathbf{u})\right] &= \mathbf{0} \quad \text{in } \omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\omega \cap \Gamma_D \\ \left[\mathbf{C}^+\boldsymbol{\epsilon}(\mathbf{u})\right]\cdot\mathbf{n} &= \mathbf{t} \quad \text{on } \partial\omega \cap \Gamma_N \\ \left[\mathbf{C}^+\boldsymbol{\epsilon}(\mathbf{u})\right]\cdot\mathbf{n} &= \mathbf{0} \quad \text{on } \partial\omega\backslash\partial\Omega \end{aligned} \tag{2.73}$$

admits a unique weak solution. As before, the solution $\mathbf{u}_\omega \in \mathcal{U}_\omega$ satisfies the variational problem

$$\int_\omega \mathbf{C}^+\boldsymbol{\epsilon}(\mathbf{u}_\omega) : \boldsymbol{\epsilon}(\mathbf{v})\mathrm{d}\mathbf{x} = \int_{\Gamma_N} \mathbf{t}\cdot\mathbf{v}\mathrm{d}s, \qquad \forall \mathbf{v} \in \mathcal{V}_\omega \tag{2.74}$$

Observe that, unlike the two-phase problem, the trial and test spaces given by

$$\mathcal{U}_\omega = \left\{\mathbf{u} \in H^1(\omega)^d : \mathbf{u}\big|_{\partial\omega\cap\Gamma_D} = \mathbf{g}\right\}, \qquad \mathcal{V}_\omega = \left\{\mathbf{v} \in H^1(\omega)^d : \mathbf{v}\big|_{\partial\omega\cap\Gamma_D} = \mathbf{0}\right\} \tag{2.75}$$

depend on the shape ω. To avoid trivial scenarios, we also require that $\Gamma_N \subseteq \partial\omega \cap \partial\Omega$ and $\partial\omega \cap \Gamma_D$ is non-trivial (i.e., has non-zero surface measure) for all admissible shapes ω (see

Figure 1.1).

The single-phase shape optimization problem $\mathbb{P}_S$ is of the form:

$$\inf_{\omega \in \mathcal{O}} J(\omega, \mathbf{u}_\omega) \tag{2.76}$$

where $\mathbf{u}_\omega$ solves (2.73). The objective function J is assumed to satisfy continuity conditions (2.10) and (2.11). For consistency, we require that J is defined in such a way that it does not depend on $\mathbf{u}$ over $\Omega \backslash \omega$. For example, in the case of the inverse problem (2.13), we shall require that $E \subseteq \omega$ for each $\omega \in \mathcal{O}$. For compliance minimization, the condition that $\partial\omega$ contains the boundary segment Γ_N where the nonzero tractions are applied ensures that J is defined appropriately. As an example of a generic objective that satisfies this condition, we can consider

$$J(\omega, \mathbf{u}) = \int_\Omega \chi_\omega g(\mathbf{u}) \mathrm{d}\mathbf{x} \tag{2.77}$$

where $g(\cdot)$ is a sufficiently regular functional.

Similar to the two-phase material distribution problem, there exists an extensive literature on the issue of existence of solutions for the shape optimization problem $\mathbb{P}_S$. Since $\mathbb{P}_S$ already begins with a smaller set of admissible shapes (in order for the state equation to be well defined), various researchers have identified smaller spaces of domains that possess a suitable uniform regularity to ensure existence of solutions. For example, Chenais [47] proved that a sufficient condition for existence of solutions is that the shapes in $\mathcal{O}$ satisfy the cone condition [1] or equivalently the Lipschitz property *uniformly*. We shall use the results of this work in what follows. In two-dimensions, Sverak proved a stronger result that only requires the number of connected components of $\Omega \backslash \omega$ to be bounded by a fixed number for all $\omega \in \mathcal{O}$ [155]. We refer to [6], [130] and the introduction of [109] for a review of various results in the literature.

A natural question concerns the relationship between $\mathbb{P}_S$ and the two-phase material distribution problem. Leaving aside the "size" of the space of admissible shapes, we note that the state equation (2.3), is an approximation to (2.73) when the compliant phase $\mathbf{C}^-$ has small stiffness. For each $0 < \varepsilon \leq 1$, let us denote by $\mathbf{u}_\omega^\varepsilon$ the solution to the boundary value problem associated with the Ersatz approximation given by

$$\int_\Omega \left[\chi_\omega + \varepsilon(1 - \chi_\omega)\right] \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}_\omega^\varepsilon) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}s, \qquad \forall \mathbf{v} \in \mathcal{V}_\Omega \tag{2.78}$$

Observe that this is the same as (2.4) with $\chi = \chi_\omega$ and $\mathbf{C}^- = \varepsilon \mathbf{C}^+$.

The so-called interface conditions

$$\left[\mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u})\right] \cdot \mathbf{n}_\omega = -\left[\varepsilon \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u})\right] \cdot \mathbf{n}_\omega, \quad \text{on } \partial\omega \backslash \partial\Omega \tag{2.79}$$

which are encoded in the weak form (2.78), approach (at least formally) the homogeneous Neumman boundary condition that is specified on the free boundary $\partial\omega\backslash\partial\Omega$ in (2.73). In fact, it can be shown that the computed solution to (2.78) indeed converges to the solution of (2.73) as the stiffness of the compliant phases vanishes. In particular, it is shown in [58] that for sufficiently small ε,

$$\|\mathbf{u}_\omega^\varepsilon - \mathbf{u}_\omega\|_{1,2,\omega} \leq C(\omega, \Omega, \mathbf{t}, \mathbf{g})\varepsilon \tag{2.80}$$

where $\mathbf{u}_\omega^\varepsilon$ and $\mathbf{u}_\omega$ denote the solutions to (2.78) and (2.73), respectively[12]. This estimate indicates that $\mathbf{u}_\omega^\varepsilon$, when restricted to ω, is a good approximation to $\mathbf{u}_\omega$. A weaker result, establishing convergence of $\mathbf{u}_\omega^\varepsilon\big|_\omega$ to $\mathbf{u}_\omega$, is given in section 3.1.4 of [3] for the analogous problem in conductivity. We use elements of this latter proof in Theorem 2.6.

Despite the relationship between the state equations, the link between the optimal solutions for the two class of problems is not obvious. As discussed before, the so-called Ersatz approach is used by the topology optimization community to solve $\mathbb{P}_S$ using the material distribution problem as its approximation. Sometimes this is attributed to reduction in the computational effort associated with the "Eulerian" framework of analysis afforded by the two-phase description. This may be viewed as a matter of choice since in principle and in practice, it is possible to carry out the analysis using a Lagrangian approach with the aid of remeshing techniques without necessarily restrict topological changes. For example, the evolution of domain can be done in the same manner as the existing level set methods on a fixed mesh while the analysis (which produces the front velocities) is carried out on a conforming finite element mesh (see example in [123]). However, most existing methods, especially level set formulations, rely so heavily on the such an approximation that it is difficult to separate the optimization algorithm from the choice of analysis.

We can analyze the issue of Ersatz approximation within the restriction framework of section 2.7. A similar analysis is carried out in [117] for a scalar transmission problem. As pointed out in [117], by choosing the parameter θ in (R1) sufficiently large, we can ensure that the shapes corresponding to $\varphi \in \mathcal{F}_R$ have a uniform Lipschitz constant thereby recovering the admissible space considered by Chenais [47]. For example, if $\theta p > n$, then $W^{1+\theta,p}(\Omega)$ is embedded in $W^{1,\infty}(\Omega)$, which is exactly the space of Lipschitz functions over Ω [74]. From the transversality condition and implicit function theorem of Clarke [50], it follows that the boundary $\partial\omega = \{\varphi = 0\}$ is Lipschitz. In the remainder of this chapter, we define

$$\mathcal{O} = \{\omega \subseteq \Omega : \chi_\omega = H(\varphi) \text{ for some } \varphi \in \mathcal{F}_R\} \tag{2.81}$$

with the additional requirements associated with the boundary conditions implicitly satis-

[12]This result [58] is given for the case of a scalar elliptic equation but it is expected that it would also hold for elasticity.

fied. In addition to being compact with respect to the $L^1(\Omega)$-metric, this a space of shapes satisfy the so-called "uniform extension" property [47, 61]. This means that the extension operator from the shapes to the extended domain Ω is uniformly continuous. More specifically, for some constant $K > 0$ and for all $\omega \in \mathcal{O}$, there is a linear continuous *extension* operator $\Psi_\omega : H^1(\omega)^d \to H^1(\Omega)^d$ such that

$$\|\Psi_\omega(\mathbf{u})\|_{1,2,\Omega} \leq K \|\mathbf{u}\|_{1,2,\omega} \tag{2.82}$$

This property will play a central role in the proof of the main result in this section.

For the sake of concreteness and brevity, we limit our attention to the compliance minimization where

$$J(\omega, \mathbf{u}) = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{u} \mathrm{d}s + \lambda \int_\Omega \chi_\omega \mathrm{d}\mathbf{x} \tag{2.83}$$

and assume homogenous displacement boundary data $\mathbf{g} = \mathbf{0}$. Given a shape $\omega \in \mathcal{O}$, the compliance associated with the Ersatz approximation is given by $J(\omega, \mathbf{u}^\varepsilon_\omega)$ while the compliance associated with the single-phase problem is $J(\omega, \mathbf{u}_\omega)$. Let $\omega_\varepsilon \in \mathcal{O}$ denote an optimal solution to the Ersatz approximation. We will show that the sequence of shapes ω_ε converges to a solution of $\mathbb{P}_S$. Note, however, that the proof also extends to the generic objective function defined by (2.77). This result provides a justification for using the Ersatz approximation to solve a material distribution problem as a surrogate for the single-phase shape optimization problem.

Theorem 2.6. *For each $0 < \varepsilon \leq 1$, let ω_ε denote a minimizer of $J(\omega, \mathbf{u}^\varepsilon_\omega)$ in $\mathcal{O}$. There exists $\omega^* \in \mathcal{O}$ such that, up to a subsequence, $\chi_{\omega_\varepsilon} \to \chi_{\omega^*}$ in $L^1(\Omega)$ as $\varepsilon \to 0$. Moreover, ω^* is a minimizer of $J(\omega, \mathbf{u}_\omega)$ in $\mathcal{O}$.*

Proof. By the compactness result of section 2.5, there exists $\omega^* \in \mathcal{O}$ such that for some subsequence, again denoted by ω_ε, $\chi_{\omega_\varepsilon} \to \chi_{\omega^*}$ in $L^1(\Omega)$ as $\varepsilon \to 0$. By going to another subsequence, we may assume that the convergence of the characteristic functions is pointwise.

Next we define an appropriate limit for the corresponding displacement fields $\mathbf{u}^\varepsilon_{\omega_\varepsilon}$ which, by definition, solve the following variational problem

$$\int_\Omega \left[\chi_{\omega_\varepsilon} + \varepsilon \left(1 - \chi_{\omega_\varepsilon}\right)\right] \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}s, \qquad \forall \mathbf{v} \in \mathcal{V}_\Omega \tag{2.84}$$

Notice that we cannot appeal to the estimate (2.9) since the ellipticity constant c now depends on ε and goes to zero with ε. Substituting $\mathbf{v} = \mathbf{u}^\varepsilon_{\omega_\varepsilon}$ in (2.84), and using (2.74) with $\omega = \omega_\varepsilon$ (note that $\mathbf{u}^\varepsilon_{\omega_\varepsilon}\big|_{\omega_\varepsilon} \in \mathcal{V}_{\omega_\varepsilon}$) for the right hand side

$$\int_{\omega_\varepsilon} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) \mathrm{d}\mathbf{x} + \varepsilon \int_{\Omega \setminus \omega_\varepsilon} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) \mathrm{d}\mathbf{x} = \int_{\omega_\varepsilon} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) \mathrm{d}\mathbf{x} \tag{2.85}$$

From the standard estimate for (2.74),

$$\|\mathbf{u}_{\omega_\varepsilon}\|_{1,2,\omega_\varepsilon} \leq K \|\mathbf{t}\| \tag{2.86}$$

and by virtue of Korn's inequality[13] $\left\|\nabla \mathbf{u}^\varepsilon_{\omega_\varepsilon}\right\|_{0,2,\omega_\varepsilon} \leq \tilde{K} \left\|\boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon})\right\|_{0,2,\omega_\varepsilon}$, we can conclude from (2.85) that

$$\left\|\nabla \mathbf{u}^\varepsilon_{\omega_\varepsilon}\right\|_{0,2,\omega_\varepsilon} + \varepsilon^{1/2} \left\|\nabla \mathbf{u}^\varepsilon_{\omega_\varepsilon}\right\|_{0,2,\Omega\backslash\omega_\varepsilon} \tag{2.87}$$

is bounded for all ε. Another use of Korn's inequalities shows that the sequence $\mathbf{u}^\varepsilon_{\omega_\varepsilon}\big|_{\omega_\varepsilon}$ is bounded in $H^1(\omega_\varepsilon)^d$. From the uniform extension property (cf. (2.82)), the sequence of extensions $\Psi_{\omega_\varepsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}\big|_{\omega_\varepsilon})$ is also bounded in $H^1(\Omega)^d$ and therefore, up to a subsequence, converges weakly to some limit $\mathbf{u}^* \in H^1(\Omega)^d$.

By analyzing the limit of (2.84) as $\varepsilon \to 0$, we next show that $\mathbf{u}^*\big|_{\omega^*}$ is the displacement field associated with the limit shape ω^*. Fix $\mathbf{v} \in \mathcal{V}_\Omega$. We have:

$$\begin{aligned}
\int_\Omega \chi_{\omega_\varepsilon} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} &= \int_\Omega \chi_{\omega_\varepsilon} \mathbf{C}^+ \boldsymbol{\epsilon}\left[\Psi_{\omega_\varepsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}\big|_{\omega_\varepsilon})\right] : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \\
&\to \int_\Omega \chi_{\omega^*} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^*) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \\
&= \int_{\omega^*} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^*) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x}
\end{aligned} \tag{2.88}$$

The proof of this convergence is similar to that in the Proposition 2.1. By boundedness of $\varepsilon^{1/2} \nabla \mathbf{u}^\varepsilon_{\omega_\varepsilon}$ in $\Omega\backslash\omega_\varepsilon$, we have for the other term in the bilinear form,

$$\int_\Omega \varepsilon(1 - \chi_{\omega_\varepsilon}) \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_{\omega_\varepsilon}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} \to 0 \tag{2.89}$$

as $\varepsilon \to 0$. Therefore, we have for any $\mathbf{v} \in \mathcal{V}_\Omega$

$$\int_{\omega^*} \mathbf{C}^+ \boldsymbol{\epsilon}(\mathbf{u}^*) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x} = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}s \tag{2.90}$$

Since any $\mathbf{v} \in \mathcal{V}_{\omega^*}$ can be extended to an element of $\mathcal{V}_\Omega$, this proves that $\mathbf{u}^*\big|_{\omega^*} = \mathbf{u}_{\omega^*}$. Moreover, it follows from continuity of J that

$$J(\omega_\varepsilon, \mathbf{u}^\varepsilon_{\omega_\varepsilon}) \to J(\omega^*, \mathbf{u}_{\omega^*}) \tag{2.91}$$

To complete the proof of the statement, we must show that ω^* is indeed a solution to $\mathbb{P}_S$. To see this, take any $\tilde{\omega} \in \mathcal{O}$. By a similar argument as above, we can show that (this also

[13]We must note that the constant associated with Korn's inequality does depend on the domain ω_ε. However, this constant is bounded due to the uniform Lipschitz property of $\mathcal{O}$ [70]. The same is true for the constant K in (2.86).

follows from estimate (2.80)),

$$J(\tilde{\omega}, \mathbf{u}^{\varepsilon}_{\tilde{\omega}}) \to J(\tilde{\omega}, \mathbf{u}_{\tilde{\omega}}) \tag{2.92}$$

For each ε, we have by optimality of ω_ε that

$$J(\omega_\varepsilon, \mathbf{u}^{\varepsilon}_{\omega_\varepsilon}) \leq J(\tilde{\omega}, \mathbf{u}^{\varepsilon}_{\tilde{\omega}}) \tag{2.93}$$

Taking the limit of (2.93) as $\varepsilon \to 0$ and using (2.91) and (2.92) yields $J(\omega^*, \mathbf{u}_{\omega^*}) \leq J(\tilde{\omega}, \mathbf{u}_{\tilde{\omega}})$. □

The above proof of convergence in (2.91) can be simplified by making use of estimate (2.80) since it implies

$$\left|J(\omega_\varepsilon, \mathbf{u}^{\varepsilon}_{\omega_\varepsilon}) - J(\omega_\varepsilon, \mathbf{u}_{\omega_\varepsilon})\right| \leq kC(\omega_\varepsilon, \Omega, \mathbf{t}, \mathbf{g})\varepsilon \tag{2.94}$$

and we can treat $J(\omega_\varepsilon, \mathbf{u}_{\omega_\varepsilon})$ following the single-phase analysis of Chenais [47]. The only issue is that the constant in this estimate, as stated, depends on the shapes ω_ε. Though expected, it is unclear from the proof in [58] if this constant is uniform on $\mathcal{O}$.

Chapter 3

Tikhonov Regularization and a Splitting Algorithm

We have discussed in the previous two chapters the lack of existence of solutions to the classical topology optimization problems. For example, it was shown that the problem of minimizing compliance in structural optimization favors non-convergent minimizing sequences of shapes that exhibit progressively finer features. The commonly used density formulations, such as the popular Solid Isotropic Material with Penalization (SIMP) approach [20, 136, 135], wherein characteristic functions representing the shapes are replaced by density fields, continue to suffer from this pathology as the built-in penalization mechanism recovers solutions that are nearly binary in the optimal regime. A manifestation of this behavior in the finite element discretization of the problem is the dependence of solutions on the level of refinement of the spatial discretization. The problem of mesh dependency, just as the ill-posedness of the continuum problem, has led many researchers to devise formulations that are stable under mesh refinement. The search for a robust and yet mathematically consistent approach continues as evidenced by the growing number of publications on this issue [31, 131, 85, 86, 172, 141, 94, 171].

We limit the following literature survey to density formulations but, as discussed above, the difficulty stems from a fundamental property of the original topology optimization problem and therefore similar measures are needed for other parameterizations of geometries, most notably the implicit functions methods (see chapter 2). Placing a restriction on the perimeter of the admissible shapes is perhaps the oldest approach in the field. The set of admissible characteristic functions is restricted to a subset of functions of bounded variation with a prescribed upper bound on their total variation [11]. Existence of solutions follows from relative compactness of bounded sequences in BV in the L^1-topology and carries over to the corresponding density formulation [127]. Due to the difficulty of discretization of functions of bounded variation and robust linearization of the total variation functional [181], the perimeter formulation has perhaps fallen out of favor in the topology optimization community though it remains a significant point of reference. More recent approaches are based on the concept of filtering, which consists of implicitly imposing regularity on each admissible density function by means of convolution of an auxiliary field with a fixed and smooth filter. With such construction, all the admissible densities inherit the regularity of the filter, thereby ensuring compactness of the design space in the L^1-topology [31].

The filtering approach works well in practice since no explicit constraints on regularity of density functions are needed. Moreover the level of complexity of the final solutions (in fact all the admissible densities) is controlled directly by the regularity of the filtering kernel. *Herein lies the major drawback: the smoother the filtering kernel, the larger the amount of the intermediate densities since the transition between the extreme values of density over the domain cannot occur too rapidly.* Therefore, with more complexity control comes more "gray" regions and this, in some respect, undermines the basic premise of the density approach in that near characteristic functions are no longer recovered in the optimal regime (i.e., "0-1" or "black-and-white" designs are not obtained). We note that a similar issue arises in the slope constraint method of Petersson and Sigmund [128] where regularity is imposed explicitly by placing a constraint on the pointwise magnitude of the density gradient.

A recent trend [85, 141, 94, 171] has focused on the so-called nonlinear filtering approaches. As pointed out in section 1.6 (see also [164]), the introduced nonlinearity usually amounts to a modification of the material interpolation model (e.g., SIMP) rather than a change in the filtering operation. For example, in the Heaviside filtering approach [85], both power law relations of SIMP—dependence of Young's modulus on ρ^p and volume on ρ—are augmented by the use of a smoothed Heaviside function. The additional parameter defining the sharpness of the Heaviside function controls the amount of gray that appears. To obtain good solutions, these parameters are often carefully increased throughout the course of the optimization algorithm.

It is no surprise that more fine-tuning is needed as one moves away from the simplicity of the original penalized density formulation. These schemes contain multiple penalty parameters in addition to the averaging effect of the underlying filter, which can adversely affect the quality of the reciprocal approximations of the objective function in the commonly used optimization algorithms, such as MMA [153], ultimately slowing down convergence rate or compromising the quality of the final solutions.

In this chapter, we examine the use of a simple Tikhonov-type regularization scheme for topology optimization. The admissible densities are defined as a subset of H^1 space with a uniform bound on their norm. In practice, this is achieved by appending the H^1 semi-norm of the density function as a penalty term to the objective function. Existence of solutions follows from the compact embedding of H^1 in L^1. Such an approach has been previously studied by Borrvall in a review paper [28] where he examines penalty terms involving the L^p-norm of the density gradient (recovering the total variation regularization for $p = 1$ and the slope constraint for $p = \infty$). This term also appears in phase field methods [32, 174, 38, 184, 157, 60] as an interfacial energy term and is accompanied by a double-well potential penalizing intermediate densities. The two terms taken together with appropriately chosen coefficients (cf. eq. (3.38)) serve as an approximation to the perimeter of the interface.

As we shall see in the present setting, filtering, in the form of inverse of the Helmholtz operator, naturally appears when the optimization iterations are obtained from a semi-implicit discretization of gradient flow associated with the regularized objective function. In contrast to the density and sensitivity filters, the effects of regularization term appear through smoothening of the gradient descent steps associated with the unregularized objective function. The next iterate is obtained from projection of the provisional density onto the space of admissible densities in order to enforce the 0-1 (void-solid) box constraint (and the pointwise move limit commonly introduced to stabilize the density evolution). We will show that, with a particular choice of projection map, this update scheme recovers the well-known forward-backward splitting algorithm [106, 46, 53, 33]. This provides an alternative perspective on the proposed approach, which is used to further investigate the theoretical and computational aspects of the algorithm leveraging the abundant literature of operator splitting and related methods. In particular, the uncoupled treatment of the Tikhonov term in the forward-backward method can be useful for more general (possibly nonsmooth) regularization approaches for topology optimization.

Finally, we note that the separation of filtering (i.e., the smoothening effect of regularization term) and projection operation in the general case offers some flexibility. In the extreme case of the L^2-projection, this algorithm, with the aid of SIMP penalization, eliminates nearly all intermediate densities regardless of the level of regularization and complexity of the final shapes. Numerically we have observed qualitatively good solutions obtained in moderate number of iterations without the need for continuation on the SIMP penalty parameter.

For the sake of brevity, in this chapter, we denote the inner product and norm associated with $L^2(\Omega)$ by $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$, respectively. Similarly, the inner product, norm and seminorm associated with $H^1(\Omega)$ are denoted by $\langle\cdot,\cdot\rangle_1$, $\|\cdot\|_1$ and $|\cdot|_1$, respectively. Moreover, for $u, v \in H^1(\Omega)$, we shall define the inner product $\langle u, v\rangle_\alpha := (1-\alpha)\langle u, v\rangle + \alpha\langle u, v\rangle_1$. Observe that for $0 < \alpha < 1$, the associated norm given by $\|\cdot\|_\alpha := \langle\cdot,\cdot\rangle_\alpha^{1/2}$ is equivalent to the standard norm $\|\cdot\|_1$.

3.1 Model problem

For the sake of completeness, we briefly recall the density formulation of the compliance minimization problem. Let $\Omega \subseteq \mathbb{R}^d$, $(d = 2, 3)$ be a bounded set with a sufficiently smooth boundary representing the design domain for the problem and consider a nontrivial partition[1] of the boundary $\partial\Omega$ into disjoint sets Γ_D and Γ_N. The objective function is given

[1] That is, $\Gamma_D \cap \Gamma_N = \emptyset$, $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N$, and $|\Gamma_D| \neq 0$

by

$$J(\rho) = \int_{\Gamma_N} \mathbf{u}_\rho \cdot \mathbf{t} \mathrm{d}s + \lambda \int_\Omega \rho \mathrm{d}\mathbf{x} \tag{3.1}$$

The displacement field $\mathbf{u}_\rho$ is the solution to the elasticity boundary value problem, given in the weak form by

$$a(\mathbf{u}, \mathbf{v}; \rho) = \ell(\mathbf{v}), \qquad \forall \mathbf{v} \in \mathcal{V} \tag{3.2}$$

where $\mathcal{V} = \{\mathbf{u} \in H^1(\Omega)^d : \mathbf{u}|_{\Gamma_D} = \mathbf{0}\}$ is the space of admissible displacements and, as before,

$$a(\mathbf{u}, \mathbf{v}; \rho) = \int_\Omega \mathbf{C}_\rho \boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v}) \mathrm{d}\mathbf{x}, \qquad \ell(\mathbf{v}) = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}s \tag{3.3}$$

are the energy bilinear and load linear forms. Moreover, $\boldsymbol{\epsilon}(\mathbf{u}) = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$ is the linearized strain tensor, $\mathbf{t} \in L^2(\Gamma_N)^d$ is the prescribed tractions on Γ_N. In the SIMP model,

$$\mathbf{C}_\rho = [\varepsilon + (1 - \varepsilon)\, \rho^p]\, \mathbf{C} \tag{3.4}$$

where $p > 1$ is the penalty parameter, $\mathbf{C}$ is the elasticity tensor of the constituent material, and $0 < \varepsilon \ll 1$ is the Ersatz parameter which ensures the well-posedness of the governing equations for every non-negative $\rho \in L^\infty(\Omega)$. In particular, observe that the bilinear form is continuous and coercive (cf. (2.6) and (2.7)) and so given a measurable density function ρ taking values between zero and one, (3.2) admits a unique weak solution $\mathbf{u}_\rho$ in $H^1(\Omega)^d$. For future use, we also recall that by the principle of minimum potential energy, $\mathbf{u}_\rho$ is alternatively characterized by

$$\mathbf{u}_\rho = \operatorname*{argmin}_{\mathbf{v} \in \mathcal{V}} \left[\frac{1}{2} a(\mathbf{v}, \mathbf{v}; \rho) - \ell(\mathbf{v}) \right] \tag{3.5}$$

where the term in the brackets is the potential energy associated with deformation field $\mathbf{v}$.

Since the second term in (3.1) represents a penalty on the volume of the material used, minimizing this objective amounts to finding the stiffest arrangement of $\mathbf{C}$ while using the least amount of material. The parameter $\lambda > 0$ determines the trade-off between the stiffness provided by the material and the amount of material that is used. Because of the monotonicity of these competing terms, it is expected that in the optimal regime, the density functions take extreme values of 0 and 1 throughout most of Ω provided that the penalty parameter p is sufficiently large. This fact was proven in [145, 133, 112] within the discrete setting where existence of solutions follows from the finite dimensionality of the problem.

We remark that frequently in the formulation of the minimum compliance problem,

instead of using a penalty term as in (3.1), an explicit constraint of the form

$$\int_\Omega \rho \mathrm{d}\mathbf{x} \leq \overline{v} \left|\Omega\right|, \qquad 0 \leq \overline{v} \leq 1 \tag{3.6}$$

is placed on the volume of the design. The two approaches are equivalent in the sense that for any prescribed volume fraction $\overline{v}$, there exists a penalty parameter λ such that the minimizer of (3.1) is also a solution to the problem with the explicit volume constraint. The converse is also true in that given $\lambda > 0$, we can define the equivalent $\overline{v}$ to be $\left|\Omega\right|^{-1} \int_\Omega \rho_\lambda^* \mathrm{d}\mathbf{x}$ where ρ_λ^* is a minimizer of the $J(\rho)$. The drawback of the present formulation is that one needs to find a suitable value for the penalty parameter λ, which may not be immediately obvious. On the other hand, it has been our experience that for the compliance minimization problem, the penalty approach is more forgiving (the volume can exceed the final volume in the course of the algorithm) and leads to qualitatively better solutions.

3.2 Restriction and Tikhonov regularization

To guarantee existence of solutions to the minimum compliance problem, the space of admissible densities must be restricted to a sufficiently regular subset of $L^\infty\left(\Omega; [0,1]\right)$. One sufficient condition, investigated here, is to require each admissible ρ to belong to $H^1(\Omega)$ with $\left|\rho\right|_1 \leq M$ for some fixed positive constant M (see also, [24, 28]). In particular, we define the space of admissible densities to be

$$\tilde{\mathcal{A}} = \left\{\rho \in H^1(\Omega) : 0 \leq \rho \leq 1 \text{ a.e.}, \left|\rho\right|_1 \leq M\right\} \tag{3.7}$$

Based on the compactness result of chapter 2, to establish existence of solutions, it is sufficient to prove that this space is compact in $L^1(\Omega)$. We note that the existence of solutions for essentially the same problem is analyzed in [24] but only a weaker result that requires a bound on p is proven.

Proposition 3.1. *The space of admissible densities $\tilde{\mathcal{A}}$ is compact in $L^1(\Omega)$.*

Proof. Let ρ_n be a sequence in $\tilde{\mathcal{A}}$. Since $\left\|\rho_n\right\| \leq \left|\Omega\right|$ and $\left|\rho_n\right|_1 \leq M$, the sequence is bounded in $H^1(\Omega)$ and so there exists $\hat{\rho} \in H^1(\Omega)$ and a subsequence, again denoted by ρ_n, such that $\rho_n \rightharpoonup \hat{\rho}$ weakly in $H^1(\Omega)$. By Rellich-Kondrachov theorem [74] and going to another subsequence, $\rho_n \to \hat{\rho}$ strongly in $L^1(\Omega)$. It remains to prove that $\hat{\rho}$ belongs to $\tilde{\mathcal{A}}$. To see that $\hat{\rho}$ satisfies the bound constraint, we can go to another subsequence such that the convergence of ρ_n to ρ is pointwise. It follows that $0 \leq \hat{\rho} \leq 1$ almost everywhere. Moreover, from the lower semicontinuity of norm under weak convergence, $\left|\hat{\rho}\right|_1 \leq \liminf_n \left|\rho_n\right|_1 \leq M$. □

An alternative to adding the "compactifying" constraint $|\rho|_1 \leq M$ is to modify the objective function as follows

$$\tilde{J}(\rho) = J(\rho) + \frac{\beta}{2} |\rho|_1^2 \tag{3.8}$$

where β is a positive coefficient. As in the case of the volume constraint, we can show that minimization of $J(\rho)$ over $\tilde{\mathcal{A}}$ is equivalent to the minimization of $\tilde{J}(\rho)$ over the space

$$\mathcal{A} = \left\{ \rho \in H^1(\Omega) : 0 \leq \rho \leq 1 \text{ a.e.} \right\} \tag{3.9}$$

in the sense that one obtains the same solution provided that M and β are suitably chosen. Observe that the Tikhonov regularization term $\frac{\beta}{2} |\rho|_1^2$ ensures that the minimizing sequences in $\mathcal{A}$ are compact in $L^1(\Omega)$ even though $\mathcal{A}$ itself is not compact. Henceforth, we focus on the latter form (3.8).

The Tikhonov regularization in (3.8) is commonly used for ill-posed inverse problems and we refer the reader to the abundant literature available (see, for example, [71] and references therein). We remark that from a theoretical perspective, regularization of densities in $H^1(\Omega)$ is more restrictive than in $BV(\Omega)$ which is sufficient for guaranteeing existence of solutions for the compliance problem. However, from a practical point of view, $H^1(\Omega)$ is a simpler space to work with and, unlike total variation, the Tikhonov regularizer is smooth (differentiable) and has a quadratic form. Moreover, practical (engineering) considerations of complexity control in topology optimization (i.e., controlling feature size and orientation) can be accommodated here. A more general form of the regularization term is given by

$$R(\rho) = \frac{1}{2} \langle \nabla\rho, \boldsymbol{\kappa} \nabla\rho \rangle = \frac{1}{2} \int_\Omega \nabla\rho(\mathbf{x}) \cdot \boldsymbol{\kappa}(\mathbf{x}) \nabla\rho(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{3.10}$$

where $\boldsymbol{\kappa}$ belongs to $L^\infty(\Omega)^{d\times d}$ and for some constants $0 < k_1 < k_2$ satisfies $k_1\mathbf{I} \leq \boldsymbol{\kappa}(\mathbf{x}) \leq k_2\mathbf{I}$ in the sense of quadratic forms for all $\mathbf{x} \in \Omega$. The existence of solutions can be shown in a similar manner since $R(\rho)$ is equivalent to the H^1 semi-norm. A suitable choice of $\boldsymbol{\kappa}$ can ensure the desired regularity of ρ in various parts of Ω, a fact illustrated later through a numerical example.

3.3 Proposed optimization algorithm

In this section, we present a rather formal derivation of the proposed optimization algorithm in order to better illustrate the main concept. A more rigorous treatment is given in the next chapter. The goal is to solve the regularized topology optimization problem

$$\min_{\rho\in\mathcal{A}} \tilde{J}(\rho) = J(\rho) + R(\rho) \tag{3.11}$$

where $J(\rho)$ is defined in (3.1), $R(\rho) = \frac{\beta}{2}\,|\rho|_1^2$, and the space of admissible density functions is given by (3.9). The *unconstrained* gradient flow corresponding to this minimization is given by

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\tilde{J}'(\rho) = -\left[J'(\rho) + R'(\rho)\right] \tag{3.12}$$

where t is a pseudo-time variable that formally characterizes the evolution of the density function $\rho(t)$ along this descent flow. The gradient of compliance in the above expression can be written as [24]

$$J'(\rho) = -\left(1-\varepsilon\right) p\rho^{p-1}\mathbf{C}\boldsymbol{\epsilon}(\mathbf{u}_\rho) : \boldsymbol{\epsilon}(\mathbf{u}_\rho) + \lambda \tag{3.13}$$

The first term is a strain energy density field whose evaluation requires the solution $\mathbf{u}_\rho$ to (3.2). The gradient of regularization term is simply

$$R'(\rho) = -\beta\Delta\rho \tag{3.14}$$

provided that $\partial\rho/\partial\mathbf{n} = 0$ on $\partial\Omega$[2]. An explicit discretization of (3.12) yields the usual gradient descent update, which due to the presence of the Laplacian term in $R'(\rho)$ requires small time increments, governed by the Courant-Friedrichs-Lewy condition, and subsequently a large number of iterations. Also, the resulting discrete dynamics may introduce additional regularization effects depending on the number and size of time increments, as discussed in [14], which in turn depend on the spatial discretization of ρ.

Due to the dependence of $J'(\rho)$ on the state equation, an implicit treatment of $J'(\rho)$ in (3.12) is not possible. Thus we consider a *semi-implicit* temporal discretization of the gradient flow equation, an approach also advocated by Bourdin and Chambolle [32] in their phase field method. Considering a fixed time increment τ and denoting by ρ_n the current density iterate at $t = n\tau$, the semi-implicit discretization of (3.12) takes the form

$$\frac{\rho^*_{n+1} - \rho_n}{\tau} = -J'(\rho_n) - R'(\rho^*_{n+1}) \tag{3.15}$$

where ρ^*_{n+1} is the interim or provisional iterate that may lie outside the admissible space $\mathcal{A}$ (i.e., violate the bounds on the admissible densities). We define the next iterate ρ_{n+1} to

[2] A remark is in order regarding the regularity of ρ and its boundary conditions implied by (3.14). We will essentially use the variational form of the gradient flow (3.12):

$$\langle \mathrm{d}\rho/\mathrm{d}t, \psi\rangle = \mathrm{d}\tilde{J}(\rho)\,[\psi]\,, \quad \forall\psi \in H^1(\Omega)$$

to evolve the density function (cf. eq. (3.31)). Here $\mathrm{d}\tilde{J}(\rho)\,[\psi]$ is the Gateaux derivative of $\tilde{J}$ at ρ in the direction of ψ. Observe $\mathrm{d}J(\rho)\,[\psi] = \langle J'(\rho), \psi\rangle$ where $J'(\rho)$ is defined in (3.13) and $\mathrm{d}R(\rho)\,[\psi] = \beta\,\langle\nabla\rho, \nabla\psi\rangle$. If additionally $\rho \in H^2(\Omega)$ and $\partial\rho/\partial\mathbf{n} = 0$ on $\partial\Omega$, then $\mathrm{d}R(\rho)\,[\psi] = \langle -\beta\Delta\rho, \psi\rangle = \langle R'(\rho), \psi\rangle$ but we do not need to place any additional constraints beyond $\rho \in H^1(\Omega)$.

be the *projection* of ρ^*_{n+1} onto $\mathcal{A}$, that is,

$$\rho_{n+1} = \Pi_{\mathcal{A}}(\rho^*_{n+1}) \tag{3.16}$$

The algorithm defined by (3.15) and (3.16) may be viewed as a semi-implicit form of the gradient projection method [39], which consists of the projection of an explicit update of (3.12). The proposed algorithm is similar to the two-step procedures for solution of the incompressible Navier-Stokes equations (cf. [114]). The standard optimality criteria (OC) algorithm also has the same two-step structure where the projection enforcing the box constraints constitutes the final step [24].

A minor modification to (3.16) allows us to accommodate a common approach for stabilizing the topology optimization algorithm, which consists of limiting the *point-wise* change in density in consecutive iterations. We can simply replace $\mathcal{A}$ in (3.16) by the subset

$$\mathcal{A}_n = \{\rho \in \mathcal{A} : \rho_n - m \leq \rho \leq \rho_n + m \text{ a.e.}\} \tag{3.17}$$

where $m \in (0, 1]$ is a prescribed *move limit*. Defining $\rho^{\mathsf{U}}_n = 1 \vee (\rho_n + m)$ and $\rho^{\mathsf{L}}_n = 0 \wedge (\rho_n - m)$, we can write

$$\mathcal{A}_n = \left\{\rho \in H^1(\Omega) : \rho^{\mathsf{L}}_n \leq \rho \leq \rho^{\mathsf{U}}_n \text{ a.e.}\right\} \tag{3.18}$$

The next iterate, accounting for this move limit constraint, is then

$$\rho_{n+1} = \Pi_{\mathcal{A}_n}\left(\rho^*_{n+1}\right) \tag{3.19}$$

Although reducing the time step τ can also increase the conservatism of the algorithm, based on our numerical experience so far, the move limit approach tends to perform better in practice and allows the use of a larger fixed step size τ. Of course, by setting $m = 1$, we get $\mathcal{A}_n = \mathcal{A}$ recovering the update (3.16).

In the numerical results in this chapter, the criterion for convergence of the algorithm is based on the change in the value of objective function $|\tilde{J}(\rho_{n+1}) - \tilde{J}(\rho_n)|/|\tilde{J}(\rho_n)|$. Aside from the choice of the projection map $\Pi_{\mathcal{A}_n}$, the proposed update scheme contains two algorithmic parameters, namely τ and m, both of which are independent of β and the spatial discretization of ρ. More generally, in an extension of this algorithm, τ and m can be varied during the course of the algorithm to speed up convergence and/or improve stability (see chapter 4).

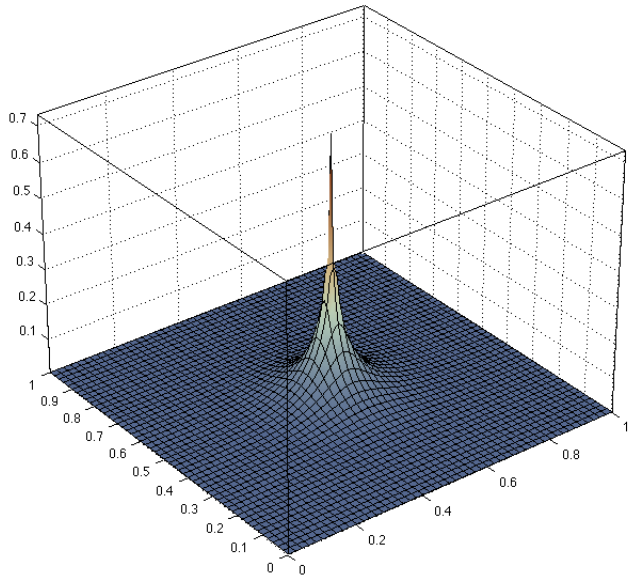


Figure 3.1: Approximate Green's function computed numerically on a square domain Ω

3.4 Comparison with filtering methods

Substituting (3.14) into (3.15) and rearranging, we can see that the provisional iterate satisfies the modified Helmholtz equation

$$\rho_{n+1}^* - \beta\tau\Delta\rho_{n+1}^* = \rho_n - \tau J'(\rho_n) \tag{3.20}$$

with homogenous Neumann boundary conditions. It is interesting to note that the right-hand side term is the gradient descent update, with step size τ, for the *unregularized* objective function $J(\rho)$. Denoting by $G_{\beta\tau}$ the Green's function associated with Helmholtz operator in (3.20), we can write

$$\rho_{n+1}^*(\mathbf{x}) = \int_\Omega G_{\beta\tau}(\mathbf{x},\mathbf{y})\left[\rho_n(\mathbf{y}) - \tau J'(\rho_n)(\mathbf{y})\right]\mathrm{d}\mathbf{y} \tag{3.21}$$

This shows that the candidate update ρ_{n+1}^* is obtained by filtering the "original" gradient descent update by the Gaussian kernel $G_{\beta\tau}$ whose support size depends on $\beta\tau$ (see Figure 3.1). Although the Helmholtz equation has been previously used as a means to carry out filtering in topology optimization [102, 94], the update expression (3.21) fundamentally differs from both density and sensitivity filtering methods.

It is in fact instructive to compare the proposed algorithm with the usual linear density filter, which essentially consists of the same two steps of smoothening and projection. Defining the filtering operator $\mathcal{P} = (1 - \beta\tau\Delta)^{-1}$, the update equation for the proposed

algorithm can be written succinctly as

$$\rho_{n+1} = (\Pi_{\mathcal{A}_n} \circ \mathcal{P}) \left[\rho_n - \tau J'(\rho_n)\right] \tag{3.22}$$

In the linear filtering method, the space of admissible densities is defined by (see section 1.3)

$$\overline{\mathcal{A}} = \{\mathcal{P}(\eta) : \eta \in L^\infty(\Omega; [0,1])\} \tag{3.23}$$

The idea is that each density function automatically inherits its smoothness from the properties of $\mathcal{P}$ while the auxiliary functions $\eta \in L^\infty(\Omega; [0,1])$ are updated in the optimization algorithm, and so smoothness does not need to be enforced explicitly. If the gradient projection method is adopted[3] to do the optimization, the update expression for the auxiliary field is

$$\eta_{n+1} = \Pi_{\overline{\mathcal{A}}_n} \left[\eta_n - \tau J'_\eta(\eta_n)\right] \tag{3.24}$$

where J'_η denotes the gradient of J with respect to η and, analogously to (3.17), $\overline{\mathcal{A}}_n$ consists of $\eta \in L^\infty(\Omega; [0,1])$ such that $|\eta - \eta_n| \leq m$ almost everywhere. The expression for the density update ρ_{n+1} is thus

$$\rho_{n+1} = \mathcal{P}(\eta_{n+1}) = \left(\mathcal{P} \circ \Pi_{\overline{\mathcal{A}}_n}\right) \{\eta_n - \tau \mathcal{P}\left[J'(\rho_n)\right]\} \tag{3.25}$$

Comparing expressions (3.22) and (3.25), the most notable difference is the order of projection and filtering. In the density filter, by construction, the smoothness of the densities is dictated by the properties of $\mathcal{P}$ (e.g., the value of $\beta\tau$) while in the proposed algorithm, the projection map also plays a role in the smoothness of the update. In the density filtering, η_n is typically binary in the optimal regime in the presence of SIMP penalization and yet, as discussed in the introduction, $\rho_n = \mathcal{P}(\eta_n)$ may contain large regions of intermediate densities depending on the smoothening effect of the filtering map $\mathcal{P}$. In the proposed algorithm, the projection map $\Pi_{\mathcal{A}_n}$ can be defined such that near binary densities are allowed. Of course, the two spaces $\mathcal{A}_n$ and $\overline{\mathcal{A}}_n$ have different structures and require different projection maps. The precise description of the projection operation is discussed next.

3.5 Definition of the projection map

We proceed to explore the possible definitions for the projection map. To this effect, consider projection with respect to the metric generated by $\|\cdot\|_\alpha$ defined in section the

[3]The optimality criteria (OC) is usually preferred to gradient (steepest) descent in structural optimization. We refer to [13] on the relationship between the two methods.

introduction. For each $\psi \in H^1(\Omega)$, let

$$\Pi^{\alpha}_{\mathcal{B}}(\psi) := \operatorname*{argmin}_{\rho \in \mathcal{B}} \|\psi - \rho\|_{\alpha} \tag{3.26}$$

As noted before, $\|\cdot\|_{\alpha}$ defines an equivalent norm to the usual H^1 norm for $0 < \alpha < 1$. Provided that $\mathcal{B}$ is a closed convex subset of $H^1(\Omega)$, the projection $\Pi^{\alpha}_{\mathcal{B}}(\psi)$ exists and is unique for any $\psi \in H^1(\Omega)$. It is straightforward to show that $\mathcal{A}_n$ is closed and convex in $H^1(\Omega)$ and since $\rho^*_{n+1} \in H^1(\Omega)$ (cf. (3.20)), the update ρ_{n+1} is well-defined if we set $\Pi_{\mathcal{A}_n} = \Pi^{\alpha}_{\mathcal{A}_n}$ in (3.19). Note that even with the addition of an explicit volume constraint (cf. (3.6)), the space $\mathcal{A}_n$ remains closed and convex. However, in a more general setting and when dealing with nonconvex constraints, the projection operation may not be well-defined. Extending the present algorithm to such cases would require replacing the nonconvex constraints by suitable convex approximations.

The parameter α determines the smoothness of the projection map. Noting that

$$\|\rho\|^2_{\alpha} = \langle \rho, \rho \rangle + \alpha \langle \nabla\rho, \nabla\rho \rangle = \|\rho\|^2 + \alpha \, |\rho|^2_1 \tag{3.27}$$

we can see that the value of α determines the trade-off between minimizing the L^2 mismatch or matching the gradient values in the projection operation defined by $\Pi^{\alpha}_{\mathcal{B}}$. As shown in the next section, the choice $\alpha = \beta\tau$ has a particular significance for the proposed algorithm.

Also of significance is the case $\alpha = 0$ when $\Pi^{\alpha}_{\mathcal{B}}$ reduces to usual the L^2-projection. This is precisely the projection used in the density filtering algorithm (see equation (3.28) below), since the auxiliary functions in $\overline{\mathcal{A}}_n$ need not be (weakly) differentiable. Even though $\mathcal{A}_n$ is *not* closed with respect to the L^2-norm, we nevertheless consider this case and define $\rho_{n+1} = \Pi^0_{\tilde{\mathcal{A}}_n}(\rho^*_{n+1})$ where $\tilde{\mathcal{A}}_n = \big\{\rho \in L^{\infty}(\Omega), \rho^{\mathsf{L}}_n \le \rho \le \rho^{\mathsf{U}}_n \text{ a.e.}\big\}$. In fact, in the continuum setting, we can explicitly write

$$\rho_{n+1} = \big(\rho^*_{n+1} \wedge \rho^{\mathsf{L}}_n\big) \vee \rho^{\mathsf{U}}_n \tag{3.28}$$

Observe that the density ρ_{n+1} resulting from this projection need not lie in $H^1(\Omega)$ and so $R(\rho_{n+1}) = \frac{\beta}{2}\,|\rho_{n+1}|^2_1$ may not be defined. As such, the use of L^2 projection is inconsistent for solving the optimization problem (3.11). However, our numerical results show that it can produce noteworthy results (optimal densities that are nearly binary even for large β values). At any rate, the intention behind regularization for the topology optimization problem is controlling the complexity of final topologies. Restricting densities to $H^1(\Omega)$ is not necessary (or perhaps desirable) from a practical perspective. In fact, one can ignore the derivation and only focus on update equation (3.22) that features a particular use of filtering.

3.6 Relation to forward-backward splitting method

We next show that the proposed algorithm is related to the forward-backward splitting method (more broadly to the auxiliary problem principle [52, 51] and cost approximation [122] methods) when $\Pi_{\mathcal{A}} = \Pi_{\mathcal{A}_n}^{\beta\tau}$, i.e., $\alpha = \beta\tau$ in the definition of the projection map. This connection allows us to place the algorithm on a more solid theoretical grounds and tap into the vast literature and results on these methods and explore the use of their many variations.

To this effect, we expand the update equation (3.19), which for the projection map defined in (3.26), can be equivalently written as

$$\rho_{n+1} = \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \left\|\rho_{n+1}^* - \rho\right\|_\alpha^2 \tag{3.29}$$

Since adding or removing constant terms or multiplying by a scalar does not affect the minimizer, we have

$$\begin{aligned}
\rho_{n+1} &= \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \langle\rho,\rho\rangle - 2\left\langle\rho_{n+1}^*,\rho\right\rangle + \alpha\left\langle\nabla\rho - 2\nabla\rho_{n+1}^*, \nabla\rho\right\rangle \\
&= \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \langle\rho,\rho\rangle - 2\left[\langle\rho_n - \tau J'(\rho_n),\rho\rangle - \beta\tau\left\langle\nabla\rho_{n+1}^*,\nabla\rho\right\rangle\right] + \alpha\left\langle\nabla\rho - 2\nabla\rho_{n+1}^*,\nabla\rho\right\rangle \\
&= \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \left\|\rho - [\rho_n - \tau J'(\rho_n)]\right\|^2 + \alpha\langle\nabla\rho,\nabla\rho\rangle + 2(\beta\tau-\alpha)\left\langle\nabla\rho_{n+1}^*,\nabla\rho\right\rangle \\
&= \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \frac{1}{2\tau}\left\|\rho - [\rho_n - \tau J'(\rho_n)]\right\|^2 + \frac{\alpha}{\beta\tau}R(\rho) + \left(1-\frac{\alpha}{\beta\tau}\right)\mathrm{d}R(\rho_{n+1}^*)[\rho]
\end{aligned} \tag{3.30}$$

where $\mathrm{d}R(\rho_{n+1}^*)[\rho]$ denotes the Gateaux derivative of R at ρ_{n+1}^* in the direction of ρ. Note that in the second equality above, we have used the fact that ρ_{n+1}^* solves the variational form of (3.20) given by

$$\left\langle\rho_{n+1}^*,\rho\right\rangle + \beta\tau\left\langle\nabla\rho_{n+1}^*,\nabla\rho\right\rangle = \langle\rho_n - \tau J'(\rho_n),\rho\rangle, \quad \forall\rho\in H^1(\Omega) \tag{3.31}$$

The first term in the minimization problem (3.30) measures the L^2 distance of ρ with the gradient descent step associated with J while the last two terms give an interpolation between $R(\rho)$ and its derivative at ρ_{n+1}^* as determined by projection parameter α. For $\alpha = \beta\tau$, this reduces to

$$\rho_{n+1} = \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}} \frac{1}{2\tau}\left\|\rho - [\rho_n - \tau J'(\rho_n)]\right\|^2 + R(\rho) \tag{3.32}$$

which is precisely the iterations defined by the so-called *forward-backward splitting* procedure for minimization problem (3.11) [46]. The intuition behind (3.32) is that the next iterate ρ_{n+1} is close to the gradient descent update on J, i.e., $\rho_n - \tau J'(\rho_n)$ while minimizing

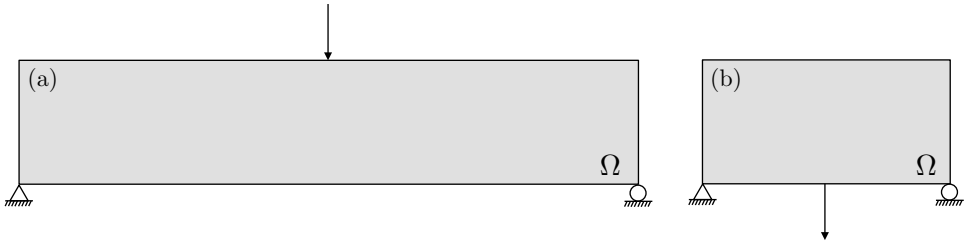


Figure 3.2: Design domain and boundary conditions for (a) the MBB beam problem (the design domain has height $h = 1$ and width $w = 6$) and (b) the bridge problem (the design domain has height $h = 1$ and width $w = 2$). In both cases, the applied load has unit magnitude.

the complexity term $R(\rho)$. The forward-backward splitting allows for separate treatment of the constituent terms of $\tilde{J}$, which is particularly useful when $R(\rho)$ is nonsmooth. Defining

$$K(\rho;\psi) = \frac{1}{2\tau}\left\|\rho - [\psi - \tau J'(\psi)]\right\|^2 + R(\rho) \tag{3.33}$$

it can be readily shown that if $\hat{\rho}$ minimizes $K(\rho;\hat{\rho})$ in $\mathcal{A}$, then $\hat{\rho}$ is also a minimizer of $\tilde{J}(\rho)$ [52]. This illustrates the fact that if the sequence ρ_n produced by iterations (3.32) converges, then the limit is the solution to the minimization problem (3.11).

It is insightful to note that the forward-back iteration is equivalent to

$$\rho_{n+1} = \underset{\rho\in\mathcal{A}_n}{\operatorname{argmin}}\; J(\rho_n) + \left\langle J'(\rho_n), \rho - \rho_n\right\rangle + \frac{1}{2\tau}\left\|\rho - \rho_n\right\|^2 + R(\rho) \tag{3.34}$$

We can see that, in the forward-backward subproblem, J is replaced by a local quadratic model whose curvature depends on $1/\tau$. The magnitude of τ affects how far ρ_{n+1} is from ρ_n. As noted before, the move limit constraint introduced in $\mathcal{A}_n$ also limits the change between ρ_n and ρ_{n+1} and so, for $\alpha = \beta\tau$ and fixed m, larger values of τ may accelerate convergence of the algorithm. Also, in light of (3.34), we can improve the performance of the algorithm by varying the step size parameter in the course of the algorithm. For example, we may choose step size τ_n in the nth iteration such that $\tau_n^{-1}I$ is a better approximation to the Hessian $J''(\rho_n)$. This issue is examined more closely in the next chapter.

3.7 Numerical investigations

In this section, we assess the performance of the proposed algorithm and present some numerical results for the compliance minimization problem. First we discuss some implementation aspects and efficiency considerations related to this algorithm. The numerical results are presented next with emphasis placed on the two extreme choices of the pro-

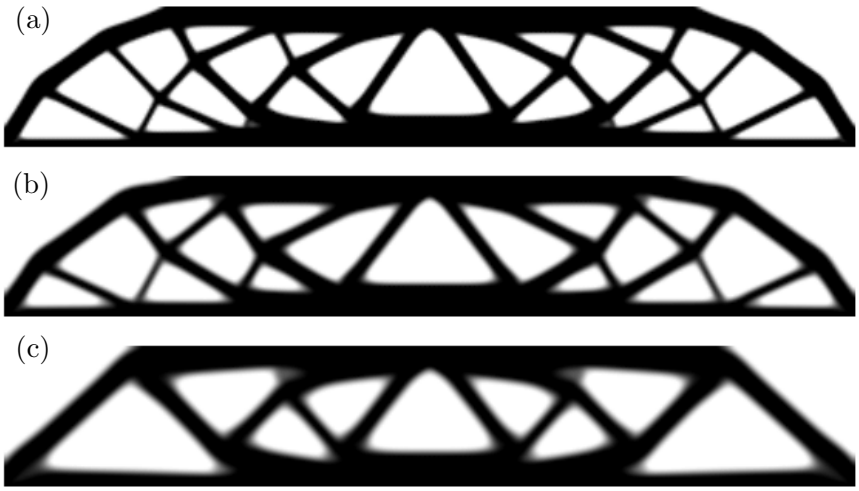


Figure 3.3: Solutions to the MBB beam problem using the forward-backward algorithm, i.e, $\alpha = \beta\tau$, and complexity parameter (a) $\beta = 0.01$ (b) $\beta = 0.03$ and (c) $\beta = 0.06$

jection parameter, namely $\alpha = 0$ (the L^2-projection) and $\alpha = \beta\tau$ (the forward-backward algorithm).

The density field here is discretized by means of finite elements which allow for the use of unstructured grids necessary for representing arbitrary design domains Ω. Given an appropriate set of basis functions $\{N_i\}_{i=1}^M$, each admissible density function is written as $\rho = \sum_{i=1}^M N_i z_i$ where $0 \leq z_i \leq 1$ for all i and so it is characterized by the vector of nodal values $\mathbf{z} = [z_i]_{i=1}^M$. The Galerkin discretization of (3.20) yields the linear system

$$(\mathbf{M} + \beta\tau\mathbf{G})\,\mathbf{z}_{n+1}^* = \mathbf{M}\mathbf{z}_n - \tau\mathbf{F}_n \tag{3.35}$$

where $[\mathbf{M}]_{ij} = \langle N_i, N_j\rangle$ and $[\mathbf{G}]_{ij} = \langle \nabla N_i, \nabla N_j\rangle$ are the standard finite element matrices and $[\mathbf{F}_n]_i = \langle J'(\rho_n), N_i\rangle$. Note that the matrix $\mathbf{M} + \beta\tau\mathbf{G}$ does not change during the course of the algorithm (unless the mesh is changed) and thus can be factored once in the beginning of the algorithm. The cost of solving (3.35) is then negligible during the subsequent iterations. We note that for large-scale problems where factorization is not feasible and iterative solvers are necessary, one could use Krylov recycling (see, for example, [72] and [121]) to take the advantage of the fact only the right-hand-side changes in the sequence of linear systems (3.35).

The discrete counterpart to the projection (3.29) is

$$\mathbf{z}_{n+1} = \operatorname*{argmin}_{\mathbf{z}_n^{\mathsf{L}} \leq \mathbf{z} \leq \mathbf{z}_n^{\mathsf{U}}} \left(\mathbf{z} - \mathbf{z}_{n+1}^*\right)^T (\mathbf{M} + \alpha\mathbf{G}) \left(\mathbf{z} - \mathbf{z}_{n+1}^*\right) \tag{3.36}$$

where $\mathbf{z}_n^{\mathsf{L}}$ and $\mathbf{z}_n^{\mathsf{U}}$ are defined in an obvious way. Note that minimization problem (3.36)

is a sparse (strictly) convex quadratic program subject to simple bound constraints and can be solved efficiently using, for example, the active set method. However, we can again exploit the fact that Hessian $\mathbf{M} + \alpha\mathbf{G}$ is fixed in the course of the optimization. Using the Cholesky decomposition $\mathbf{M} + \alpha\mathbf{G} = \mathbf{R}^T\mathbf{R}$, we can write

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}_n^{\mathsf{L}} \leq \mathbf{z} \leq \mathbf{z}_n^{\mathsf{U}}}{\operatorname{argmin}} \left\| \mathbf{R}\mathbf{z} - \mathbf{R}\mathbf{z}_{n+1}^* \right\|^2 \tag{3.37}$$

where $\|\cdot\|$ denotes the standard Euclidean norm. Therefore, upon calculation of $\mathbf{R}$ once in the beginning of the algorithm, we only need to solve a bound constrained sparse least squares problem in each iteration, a simpler problem which can be solved efficiently, for example, by algorithms proposed in [2]. In fact, with this approach, the dominant cost in each iteration of the topology optimization algorithm is computing the compliance sensitivities, i.e., vector $\mathbf{F}_n$, which requires the solution to the elasticity system (3.2). We note that this is still the case if one wishes to enforce the volume constraint explicitly (cf. (3.6)). This requires including an additional linear constraint of the form $\mathbf{v}^T\mathbf{z} \leq \overline{v}\,|\Omega|$ where $[\mathbf{v}]_i = \int_\Omega N_i \mathrm{d}\mathbf{x}$ and the above-mentioned algorithms for solving sparse quadratic programs or least squares problems are capable of handling it.

For the sake of simplicity and following the common approach, we use the same finite element mesh describing the density field to solve the state equation. The concept of generalized isoparametric finite elements [95] is fitting as the density field (and consequently $\mathbf{C}_\rho$) and the displacement field are discretized on the same mesh. Within this framework, two accuracy considerations concerning (3.20) and (3.2) should guide the appropriate choice of finite element discretization (type of basis functions and level of mesh refinement). Though, we chose to use a fixed grid for the entire course of optimization for the results in this paper, these criteria can be used to devise an adaptive finite element strategy.

We consider two benchmark compliance minimization problems, namely the MBB beam problem [119] and the bridge problem, whose domain geometry and prescribed loading and boundary conditions[4] are shown in Figure 3.2. The value of volume coefficient λ was set to $200\,|\Omega|^{-1}$ and $70\,|\Omega|^{-1}$ for these problems, respectively. For all the results, the constituent material $\mathbf{C}$ was assumed to be isotropic with unit Young's modulus and Poisson's ratio of $\nu = 0.3$ and the Ersatz stiffness was set to $\epsilon = 10^{-4}$. The SIMP penalty parameter, the move limit, and the step size were fixed at $p = 3$, $m = 0.02$ and $\tau = 0.75\lambda^{-1}$ throughout the course of optimization (i.e., no continuation was carried out) and the convergence tolerance of 10^{-6} was used. The initial guess in all cases was taken to be uniform density field with value of 0.5.

[4]Strictly speaking, point loads and supports do not satisfy the regularity assumptions stated earlier and the solution to the governing boundary value problem in the continuum setting may not exist in $H^1(\Omega)^d$. However, we use these boundary conditions so that resulting topology optimization solutions can be compared to those in the literature.

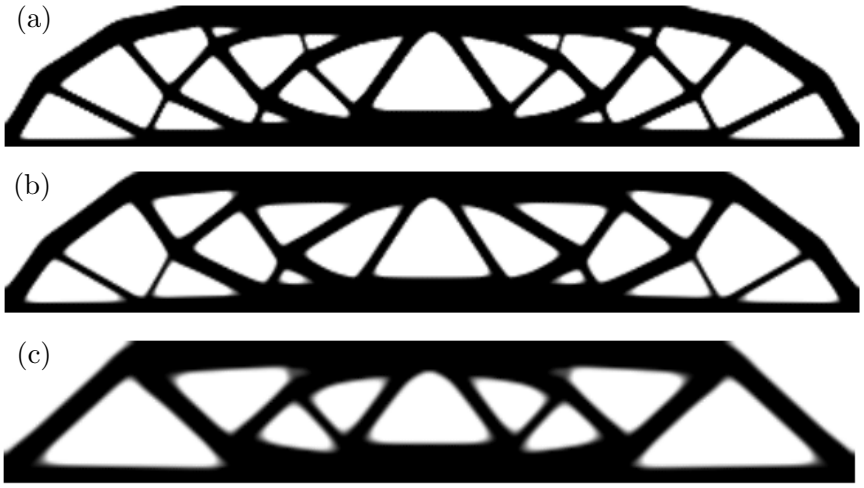


Figure 3.4: Solutions to the MBB beam problem using the L^2 projection, i.e, $\alpha = 0$, and complexity parameter (a) $\beta = 0.01$ (b) $\beta = 0.02$ and (c) $\beta = 0.05$

The first set of results explores the influence of β on the complexity of solutions to the MBB beam problem. The results were obtained using a uniform mesh of 360×60 standard bilinear quadrilateral elements for both density and displacement fields. Figure 3.3 shows the solutions obtained using the forward-backward algorithm, i.e., $\alpha = \beta\tau$, for various values of β. As expected, increase in β results in smoother final solutions with fewer but large members. The same is true for the L^2-projection (with $\alpha = 0$) as shown in Figure 3.4. Note, however, that the final densities are nearly binary with this type of projection even for large of values of β. In fact, only 10% of the nodal densities in the solution shown in 3.4(c) had values in the range $(0.05, 0.95)$ in contrast to the 25% for the solution in 3.3(c) using the forward-backward algorithm. For the sake of comparison, we have computed another measure of discreteness, proposed in [141], and the results are presented in Table 3.1.

Next we investigate the influence of mesh size on the final densities. The MBB beam problem was solved using finer grids consisting of 600×100 and 900×150 elements using the same complexity parameters as in the previous results. The final topologies were nearly identical for all the parameters (forward-backward and L^2-projection schemes) and due to the similarity with those obtained from coarser 360×60 mesh are not shown. In the case of L^2-projection, even though we evidently have convergence of the final densities under mesh refinement with respect to the L^p-norm, the regularization term $R(\rho)$ blows up and convergence does not hold in $H^1(\Omega)$. For $\beta = 0.01$, the values of the regularization term for the final solutions were 7.56, 9.89 and 9.93 for the 360×60, 600×100 and 900×150 meshes, respectively. We note, however, that the perimeter (total variation of optimal density field) remained bounded under mesh refinement, which suggests a possible

Projection type	$\alpha = \beta\tau$			$\alpha = 0$		
β	0.01	0.03	0.06	0.01	0.02	0.05
$\int_{\Gamma_N} \mathbf{u}_\rho \cdot \mathbf{t} \mathrm{d}s$	101.91	101.32	100.72	95.92	95.16	94.57
$\lambda \int_\Omega \rho \mathrm{d}\mathbf{x}$	15.43	16.22	16.83	16.40	16.73	17.53
$\frac{\beta}{2} \int_\Omega \lvert\nabla\rho\rvert^2 \mathrm{d}\mathbf{x}$	5.706	8.626	8.927	7.563	11.11	11.29
$\frac{4}{\lvert\Omega\rvert} \int_\Omega \rho(1-\rho) \mathrm{d}\mathbf{x}$	7.62%	13.7%	14.7%	3.22%	5.00%	8.65%

Table 3.1: Summary of the results for the MBB beam problem

connection with perimeter constraint problem. For the case of $\beta = 0.01$, the total variation of the final solutions were 33.3, 32.9 and 32.7 for these meshes, respectively. Similarly, for $\beta = 0.05$, the total variation of the final solutions were 20.1, 19.7, and 19.5, respectively.

We intend to investigate this observation more in our future work but for now we note that the phase field approximation of the perimeter constraint problem is given by (see, for example [32, 38])

$$\min_{\rho \in \mathcal{A}} J(\rho) + \gamma \left[\delta \left| \rho \right|_1^2 + \frac{1}{\delta} \int_\Omega W(\rho) \mathrm{d}\mathbf{x} \right] \tag{3.38}$$

where $\delta > 0$ and $W(\rho)$ is a strictly positive function that only vanishes at $\rho = 0$ and $\rho = 1$. Observe that the first term is identical to the present Tikhonov regularization term when $\gamma\delta = 2\beta$. Moreover, a binary density field (which is what our algorithm nearly produces when $\alpha = 0$) minimizes the second term in the phase field approximation. This formal argument supports our numerical observations and highlights the fact that the underlying penalization mechanism in SIMP (combined with effects of nonsmooth L^2-projection) is an effective replacement for the penalization term in the phase field approximation of the perimeter constraint problem.

The last set of numerical results, shown in Figure 3.5, is for the bridge problem. The results were obtained for $\beta = 0.02$ and $\alpha = 0$ (i.e., the L^2-projection) using a structured square mesh and an unstructured mesh consisting of linear convex polygons (see [149, 162] and chapter 5 for the finite element formulation). Both meshes are made up of 20,000 elements. The final densities, shown in 3.5(a) and (b), are nearly identical despite the difference in the choice of the spatial discretization. It is interesting to note that even though the polygonal mesh used was not symmetric about the midspan axis, the final topology is symmetric. A summary of the mesh refinement study for the bridge is given in Table 3.2. Again we can see that the total variation of final solutions remains constant under mesh refinement while the regularization term grows.

For the sake of comparison, we also solved an equivalent problem (i.e., with the same parameter values for β and τ) using the density filtering method (cf. equations (3.24) and (3.25))[5]. It can be seen from Figure 3.5(c) that the final topology is the same but the

[5]Note that the smoothness of filter $\mathcal{P}$ depends on $\beta\tau$ for both density filtering and the proposed L^2-projection

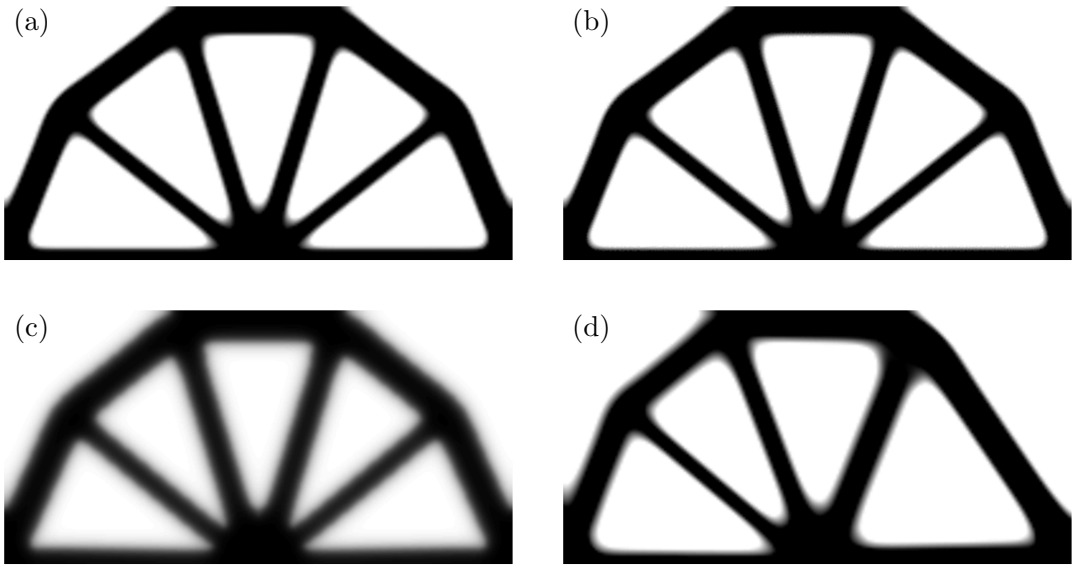


Figure 3.5: Solutions to the the bridge problem using (a) L^2-projection, isotropic regularization and structured square mesh (b) L^2-projection, isotropic regularization and unstructured polygonal mesh (c) equivalent (same β and τ as the previous two cases) density filtering (d) L^2-projection scheme with anisotropic regularization term

Grid size	100×50	150×75	200×100	300×150
Number of iterations	91	105	94	93
$\int_{\Gamma_N} \mathbf{u}_\rho \cdot \mathbf{t} \mathrm{d}s$	20.35	20.84	21.22	21.81
$\lambda \int_\Omega \rho \mathrm{d}\mathbf{x}$	25.06	24.43	24.15	23.94
$\frac{\beta}{2} \int_\Omega \lvert\nabla\rho\rvert^2 \mathrm{d}\mathbf{x}$	3.340	3.828	4.214	4.544
$\int_\Omega \lvert\nabla\rho\rvert \mathrm{d}\mathbf{x}$	11.53	11.55	11.54	11.51

Table 3.2: Summary of mesh refinement study for the bridge problem with $\beta = 0.02$ and L^2-projection

Figure 3.6: The original image (left) with noise added (middle) and its reconstruction (left) using total variation minimization algorithm of the form (3.40). Images courtesy of [43]

density field is heavily smeared. In fact, only less than 48% of the nodal densities are in the range $[0, 0.05] \cup [0.95, 1]$ in contrast to 91% for the solution using the L^2-projection scheme. In terms of the discreteness measure of [141], the density filtering solution has a value of 33.1% while it is 7.04% for the L^2-projection result.

Lastly, we solved the bridge problem for the more general regularization term (3.10) with

$$\boldsymbol{\kappa}(\mathbf{x}) = \begin{pmatrix} 0.05 & -0.03 \\ -0.03 & 0.05 \end{pmatrix} \tag{3.39}$$

taken to be constant over the entire domain. The eigenvectors of $\boldsymbol{\kappa}$, namely $\mathbf{v}_1 = (1,1)^T/\sqrt{2}$ and $\mathbf{v}_2 = (1,-1)^T/\sqrt{2}$, are rotated 45° from the horizontal axis and the corresponding eigenvalues are $\beta_1 = 0.02$ and $\beta_2 = 0.08$, respectively. We can see from the solution in Figure 3.5(d) that the introduced anisotropy of the regularization term breaks the symmetry. The diagonal members on the right half, which are nearly perpendicular to $\mathbf{v}_2$, are penalized more and are thus collapsed into one member. This example illustrate the potentials of the proposed regularization scheme for control of local orientation and feature size.

3.8 Extension to nonsmooth regularizers

The use of operator splitting methods (such as the forward-backward algorithm) seems promising for nonsmooth regularizers for the topology optimization. In particular, it would be interesting to explore such a decoupling approach for the perimeter constrained problem where the regularization term $R(\rho)$ is defined to be the total variation of the density field $\int_\Omega |\nabla\rho| \, \mathrm{d}\mathbf{x}$. In this case, the forward-backward splitting leads to simpler subproblems of

scheme. By contrast, the introduced complexity in the forward-backward algorithm only depends on β as evident from (3.34).

Figure 3.7: Solution to the MBB beam problem using total variation regularization and the forward-backward splitting algorithm

the form

$$\min_{\rho \in BV(\Omega;[0,1])} \|\rho - g_n\|^2 + k \int_\Omega |\nabla \rho| \, \mathrm{d}\mathbf{x} \tag{3.40}$$

for which efficient algorithms can be found in the image processing literature (e.g., [43, 33]; also see Figure 3.6 for an example of an image denoising problem). Owing to the simple structure of (3.40), there is no need to approximate total variation by a differentiable functional and therefore the effects of this regularizer can be captured with high degree of fidelity.

As a preliminary result, we solved the MBB beam problem using a piecewise constant discretization of the density field on a grid of 420×70 elements. The forward-backward subproblem was solved using the FISTA algorithm [17]. It can be seen from the result shown in Figure 3.7 that the final solution is almost completely binary (over 97% of element densities are zero or one) which is expected since the total variation does not penalize discontinuities.

We end with a general remark on the justification of restriction formulations for topology optimization such as the one presented here. Though they seemingly involve an arbitrary modification of the original problem (e.g., requiring the density fields to be uniformly smooth in the filtering method or belonging to a bounded subset of H^1 in the present work), such restrictions are more than a theoretical tool since they can be used to enforce manufacturing constraints on the admissible shapes that can be built for engineering applications or, in the case of inverse problems (e.g. obstacle identification), introduce *a priori* knowledge about the regularity of the unknown geometry. The ultimate test of the resulting algorithms, aside from usual criteria of robustness, feasibility and ease of implementation and computational cost, currently rests on "qualitative" inspection of the final topologies.

Chapter 4

A Closer Look at the Splitting Algorithm

In this chapter, we continue to explore the use of splitting algorithms for solving regularized topology optimization problems. The context is the classical structural design problems (minimum compliance and complaint mechanism design), parameterized by means of density functions, whose ill-posedness is addressed by introducing a Tikhonov regularization term. The proposed forward-backward splitting algorithm treats the constituent terms of the cost functional separately which allows for employing suitable approximations of the structural objective. We will show that one such approximation, inspired by the optimality criteria (OC) algorithm and reciprocal expansions, improves convergence and leads to an update scheme that resembles the well-known heuristic *sensitivity filtering* method.

As previously discussed, the density filtering method, implicitly enforces a prescribed degree of smoothness on all the admissible density fields that define the topology [31]. This method and its variations are consistent in their use of sensitivity information in the optimization algorithm since the sensitivity of the objective and constraint functions are computed with respect to the associated auxiliary fields whose filtering defines the densities. By contrast, the sensitivity filtering method [142], which precedes the density filters and is typically described at the discrete level, performs the smoothening operation directly on the sensitivity field after a heuristic scaling step. The filtered sensitivities then enter the update scheme that evolves the design despite the fact they do not correspond to the cost function of the optimization problem. While the sensitivity filtering has proven effective in practice for certain class of problems (for compliance minimization, it enjoys faster convergence than the density filter counterpart), a proper justification has remained elusive. As pointed out by Sigmund [141], it is generally believed that "the filtered sensitivities correspond to the sensitivities of a smoothed version of the original objective function" even though "it is probably impossible to figure out what objective function is actually being minimized." This view is confirmed in this chapter, as we will show that an algorithm with calculations similar to what is done in the sensitivity filtering can be derived in a *consistent* manner from a proper regularization of the objective.

It was shown in chapter 3 that the derived update expression for the forward-backward algorithm naturally contains a particular use of Helmholtz filtering (cf. eq. (3.22)), where in contrast to density and sensitivity filtering methods, the filtered quantity is the gradient

descent step associated with the original structural objective. The key observation made here is that if the gradient descent step in this algorithm is replaced by the OC update, then the interim density has a similar form to that of the sensitivity filter and in fact produces similar results (cf. Figure 4.2). To make such a leap rigorous, we essentially embed the same reciprocal approximation of compliance that is at the heart of the OC scheme in the forward-backward algorithm. This leads to a variation of the forward-backward splitting algorithm that is consistent, demonstrably convergent and computationally tractable.

Within the more general framework presented here, we will examine the choice of move limits and the step size parameter more closely and discuss strategies that can improve the convergence of the algorithm while maintaining the quality of final solutions. We also discuss a two-metric variant of the splitting algorithm that removes the computational overhead associated with the bound constraints on the density field without compromising convergence and quality of optimal solutions. In particular, we present and investigate scheme based on the two-metric projection method of [25, 79] that allows for the use of a more convenient metric for the projection step enforcing these bound constraints. This algorithm requires a simple and computationally inexpensive modification to the splitting scheme but features a min/max-type projection operation similar to OC-based filtering methods. We will see from the numerical examples that the two-metric variation retains the convergence characteristics of the forward-backward algorithm for various choices of algorithmic parameters. The details of the two types of algorithms are described for the finite-dimensional optimization problem obtained from the usual finite element approximation procedure, which we prove is convergent for Tikhonov-regularized compliance minimization problem.

As in chapter 3, the inner product and norm associated with $L^2(\Omega)$ are denoted as $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$, respectively. Also the inner product, norm and semi-norm associated with $H^k(\Omega)$ are denoted by $\langle\cdot,\cdot\rangle_k$, $\|\cdot\|_k$ and $|\cdot|_k$, respectively. Given a bounded and positive-definite linear operator $\mathcal{B}$, we write $\langle u, v\rangle_{\mathcal{B}} \equiv \langle u, \mathcal{B}v\rangle$ and the associated norm by $\|u\|_{\mathcal{B}} \equiv \langle u, u\rangle_{\mathcal{B}}^{1/2}$. Similarly, the standard Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$ is denoted by $\|\mathbf{v}\|$ and given a positive-definite matrix $\mathbf{B}$, we define $\|\mathbf{v}\|_{\mathbf{B}} = \left(\mathbf{v}^T\mathbf{B}\mathbf{v}\right)^{1/2}$. The ith components of vector $\mathbf{v}$ and the (i,j)-th entry of matrix $\mathbf{B}$ are written as $[\mathbf{v}]_i$ and the $[\mathbf{B}]_{ij}$, respectively.

4.1 Problem statement

As in chapter 3, we consider the Tikhonov regularized compliance minimization problem given by

$$\min_{\rho\in\mathcal{A}} \ \tilde{J}(\rho) = J(\rho) + \frac{\beta}{2}\,|\rho|_1^2 \tag{4.1}$$

where $\beta > 0$ is a positive constant and $J(\rho)$ is defined by (3.1) with the minor difference that we use the classical SIMP model for stiffness,

$$\mathbf{C}_\rho = \rho^p \mathbf{C} \tag{4.2}$$

and accordingly define the space of admissible densities to be

$$\mathcal{A} = \left\{ \rho \in H^1(\Omega) : \delta_\rho \leq \rho \leq 1 \text{ a.e.} \right\} \tag{4.3}$$

where $0 < \delta_\rho \ll 1$ is a small positive constant. The reason is that later in section 4.3, we will consider Taylor expansions of the objective function in $1/\rho$ and the positive lower bound on the density field simplifies the notation. Notice that the governing boundary value problem is again well-posed since the energy bilinear form is continuous and coercive with uniform constants for all $\rho \in \mathcal{A}$.

For brevity and emphasizing the quadratic form of the Tikhonov regularization, in the next two sections, we write the regularizer generically as

$$\frac{1}{2} \langle \rho, \mathcal{R}\rho \rangle \tag{4.4}$$

where $\mathcal{R}$ is a linear, self-adjoint and positive semi-definite operator on $\mathcal{A}$. Recall that under an additional assumption of $\partial\rho/\partial\mathbf{n} = 0$ on $\partial\Omega$ and $\rho \in H^2(\Omega)$, the Tikhonov regularization term can be written as $\frac{1}{2}\langle \rho, -\beta\Delta\rho \rangle$. Similarly, the more general regularization term $\frac{1}{2}\langle \nabla\rho, \boldsymbol{\kappa}\nabla\rho \rangle$ in which $\boldsymbol{\kappa}(\mathbf{x})$ is a bounded and positive-definite matrix can be written as $\frac{1}{2}\langle \rho, -\nabla\cdot(\boldsymbol{\kappa}\nabla\rho) \rangle$. However, the additional assumption on densities are in fact not required.

Lastly, we recall that the gradient of compliance (with respect to variations of density in the L^2-metric) is given by [24]

$$J'(\rho) = -E(\rho) + \lambda \tag{4.5}$$

where $E(\rho) = p\rho^{p-1}\mathbf{C}\boldsymbol{\epsilon}(\mathbf{u}_\rho) : \boldsymbol{\epsilon}(\mathbf{u}_\rho)$ is a strain energy density field. Note that $E(\rho)$ is non-negative for any admissible density and this is related to the monotonicity of the self-adjoint compliance problem: given densities ρ_1 and ρ_2 such that $\rho_1 \leq \rho_2$ a.e., one can show $\ell(\mathbf{u}_{\rho_1}) \geq \ell(\mathbf{u}_{\rho_2})$. This property is the main reason why we restrict our attention to compliance minimization (though in section 4.9, we will provide an example of compliant mechanism design which is not self-adjoint). Observe that $\hat{\rho}$ is a stationary point of J if

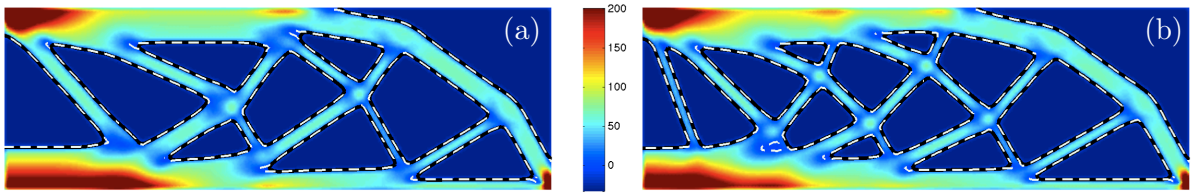


Figure 4.1: Plot of $E(\rho) - \lambda$ for two solutions to the MBB beam problem with $\beta = 0.01$: (a) corresponds to solution shown in Figure 4.5(b) and (b) corresponds to the solution shown in Figure 4.5(c). The black line is the contour line for $\rho = 1/2$ and the dashed white line is the contour line where $E(\rho) = \lambda$. Note that only half the design domain is shown and the range of the color-bar is limited to $[-\lambda, 6\lambda]$ for better visualization.

$$\begin{cases} E(\hat{\rho})(\mathbf{x}) < \lambda, & \text{if } \hat{\rho}(\mathbf{x}) = \delta_\rho \\ E(\hat{\rho})(\mathbf{x}) = \lambda, & \text{if } \delta_\rho < \hat{\rho}(\mathbf{x}) < 1 \\ E(\hat{\rho})(\mathbf{x}) > \lambda, & \text{if } \hat{\rho}(\mathbf{x}) = 1 \end{cases} \tag{4.6}$$

Thus, in regions where $E(\hat{\rho})$ exceeds the penalty parameter λ (regions that experience "large" deformation), density is at its maximum. Similarly, below this cutoff value the density is equal to the lower bound δ_ρ. Everywhere else, i.e., in the regions of intermediate density, the strain energy density is equal to the penalty parameter λ.

Figure 4.1 shows the distribution of $E(\rho) - \lambda$ for solutions to (4.1) obtained using the proposed algorithm (cf. section 4.9 and Figures 4.5(b) and (c)). Superimposed are the contour lines associated with $\rho = 1/2$ (plotted in black) representing the boundary of the optimal shape and $E(\rho) = \lambda$ (plotted in dashed white). The fact that these lines are nearly coincident shows that the solutions to the regularized problem, at least for sufficiently small regularization parameter β, are close to ideal in the sense that they nearly satisfy the stationarity condition for the structural objective J.

4.2 General splitting algorithm

In this section, we discuss a generalization of the forward-backward splitting algorithm presented in chapter 3 for solving the regularized compliance minimization problem. The key idea behind this and other similar decomposition methods [52, 46, 122] is the separate treatment of constituent terms of the cost function.

A general algorithm for finding a minimizer of $\tilde{J}(\rho)$ consists of subproblems of the form:

$$\rho_{n+1} = \underset{\rho \in \mathcal{A}_n}{\operatorname{argmin}} \; J(\rho_n) + \langle \rho - \rho_n, J'(\rho_n) \rangle + \frac{1}{2\tau_n} \|\rho - \rho_n\|^2_{\mathcal{H}_n} + \frac{1}{2} \langle \rho, \mathcal{R}\rho \rangle \tag{4.7}$$

where $\mathcal{H}_n$ is a bounded and positive-definite linear operator. Compared to (4.1), we can see that while the regularization term has remained intact, J is replaced by a local quadratic model around ρ_n in which $\mathcal{H}_n$ may be viewed as an approximation to the Hessian of J evaluated at ρ_n. Note that constant terms such as $J(\rho_n)$ and $\langle \rho_n, J'(\rho_n) \rangle$ do not affect the optimization but are provided to emphasize the expansion of J. Moreover, $\tau_n > 0$ is a *step size* parameter that determines the curvature of this approximation. For sufficiently small τ_n (large curvature), the approximation is conservative in that it majorizes (lies above) J, which is crucial in guaranteeing descent in each iteration and overall convergence of the algorithm.

We have included another limiting measure in (4.7), a minor departure from the above-mentioned references, by replacing the constraint set $\mathcal{A}$ by a subset $\mathcal{A}_n$ in order to limit the point-wise change in the density to a specified *move limit* m_n. More specifically, as in chapter 3, we have defined

$$\mathcal{A}_n = \{\rho \in \mathcal{A} : |\rho - \rho_n| \leq m_n \text{ a.e.}\} = \left\{\rho \in H^1(\Omega) : \rho_n^{\mathsf{L}} \leq \rho \leq \rho_n^{\mathsf{U}} \text{ a.e.}\right\} \tag{4.8}$$

where in the latter expression

$$\rho_n^{\mathsf{L}} = \delta_\rho \wedge (\rho_n - m_n), \quad \rho_n^{\mathsf{U}} = 1 \vee (\rho_n + m_n) \tag{4.9}$$

The presence of move limits (akin to a trust region strategy) is common in topology optimization literature as a means to stabilize the topology optimization algorithm, especially in the early iterations to prevent members from forming too prematurely. As we will show with an example, this is only important when a smaller regularization parameter is used and the final topology is complex. Near the optimal solution, the move limit strategy is typically inoperative. Of course, by setting $m_n \equiv 1$, we can get $\mathcal{A} = \mathcal{A}_n$ and recover the usual form of (4.7).

Ignoring the constant terms and with simple rearrangement, we can show that (4.7) is equivalent to

$$\rho_{n+1} = \underset{\rho \in \mathcal{A}_n}{\operatorname{argmin}} \ \left\|\rho - \rho_{n+1}^*\right\|^2_{(\mathcal{H}_n + \tau_n \mathcal{R})} \tag{4.10}$$

where the *interim density* ρ_{n+1}^* is given by

$$\rho_{n+1}^* = (\mathcal{H}_n + \tau_n \mathcal{R})^{-1} \left[\mathcal{H}_n \rho_n - \tau_n J'(\rho_n)\right] \tag{4.11}$$

Alternatively, the interim density can be written as a Newton-type update where the gradient of $\tilde{J}$ is scaled by the inverse of its approximate Hessian, namely

$$\rho_{n+1}^* = \rho_n - \tau_n (\mathcal{H}_n + \tau_n \mathcal{R})^{-1} \left[J'(\rho_n) + \mathcal{R}\rho_n\right] \tag{4.12}$$

Returning to (4.10), we can see that next density ρ_{n+1} is defined as the *projection* of the interim density, with respect to the norm defined by $\mathcal{H}_n + \tau_n \mathcal{R}$, onto the constraint space $\mathcal{A}_n$. From the assumptions on properties of $\mathcal{H}_n$ and the Tikhonov regularization operator $\mathcal{R}$ and the fact that $\mathcal{A}_n$ is a closed convex subset of $H^1(\Omega)$, it follows that the projection is well-defined and there is a unique update ρ_{n+1}.

By setting $\mathcal{R} = -\beta\Delta$, which corresponds to the regularization term of (4.1) and choosing $\mathcal{H}_n$ to be the identity map $\mathcal{I}$, we recover the forward-backward algorithm investigated in the previous chapter. In this case, the interim update satisfies the Helmholtz equation

$$(\mathcal{I} - \tau_n \beta \Delta)\, \rho^*_{n+1} = \rho_n - \tau_n J'(\rho_n) \tag{4.13}$$

Note that the right hand side is the usual gradient descent step (with step size τ_n) associated with J (the forward step) and the interim density is obtained from application of the inverse of the Helmholtz operator (the backward step), which can be viewed as the filtering of right-hand-side with the Gaussian Green's function of the Helmholtz equation[1]. As mentioned in the introduction, this appearance of filtering is fundamentally different from density and sensitivity filtering methods. Moreover, the projection operation in this case is with respect to a scaled Sobolev metric, namely

$$\rho_{n+1} = \underset{\rho \in \mathcal{A}_n}{\operatorname{argmin}} \ \left\|\rho - \rho^*_{n+1}\right\|^2 + \beta\tau_n \left|\rho - \rho^*_{n+1}\right|_1^2 \tag{4.14}$$

which numerically requires the solution to a box-constrained convex quadratic program. In chapter 3, we also explored an "inconsistent" variation of this algorithm where we neglected the second term in (4.14) and essentially used the L^2-metric for the projection step. Due to the particular geometry of the box constraints in $\mathcal{A}_n$, the L^2-projection has the explicit solution given by

$$\rho_{n+1} = \left(\rho^*_{n+1} \wedge \rho_n^{\mathsf{L}}\right) \vee \rho_n^{\mathsf{U}} \tag{4.15}$$

The appeal of this min/max type operation is that it is trivial from the computational point of view. Moreover, it coincides with the last step in the OC update scheme [24]. However, this is an inconsistent step for Tikhonov regularized problem since ρ_{n+1} need not lie in $H^1(\Omega)$. In fact, strictly speaking, (4.15) is valid only if $\mathcal{A}_n$ is enlarged from functions in $H^1(\Omega)$ to all functions in $L^2(\Omega)$ bounded below by ρ_n^{L} and above by ρ_n^{U}. In spite of this inconsistency, the algorithm composed of (4.13) and (4.15) was convergent and numerically shown to produce noteworthy solutions with minimal intermediate densities. This merits a separate investigation since as suggested in chapter 3, this algorithm may in fact solve a smoothed version of the perimeter constraint problem where the regularization term is

[1]The designations "forward" and "backward" step come from the fact that (4.13) can be written as $\rho^*_{n+1} = (\mathcal{I} + \tau_n \mathcal{R})^{-1} (\mathcal{I} - \tau_n J')\, \rho_n$. Similarly, (4.11) has equivalent expression $\rho^*_{n+1} = \left(\mathcal{I} + \tau_n \mathcal{H}_n^{-1} \mathcal{R}\right)^{-1} \left(\mathcal{I} - \tau_n \mathcal{H}_n^{-1} J'\right) \rho_n$.

the total variation of the density field. We will return to the use of L^2-projection later in section 6.4 but *this time in a consistent manner with the aid of the two-metric projection approach of [25, 79].*

4.3 Optimality criteria and sensitivity filtering

In structural optimization, the optimality criteria (OC) method is preferred to the gradient descent algorithm since it typically enjoys faster convergence (see [13] on the relationship between the two methods). Our interest here in the OC method is that the density and sensitivity filtering methods are typically implemented in the OC framework. Moreover, as we shall see, this examination will lead to the choice of $\mathcal{H}_n$ in the algorithm (4.7).

The interim density in the OC method for the compliance minimization problem (in the absence of regularization) is obtained from the fixed point iteration

$$\rho_{n+1}^* = \rho_n \left[\frac{E(\rho_n)}{\lambda}\right]^{1/2} \equiv \rho_n \left[e_\lambda(\rho_n)\right]^{1/2} \tag{4.16}$$

Note that the strain energy density $E(\rho_n)$ and subsequently its normalization $e_\lambda(\rho_n)$ are non-negative for any admissible density ρ_n and therefore ρ_{n+1}^* is well-defined. Recalling the necessary condition of optimality for an optimal density $\hat{\rho}$ stated in (4.6), it is evident that such $\hat{\rho}$ is a fixed point of the OC iteration. Intuitively, the current density ρ_n is increased (decreased) in regions where $E(\rho_n)$ is greater (less) than the penalty parameter λ by a factor of $[e_\lambda(\rho_n)]^{1/2}$. The next density ρ_{n+1} in the OC is given by (4.15).

It is more useful here to adopt an alternative view of the OC scheme, namely that the OC update can be seen as the solution to an approximate subproblem where compliance is replaced by a Taylor expansion in the intermediate field ρ^{-1} [82]. The intuition behind such expansion is that locally compliance is inversely proportional to density. In particular, ρ_{n+1}^* can be shown to be the stationary point of the "reciprocal approximation" around ρ_n defined by

$$J_{\mathsf{rec}}(\rho;\rho_n) \equiv \ell(\mathbf{u}_{\rho_n}) + \left\langle \frac{\rho_n}{\rho}\left(\rho-\rho_n\right), -E(\rho_n)\right\rangle + \lambda\int_\Omega \rho \mathrm{d}\mathbf{x} \tag{4.17}$$

Note that the expansion in the inverse of density is carried out only for the compliance term, and the volume term, which is already linear, is not altered. The expression for $J_{\mathsf{rec}}(\rho;\rho_n)$ can be alternatively written as

$$J_{\mathsf{rec}}(\rho;\rho_n) = J(\rho_n) + \langle \rho-\rho_n, J'(\rho_n)\rangle + \frac{1}{2}\left\langle \rho-\rho_n, \frac{2E(\rho_n)}{\rho}\left(\rho-\rho_n\right)\right\rangle \tag{4.18}$$

which highlights the fact that the (nonlinear) curvature term in (4.18) makes it a more

accurate approximation of compliance compared to the linear expansion. With regard to the OC update, one can show that the interim update satisfies $J'_{\mathsf{rec}}(\rho^*_{n+1};\rho_n)=0$, and its L^2-projection is indeed the minimizer of $J_{\mathsf{rec}}(\rho;\rho_n)$ over $\mathcal{A}_n$ (again enlarged to L^2).

We now turn to the *sensitivity filtering* method, which is described with the OC algorithm. Let $\mathcal{P}$ denote a linear filtering map, for example, the Helmholtz filter $\mathcal{P}=(\mathcal{I}-r^2\Delta)^{-1}$ discussed before or the convolution filter of radius radius r [31, 29]

$$\mathcal{P}(\psi)(\mathbf{x})\equiv\int_\Omega F_r(\mathbf{x}-\mathbf{y})\psi(\mathbf{y})\mathrm{d}\mathbf{y} \tag{4.19}$$

where the kernel is the linear hat function $F_r(\mathbf{x})=\max\left(1-|\mathbf{x}|/r,0\right)$. The main idea in the sensitivity filtering method is that $e_\lambda(\rho_n)$ is heuristically replaced by the following smoothed version[2]

$$\tilde{e}_\lambda(\rho_n)\equiv\frac{1}{\rho_n}\mathcal{P}\left[\rho_n e_\lambda(\rho_n)\right] \tag{4.20}$$

before entering the OC update. The interim density update is thus given by

$$\rho^*_{n+1}=\rho_n\left[\tilde{e}_\lambda(\rho_n)\right]^{1/2}=\rho_n\left\{\frac{\mathcal{P}\left[\rho_n e_\lambda(\rho_n)\right]}{\rho_n}\right\}^{1/2}=\rho_n^{1/2}\mathcal{P}\left[\rho_n e_\lambda(\rho_n)\right]^{1/2} \tag{4.21}$$

A key observation made here is that if we replace the gradient decent step in forward-backward algorithm (cf. (4.13)) with the OC step, we obtain a similar update scheme to that of the sensitivity filtering method. More specifically, note that (4.13) can be written as $\rho^*_{n+1}=\mathcal{P}\left[\rho_n-\tau_n J'(\rho_n)\right]$. Substituting the term in the bracket with $\rho_n\left[e(\rho_n)\right]^{1/2}$ gives

$$\rho^*_{n+1}=\mathcal{P}\left\{\rho_n\left[e_\lambda(\rho_n)\right]^{1/2}\right\} \tag{4.22}$$

which resembles (4.21). In fact, as illustrated in Figure 4.2, the two expressions produce very similar final results (in particular, observe the similarity between the patches of intermediate density in the corners that is characteristic of the sensitivity filtering method). Of course, the leap from the forward-backward algorithm to (4.22), just like the sensitivity filtering method, lacks mathematical justification. However, we will expand upon this observation and next derive the algorithm similar to this empirical modification of the forward-backward algorithm in a consistent manner.

4.4 Embedding the reciprocal approximation

Recalling the role of the reciprocal approximation of compliance in the OC method, the key idea is to embed such an approximation in the general subproblem of (4.7). We do so

[2]Notice that the filtering map is applied to the scaling of $e_\lambda(\rho_n)$ by the density field itself, which is not easy to explain/justify.

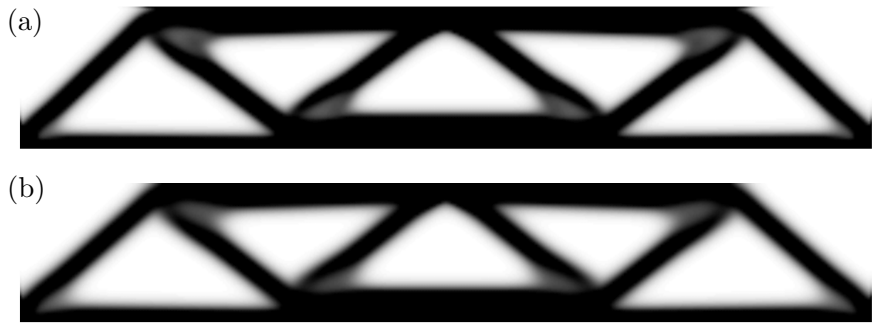


Figure 4.2: (a) The solution to the MBB beam problem using the sensitivity filtering method (consisting of (4.21) and (4.15)) (b) The solution using the update steps (4.22) and (4.15). In both cases, $\mathcal{P}$ was taken to be the Helmholtz filter and the move limit was set to $m_n = 0.25$

by choosing $\mathcal{H}_n$ to be the Hessian of $J_{\mathsf{rec}}(\rho;\rho_n)$ evaluated at ρ_n, namely[3]

$$\mathcal{H}_n = J''_{\mathsf{rec}}(\rho_n;\rho_n) = \frac{2E(\rho_n)}{\rho_n}\mathcal{I} \tag{4.23}$$

As noted earlier, $E(\rho)$ is a non-negative function for any admissible ρ but may vanish in some subset of Ω. This means that $\mathcal{H}_n$ is only positive semi-definite and does not satisfy the definiteness requirement for use in (4.7). We can remedy this by replacing $E(\rho_n)$ in (4.23) with $E(\rho_n) \wedge \delta_E$ where $0 < \delta_E \ll \lambda$ is a prescribed constant. However, in most compliance problems (e.g., the benchmark problem considered later in section 4.9) the strain energy field is strictly positive for all admissible densities. In fact, the regions with zero strain energy density do not experience any deformation and in light of the conditions of optimality (4.6) should be assigned the minimum density. Therefore, to simplify the matters, *we assume in the remainder of this section that the loading and support conditions defined on* Ω *are such that* $E(\rho) \geq \delta_E$ *almost everywhere* for all $\rho \in L^\infty(\Omega;[\delta_\rho,1])$.

Comparing the quadratic approximation of J with this choice of $\mathcal{H}_n$ and the reciprocal approximation itself (cf. (4.18)), we see that the difference is in their curvature terms (the linear terms of course match). The curvature of the quadratic model depends on and can be controlled by τ_n while the nonlinear curvature in J_{rec} is a function of ρ.

Substituting (4.23) into (4.11), the expression for the interim density becomes

$$\left[\frac{2E(\rho_n)}{\rho_n}\mathcal{I} + \tau_n\mathcal{R}\right]\rho^*_{n+1} = 2E(\rho_n) + \tau_n\left[E(\rho_n) - \lambda\right] = (2+\tau_n)\,E(\rho_n) - \tau_n\lambda \tag{4.24}$$

[3]We note that the use of quadratic approximations of the reciprocal approximation has also been pursued in [83, 84].

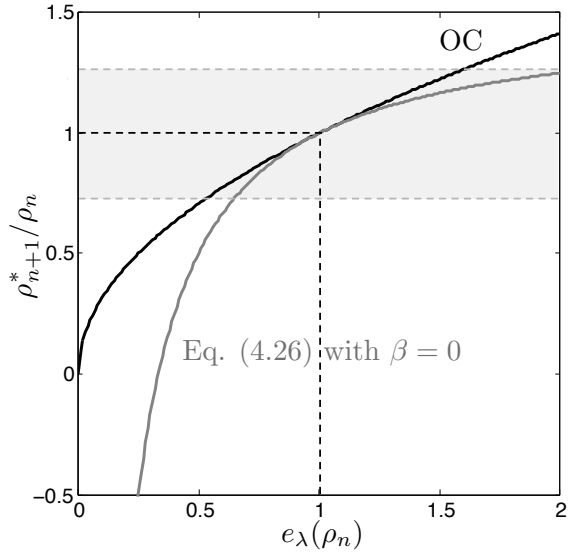


Figure 4.3: Comparison between scaling terms appearing in the OC update and right hand side of (4.26). The OC is more aggressive in regions $e_\lambda(\rho_n) > 1$ and less aggressive when $e_\lambda(\rho_n) < 1$.

Multiplying by $\rho_n / [2E(\rho_n)]$ and simplifying yields

$$\left[\mathcal{I} + \frac{\rho_n}{2E(\rho_n)}\tau_n\mathcal{R}\right]\rho^*_{n+1} = \rho_n\left[\left(1 + \frac{\tau_n}{2}\right) - \frac{\tau_n}{2e_\lambda(\rho_n)}\right] \tag{4.25}$$

To better understand the characteristics of this update, let us specialize to the case of Tikhonov regularization and set $\tau_n = 1$ (so that the quadratic model and the reciprocal approximation have the same curvature at ρ_n). This gives

$$\left[\mathcal{I} - \frac{\rho_n}{2E(\rho_n)}\beta\Delta\right]\rho^*_{n+1} = \rho_n\left[\frac{3}{2} - \frac{1}{2e_\lambda(\rho_n)}\right] \tag{4.26}$$

First note that in the absence of regularization (i.e., $\beta = 0$), the update relation has the same fixed-point iteration form as the OC update with the ratio $e_\lambda(\rho_n)$ determining the scaling of ρ_n. The scaling field here is $3/2 - 1/[2e_\lambda(\rho_n)]$ whereas in the OC method it is given by $[e_\lambda(\rho_n)]^{1/2}$. As shown in Figure 4.3, the scaling fields and their derivatives coincide in the regions where $e_\lambda(\rho_n) = 1$, which means that locally the two are similar. The reduction in density is more aggressive with this scaling when $e_\lambda(\rho_n) < 1$ whereas the OC update leads to larger increase for $e_\lambda(\rho_n) > 1$.

As with the forward-backward algorithm (cf. equation (4.13)), the presence of regularization again leads to the appearance of Helmholtz filtering (the inverse of left-hand-side operator) but with two notable differences. First, the right-hand-side term now is an OC-like scaling of density instead of the gradient descent step (the same is true in (4.25) for

an arbitrary step size τ_n). Furthermore, the filtering is not uniform across the domain and its degree of smoothening is scaled by $\rho_n / [2E(\rho_n)]$. The important result here is that, by embedding the reciprocal approximation of compliance in our quadratic model, we are able to obtain a relation for the ρ^*_{n+1} that features an OC-like right-hand-side and its filtering, very much similar in form to the (heuristically) fabricated update scheme of (4.22) that was compared to the sensitivity filtering.

Another key difference between the forward-backward algorithm and the OC-based filtering methods is that the projection of ρ^*_{n+1} defining the next iterate ρ_{n+1} in the forward-backward scheme is with respect to the metric induced by $\mathcal{H}_n + \tau_n \mathcal{R}$ in contrast to the L^2-projection given by (4.15). As discussed before L^2-projection is well-suited for the geometry of the constraint set $\mathcal{A}_n$ due to decomposition of box constraints. It may be tempting to inconsistently use the interim density (4.25) with the L^2-projection but this is not necessarily guaranteed to decrease the cost function[4]. Arbitrary projections of unconstrained Newton steps is not mathematically warranted.

In section 4.8, we explore a variant of the splitting algorithm that is related to the two-metric projection method of [25, 79], and allows for the use of a more convenient metric for the projection step. This can be done provided that the operator whose associated norm defines the gradient[5] is modified appropriately in the regions where the constraints are active. More specifically, in the interim update step (cf. (4.12)), $\mathcal{H}_n + \tau_n \mathcal{R}$ is modified to produce a linear operator $\mathcal{D}_n$ with a particular structure that eliminates the coupling between regions of active and free constraints. The projection of the interim density given by

$$\rho^*_{n+1} = \rho_n - \tau_n \mathcal{D}_n^{-1} \left[J'(\rho_n) + \mathcal{R}\rho_n \right] \tag{4.27}$$

with respect to the L^2-norm is then guaranteed to decrease the cost function[6]. Note that when there are no active constraints (e.g., in the beginning of the algorithm when the density field takes mostly intermediate values), $\mathcal{D}_n = \mathcal{H}_n + \tau_n \mathcal{R}$ and (4.25) holds for the interim update and its L^2-projection produces the next iterate. In general, (4.25) holds locally for the regions where the box constraints are not active (i.e., regions of intermediate density) and so the analogy to the sensitivity filtering method holds in such regions.

To avoid some technical nuisances (that the L^2-projection on $H^1(\Omega)$ is not well-defined) and avoid the cumbersome notation required to precisely define $\mathcal{D}_n$ in the continuum setting (that may obscure the simple procedure for its construction), we defer the details to

[4]Numerically one would observe that such an inconsistent algorithm excessively removes material and leads to final solutions with low volume fraction

[5]Recall that $\mathcal{B}^{-1} f'(\rho)$ is the gradient of functional f with respect to the metric induced by $\mathcal{B}$. As such, Newton's method and its variations (such as the present framework) can be thought of as gradient descent algorithms with respect to a variable metric defined by the (approximate) Hessian.

[6]There is the technical issue that L^2-projection on a subset of $H^1(\Omega)$ is not well-defined, which is why we defer the exact outline of the two-metric projection method to the discrete setting where this issue does not arise.

section 4.8 where we describe the algorithm for the finite-dimensional optimization problem obtained from the usual finite element approximation procedure. The intuition developed in the preceding discussion carries over to the discrete setting.

4.5 Finite element approximation

We begin with describing the approximate "finite element" optimization problem, based on a typical choice of discretization spaces, and establish the convergence of the corresponding optimal solutions to a solution of the continuum problem (4.1) in the limit of mesh refinement. Our result proves strong convergence of a subsequence of solutions, and therefore rules out the possibility of numerical instabilities such as checkerboard patterns observed in density-based methods. We remark that similar results are available for the density-based restriction formulations (see for example [128, 127, 31]) and the proof is along the same lines. Such convergence results are essential in justifying an overall optimization approach where one first discretizes a well-posed continuum problem and then chooses an algorithm to solve the resulting finite dimensional problem (this is the procedure adopted in this work). Then, with the FE convergence result in hand, the only remaining task is to analyze the convergence of the proposed optimization algorithms, which is discussed in sections 4.7 and 4.8.

Consider partitioning of Ω into pairwise disjoint finite elements $\mathcal{T}_h = \{\Omega_e\}_{e=1}^{l}$ with characteristic mesh size h. Let $\mathcal{A}_h$ be the FE subspace of $\mathcal{A}$ based on this partition:

$$\mathcal{A}_h = \left\{\rho \in C^0(\overline{\Omega}) : \rho|_{\Omega_e} \in P(\Omega_e), \forall e = 1, \ldots, l\right\} \cap \mathcal{A} \tag{4.28}$$

where $P(\Omega_e)$ is a space of polynomial (rational in the case of polygonal elements—see section 5.1) functions defined on Ω_e. Similarly, we define:

$$\mathcal{V}_h = \left\{\mathbf{u} \in C^0(\overline{\Omega})^d : [\mathbf{u}]_i\,|_{\Omega_e} \in P(\Omega_e), \forall e = 1, \ldots, l, \forall i = 1, \ldots, d\right\} \cap \mathcal{V} \tag{4.29}$$

We also assume that the mesh $\mathcal{T}_h$ is chosen in such a way that the transition from Γ_D to Γ_N is properly aligned with the mesh. In practice, both density and displacement fields are discretized with linear elements (e.g., linear triangles, bilinear quads or linearly-complete convex polygons in two spatial dimensions). To avoid any ambiguity regarding the definition of the FE partitions, we assume a regular refinement of the meshes such that the resulting finite element spaces are ordered, e.g., $\mathcal{A}_h \supseteq \mathcal{A}_{h'}$ whenever $h \leq h'$. We consider the limit $h \to 0$ to establish convergence of solutions under mesh refinement.

What is needed in the proof of convergence is the existence of an interpolation operator

$\mathcal{I}_h : \mathcal{V} \to \mathcal{V}_h$ such that for all $\mathbf{u} \in \mathcal{V} \cap H^k(\Omega)^d$,

$$\left\| \mathcal{I}_h(\mathbf{u}) - \mathbf{u} \right\|_1 \leq C h^{k-1} \left| \mathbf{u} \right|_k \tag{4.30}$$

which in turn shows that $\mathcal{I}_h \mathbf{u} \to \mathbf{u}$ as $h \to 0$. Similarly, we need the mapping $i_h : \mathcal{A} \to \mathcal{A}_h$ for the design space such that $i_h(\rho) \to \rho$ as $h \to 0$. The construction of such interpolants is standard in finite element approximation theory, see for example [34].

The approximate finite element problem, specialized to Tikhonov regularization, is defined by

$$\min_{\rho \in \mathcal{A}_h} \tilde{J}_h(\rho) := J_h(\rho) + \frac{\beta}{2} \left| \rho \right|_1^2 \tag{4.31}$$

where $J_h(\rho) := \ell(\mathbf{u}_{\rho,h}) + \lambda \int_\Omega \rho \mathrm{d}\mathbf{x}$ and $\mathbf{u}_{\rho,h}$ is the solution to the Galerkin approximation of the state equation given by

$$a(\mathbf{u}_h, \mathbf{v}_h; \rho) = \ell(\mathbf{v}_h), \qquad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{4.32}$$

By the principle of minimum potential (cf. (3.5)), we can write

$$\ell(\mathbf{u}_{\rho,h}) = -2 \min_{\mathbf{v}_h \in \mathcal{V}_h} \left[\frac{1}{2} a(\mathbf{v}_h, \mathbf{v}_h; \rho) - \ell(\mathbf{v}_h) \right] = \max_{\mathbf{v}_h \in \mathcal{V}_h} \left[2\ell(\mathbf{v}_h) - a(\mathbf{v}_h, \mathbf{v}_h; \rho) \right] \tag{4.33}$$

From the above relation, it is easy to see that $\mathcal{V}_h \subseteq \mathcal{V}$ implies $\ell(\mathbf{u}_{\rho,h}) \leq \ell(\mathbf{u}_\rho)$ for any given ρ, and therefore

$$\tilde{J}_h(\rho) \leq \tilde{J}(\rho) \tag{4.34}$$

that is, the finite approximation of the state equation leads to a smaller computed value of the cost function for any density field.

Consider a sequence of FE partitions $\mathcal{T}_h$ with $h \to 0$ and let ρ_h be the optimal solution to the associated finite element approximation (4.31), i.e., minimizer of $\tilde{J}_h$ in $\mathcal{A}_h$. We first show the sequence ρ_h is bounded in $H^1(\Omega)$. To see this, fix h_0 in this sequence. If $\hat{\rho}_h$ is the minimizer of $\tilde{J}$ in $\mathcal{A}_h$ (there is no approximation of the displacement field involved here), then

$$\tilde{J}(\hat{\rho}_h) \leq \tilde{J}(\rho_{h_0}) \tag{4.35}$$

since $\rho_{h_0} \in \mathcal{A}_{h_0} \subseteq \mathcal{A}_h$. Now, from the definition of ρ_h and (4.34), we have $\tilde{J}_h(\rho_h) \leq \tilde{J}_h(\hat{\rho}_h) \leq \tilde{J}(\hat{\rho}_h)$ and so

$$\tilde{J}_h(\rho_h) \leq \tilde{J}(\rho_{h_0}) = \tilde{J}_{h_0}(\rho_{h_0}) + [J(\rho_{h_0}) - J_{h_0}(\rho_{h_0})] := \tilde{J}_{h_0}(\rho_{h_0}) + \epsilon_{h_0} \tag{4.36}$$

where ϵ_{h_0} is the finite element error in computing compliance of ρ_{h_0} on mesh $\mathcal{T}_{h_0}$. Since

(4.36) holds for all $h \leq h_0$, we conclude that

$$\limsup_{h \to 0} \tilde{J}_h(\rho_h) \leq \tilde{J}_{h_0}(\rho_{h_0}) + \epsilon_{h_0} \tag{4.37}$$

Both the compliance and volume terms in $\tilde{J}_h(\rho_h)$ are uniformly bounded, and so (4.37) shows $\limsup_h |\rho_h|_1^2 < \infty$. Thus the sequence ρ_h is bounded in $H^1(\Omega)$.

By Rellich's theorem [74], we have convergence of a subsequence, again denoted by ρ_h, strongly in $L^2(\Omega)$ and weakly in $H^1(\Omega)$ to some $\rho^* \in \mathcal{A}$[7]. We next show that ρ^* is a solution to continuum problem, thereby establishing the convergence of the FE approximate problem. First note that by lower semi-continuity of the norm under weak convergence,

$$|\rho^*|_1^2 \leq \liminf_h |\rho_h|_1^2 \tag{4.38}$$

Furthermore, to show convergence of $\mathbf{u}_{\rho_h,h}$ to $\mathbf{u}_{\rho^*}$ in $H^1(\Omega)^d$, first note that Proposition 2.1 implies that up to a subsequence $\mathbf{u}_{\rho_h} \to \mathbf{u}_{\rho^*}$ as $h \to 0$. Moreover,

$$\begin{aligned} \|\mathbf{u}_{\rho^*} - \mathbf{u}_{\rho_h,h}\|_1 &\leq \|\mathbf{u}_{\rho^*} - \mathbf{u}_{\rho_h}\|_1 + \|\mathbf{u}_{\rho_h} - \mathbf{u}_{\rho_h,h}\|_1 \\ &\leq \|\mathbf{u}_{\rho^*} - \mathbf{u}_{\rho_h}\|_1 + \frac{M}{c} \|\mathbf{u}_{\rho_h} - \mathcal{I}_h(\mathbf{u}_{\rho_h})\|_1 \\ &\leq \|\mathbf{u}_{\rho^*} - \mathbf{u}_{\rho_h}\|_1 + \hat{C} h \, |\mathbf{u}_{\rho_h}|_2 \end{aligned} \tag{4.39}$$

where the second inequality follows from Cea's lemma [34] (here M and c denote the continuity and ellipticity constants for the bilinear form) and last inequality follows from estimate (4.30). Hence $\mathbf{u}_{\rho_h,h} \to \mathbf{u}_{\rho^*}$ in $H^1(\Omega)^d$ and so $J_h(\rho_h) \to J(\rho^*)$. Together with the above inequality, we have

$$\tilde{J}(\rho^*) \leq \liminf_h \tilde{J}_h(\rho_h) \tag{4.40}$$

To establish optimality of ρ^*, take any $\rho \in \mathcal{A}_h$. The definition of ρ_h as the optimal solution to (4.31) implies

$$\tilde{J}_h(\rho_h) \leq \tilde{J}_h\left[i_h(\rho)\right] \tag{4.41}$$

Using a similar argument as above, we can pass (4.41) to the limit to show $\tilde{J}(\rho^*) \leq \tilde{J}(\rho)$.

4.6 The discrete problem

We proceed to obtain explicit expressions for the discrete problem (4.31) for a given finite element partition $\mathcal{T}_h$. For each $\rho_h \in \mathcal{A}_h$, we have the expansion $\rho_h(\mathbf{x}) = \sum_{k=1}^m [\mathbf{z}]_k N_k(\mathbf{x})$ where $\mathbf{z}$ is the vector of nodal densities characterizing ρ_h and $\{N_k\}_{k=1}^m$ the set of finite

[7] To see that ρ^* satisfies the bound constraints, as in the proof of Proposition 3.1, we can consider a subsequence for which the convergence is pointwise.

element basis functions for $\mathcal{A}_h$[8]. The finite-dimensional space corresponding to $\mathcal{A}_h$ is simply the closed cube $[\delta_\rho, 1]^m$. Moreover, the vector form for the Tikhonov regularization term is

$$\frac{\beta}{2}\left|\rho_h\right|_1^2 = \frac{1}{2}\mathbf{z}^T\mathbf{G}\mathbf{z} \tag{4.42}$$

where $\mathbf{G}$ is the usual finite element matrix defined by $[\mathbf{G}]_{k\ell} = \beta\int_\Omega \nabla N_k \cdot \nabla N_\ell \mathrm{d}\mathbf{x}$, which is positive semi-definite. Similarly, the volume term $\int_\Omega \rho \mathrm{d}\mathbf{x}$ can be written as $\mathbf{z}^T\mathbf{v}$ where $[\mathbf{v}]_k = \int_\Omega N_k \mathrm{d}\mathbf{x}$.

With regard to state equation (4.32), we make one approximation in the energy bilinear form[9] by assuming that the density field has a constant value over each element, equal to the centroidal value, in the bilinear form. If $\mathbf{x}_e$ denotes the location of the centroid of element Ω_e, we replace each $\rho_h(\mathbf{x})$ by[10]

$$\sum_{e=1}^{l} \chi_{\Omega_e}(\mathbf{x})\rho_h(\mathbf{x}_e) \tag{4.43}$$

in the state equation. The use of piecewise element density is common practice in topology optimization (cf. section 1.4) and makes the calculations and notation simpler. If $\{\mathbf{N}_i\}_{i=1}^q$ denotes the basis functions for the displacement field such that $\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^q [\mathbf{U}]_i \mathbf{N}_i(\mathbf{x})$, the vector form of (4.32) is given by

$$\mathbf{K}\mathbf{U} = \mathbf{F} \tag{4.44}$$

where the load vector $[\mathbf{F}]_i = \int_{\Gamma_N} \mathbf{t}\cdot\mathbf{N}_i \mathrm{d}s$ and the stiffness matrix, with the above approximation of density, is

$$[\mathbf{K}]_{ij} = \int_\Omega \rho_h^p \mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j \mathrm{d}\mathbf{x} = \sum_{e=1}^{l} [\rho_h(\mathbf{x}_e)]^p \int_{\Omega_e} \mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j \mathrm{d}\mathbf{x} \tag{4.45}$$

Let us define the matrix $\mathbf{P}$ whose (e,k)-entry is given by $[\mathbf{P}]_{ek} = N_k(\mathbf{x}_e)$. Then

$$\rho_h(\mathbf{x}_e) = \sum_{k=1}^{m} N_k(\mathbf{x}_e)\,[\mathbf{z}]_k = \sum_{k=1}^{m} [\mathbf{P}]_{ek}\,[\mathbf{z}]_k = [\mathbf{P}\mathbf{z}]_e \tag{4.46}$$

The vector $\mathbf{P}\mathbf{z}$ thus gives the vector of elemental density values. Returning to (4.45) and denoting the *element stiffness* matrix by $\mathbf{k}_e = \int_{\Omega_e} \mathbf{C}\nabla\mathbf{N}_i : \nabla\mathbf{N}_j \mathrm{d}\mathbf{x}$, we have the simplified

[8] Naturally we assume that the basis functions are such that for any $\mathbf{z} \in [\delta_\rho, 1]^m$, the associated density field takes values in $[\delta_\rho, 1]$. This is satisfied, for example, if $0 \le N_k \le 1$ for all k, which is the case for linear convex n-gons [162].

[9] This is a departure from the previous section but it can be accounted for in the convergence analysis.

[10] As before, χ_A is the characteristic function associated with set A.

expression for the global stiffness matrix

$$\mathbf{K}(\mathbf{z}) = \sum_{e=1}^{l} \left([\mathbf{P}\mathbf{z}]_e\right)^p \mathbf{k}_e \tag{4.47}$$

The summation effectively represents the assembly routine in practice. We note the continuity and ellipticity of the energy bilinear form and non-degeneracy of the finite element partition imply that the eigenvalues of $\mathbf{K}(\mathbf{z})$ are bounded below by c_h and above by M_h (which depend on the mesh size, cf. chapter 9 of [34]) for all admissible density vectors $\mathbf{z} \in [\delta_\rho, 1]^m$.

The discrete optimization problem (4.31) can now be equivalently written as (with a slight abuse of notation for J and $\tilde{J}$)

$$\min_{\mathbf{z}\in[\delta_\rho,1]^m} \tilde{J}(\mathbf{z}) := J(\mathbf{z}) + \frac{1}{2}\mathbf{z}^T\mathbf{G}\mathbf{z} \tag{4.48}$$

where

$$J(\mathbf{z}) = \mathbf{F}^T\mathbf{U}(\mathbf{z}) + \lambda \mathbf{z}^T\mathbf{v} \tag{4.49}$$

and $\mathbf{U}(\mathbf{z})$ is the solution to $\mathbf{K}(\mathbf{z})\mathbf{U} = \mathbf{F}$. Observe that matrices $\mathbf{P}$ and $\mathbf{G}$, the vector $\mathbf{v}$, as well as the element stiffness matrices $\mathbf{k}_e$ and load vector $\mathbf{F}$ are all fixed and do not change in the course of optimization. Thus they can be computed once in the beginning and stored.

The gradient of J with respect to the nodal densities $\mathbf{z}$ can readily computed as

$$\partial_k J(\mathbf{z}) = -\mathbf{U}(\mathbf{z})^T \left(\partial_k \mathbf{K}\right) \mathbf{U}(\mathbf{z}) + \lambda [\mathbf{v}]_k \tag{4.50}$$

The expression for $\partial_k \mathbf{K}$ can be obtained from (4.47). Defining the vector of strain energy densities $[\mathbf{E}(\mathbf{z})]_e = p\,[\mathbf{P}\mathbf{z}]_e^{p-1}\,\mathbf{U}(\mathbf{z})^T\mathbf{k}_e\mathbf{U}(\mathbf{z})$, we have

$$\nabla J(\mathbf{z}) = -\mathbf{P}^T\mathbf{E}(\mathbf{z}) + \lambda\mathbf{v} \tag{4.51}$$

With the first order gradient information in hand, we can find the reciprocal approximation[11] of compliance about point $\mathbf{y}$ as

$$J_{\mathsf{rec}}(\mathbf{z};\mathbf{y}) \equiv J(\mathbf{y}) + \lambda\left(\mathbf{z}-\mathbf{y}\right)^T\mathbf{v} + \sum_{k=1}^{m}\left(\frac{[\mathbf{y}]_k}{[\mathbf{z}]_k}\right)\left([\mathbf{z}]_k - [\mathbf{y}]_k\right)\left[-\mathbf{P}^T\mathbf{E}(\mathbf{y})\right]_k \tag{4.52}$$

The Hessian of $J_{\mathsf{rec}}(\mathbf{z};\mathbf{y})$, evaluated at $\mathbf{z} = \mathbf{y}$, is a diagonal matrix with entries

$$h_k(\mathbf{y}) = \partial_{kk} J_{\mathsf{rec}}(\mathbf{y};\mathbf{y}) = \frac{2}{[\mathbf{y}]_k}\left[\mathbf{P}^T\mathbf{E}(\mathbf{y})\right]_k, \quad k = 1,\ldots,m \tag{4.53}$$

[11] The reciprocal approximation to $f(\mathbf{x})$ at point $\mathbf{y}$ is given by $f(\mathbf{y}) + \sum_{k=1}^{m}\left[x_k^{-1}y_k\left(x_k - y_k\right)\partial_k f(\mathbf{y})\right]$

The entries of the vector $\mathbf{E}(\mathbf{y})$ are non-negative for all admissible nodal densities but can be zero and therefore Hessian of $J_{\mathsf{rec}}(\mathbf{z};\mathbf{y})$ is only positive semi-definite.

4.7 Algorithms for the discrete problem

We begin with the generalization of the forward-backward algorithm for solving the discrete problem (4.48) before discussing the two-metric projection variation. As in section 4.2, we consider a splitting algorithm with iterations of the form

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}_n^{\mathsf{L}} \le \mathbf{z} \le \mathbf{z}_n^{\mathsf{U}}}{\operatorname{argmin}} \; Q_J(\mathbf{z};\mathbf{z}_n,\tau_n) + \frac{1}{2}\mathbf{z}^T\mathbf{G}\mathbf{z} \tag{4.54}$$

where, compared to (4.48), the regularization term is unchanged while J is replaced by the following local quadratic model around current iterate $\mathbf{z}_n$

$$Q_J(\mathbf{z};\mathbf{z}_n,\tau_n) = J(\mathbf{z}_n) + (\mathbf{z}-\mathbf{z}_n)^T \nabla J(\mathbf{z}_n) + \frac{1}{2\tau_n}\left\|\mathbf{z}-\mathbf{z}_n\right\|_{\mathbf{H}_n}^2 \tag{4.55}$$

The move limit constraint is accounted for through the bounds

$$\left[\mathbf{z}_n^{\mathsf{L}}\right]_k = \max\left(\delta_\rho, [\mathbf{z}_n]_k - m_n\right), \quad \left[\mathbf{z}_n^{\mathsf{U}}\right]_k = \min\left(1, [\mathbf{z}_n]_k + m_n\right), \qquad k = 1,\ldots,m \tag{4.56}$$

In order to embed the curvature information from the reciprocal approximation (4.52) in the quadratic model, we choose

$$\mathbf{H}_n = \operatorname{diag}(\hat{h}_1(\mathbf{z}_n),\ldots,\hat{h}_m(\mathbf{z}_n)) \tag{4.57}$$

where $\hat{h}_k(\mathbf{z}_n) \equiv \max(h_k(\mathbf{z}_n),\delta_E)$ and, as defined before, $0 < \delta_E \ll \lambda$ is a small positive constant. This modification not only ensures that $\mathbf{H}_n$ is positive definite but also that the eigenvalues of $\mathbf{H}_n$ are uniformly bounded above and below, a condition that is useful for the proof of convergence of the algorithm [26]. Observe that for all $\mathbf{z} \in [\delta_\rho, 1]^m$,

$$0 \le h_k(\mathbf{z}) \le 2\delta_\rho^{-1}\left\|\mathbf{E}(\mathbf{z})\right\|_\infty \le 2p\delta_\rho^{-p-1}M_h\left\|\mathbf{U}(\mathbf{z})\right\|^2 \le 2p\delta_\rho^{-p-1}M_h c_h^{-2}\left\|\mathbf{F}\right\|^2 \tag{4.58}$$

where we used the fact that $\mathbf{U}^T\mathbf{k}_e\mathbf{U} \le \delta_\rho^{-p}\mathbf{U}^T\mathbf{K}(\mathbf{z})\mathbf{U} \le \delta_\rho^{-p}M_h\left\|\mathbf{U}(\mathbf{z})\right\|^2$ and that the eigenvalues of $\mathbf{K}^{-1}$ are bounded above by c_h^{-1}.

The step size parameter τ_n in (4.54) must be sufficiently small so that the quadratic model is a conservative approximation and majorizes J. If $\tau_n > 0$ is chosen so that the update $\mathbf{z}_{n+1}$ satisfies

$$J(\mathbf{z}_{n+1}) \le Q_J(\mathbf{z}_{n+1};\mathbf{z}_n,\tau_n) \tag{4.59}$$

then one can show [26]

$$\tilde{J}(\mathbf{z}_n) - \tilde{J}(\mathbf{z}_{n+1}) \geq \frac{1}{2\tau_n} \|\mathbf{z}_n - \mathbf{z}_{n+1}\|^2_{\mathbf{H}_n} \tag{4.60}$$

If $\mathbf{z}_n$ is a stationary point of $\tilde{J}$, that is $(\mathbf{z} - \mathbf{z}_n)^T \nabla \tilde{J}(\mathbf{z}_n) \geq 0$ for all $\mathbf{z} \in [\delta_\rho, 1]^m$, then $\mathbf{z}_{n+1} = \mathbf{z}_n$ for all $\tau_n > 0$. To see this, we write (4.54) equivalently as

$$\min_{\mathbf{z}_n^{\mathsf{L}} \leq \mathbf{z} \leq \mathbf{z}_n^{\mathsf{U}}} (\mathbf{z} - \mathbf{z}_n)^T \nabla \tilde{J}(\mathbf{z}_n) + \frac{1}{2\tau_n} \|\mathbf{z} - \mathbf{z}_n\|^2_{\mathbf{H}_n + \tau_n \mathbf{G}} \tag{4.61}$$

Since $\mathbf{H}_n + \tau_n \mathbf{G}$ is positive definite and $\mathbf{z}_n$ is a stationary point, the objective function is strictly positive for all $\mathbf{z} \in \left[\mathbf{z}_n^{\mathsf{L}}, \mathbf{z}_n^{\mathsf{U}}\right], \mathbf{z} \neq \mathbf{z}_n$ while it vanishes at $\mathbf{z} = \mathbf{z}_n$, thereby establishing optimality of $\mathbf{z}_n$ for subproblem (4.54). Otherwise, if $\mathbf{z}_n$ is not a stationary point of $\tilde{J}$, then $\mathbf{z}_{n+1} \neq \mathbf{z}_n$ for sufficiently small τ_n, and (4.60) shows that there is a decrease in the objective function. This latter fact shows that *the algorithm is monotonically decreasing*.

A step size parameter satisfying (4.59) is guaranteed to exist if J has a Lipschitz gradient, that is, for some positive constant L,

$$\|\nabla J(\mathbf{z}) - \nabla J(\mathbf{y})\| \leq L \|\mathbf{z} - \mathbf{y}\|, \quad \forall \mathbf{z}, \mathbf{y} \in \mathrm{dom}(J) \tag{4.62}$$

One can show[12] $J(\mathbf{z}) \leq Q_J(\mathbf{z}; \mathbf{z}_n; \tau_n)$ for all $\mathbf{z} \in [\delta_\rho, 1]^m$ if the step size satisfies

$$\tau_n^{-1} \mathbf{H}_n > L\mathbf{I} \tag{4.63}$$

in the sense of quadratic forms, i.e., $\tau_n^{-1}\mathbf{H}_n - L\mathbf{I}$ is positive definite [26]. We verify that the gradient of compliance ∇J given by (4.51) is indeed Lipschitz:

$$\begin{aligned}
\|\nabla J(\mathbf{z}) - \nabla J(\mathbf{y})\| &= p \|\mathbf{E}(\mathbf{z}) - \mathbf{E}(\mathbf{y})\| \\
&\leq p \left[\sum_{e=1}^{l} \left([\mathbf{P}\mathbf{z}]_e^{p-1} \delta_\rho^{-p} M_h \|\mathbf{U}(\mathbf{z})\|^2 - [\mathbf{P}\mathbf{y}]_e^{p-1} \delta_\rho^{-p} M_h \|\mathbf{U}(\mathbf{y})\|^2 \right)^2 \right]^{1/2} \\
&\leq p\delta_\rho^{-p} M_h \left[\sum_{e=1}^{l} \left([\mathbf{P}\mathbf{z}]_e c_h^{-2} \|\mathbf{F}\|^2 - [\mathbf{P}\mathbf{y}]_e c_h^{-2} \|\mathbf{F}\|^2 \right)^2 \right]^{1/2} \\
&\leq p\delta_\rho^{-p} M_h c_h^{-2} \|\mathbf{F}\|^2 \|\mathbf{P}\mathbf{z} - \mathbf{P}\mathbf{y}\| \\
&\leq p\delta_\rho^{-p} M_h c_h^{-2} \|\mathbf{F}\|^2 \|\mathbf{z} - \mathbf{y}\|
\end{aligned} \tag{4.64}$$

The step size τ_n can be selected with *a priori* knowledge of the Lipschitz constant L but this may be too conservative and may slow down the convergence of the algorithm. Instead, in each iteration, one can gradually decrease the step size via a backtracking routine until

[12] This is in fact stronger than (4.59)

$\mathbf{z}_{n+1}$ satisfies (4.59). An alternative, possibly weaker, descent condition is the Armijo rule which requires that for some constant $0 < \nu < 1$, the update satisfies

$$\tilde{J}(\mathbf{z}_n) - \tilde{J}(\mathbf{z}_{n+1}) \geq \nu \left(\mathbf{z}_n - \mathbf{z}_{n+1}\right)^T \nabla \tilde{J}(\mathbf{z}_n) \tag{4.65}$$

Though the implementation of such step size routines is straightforward, due to the high cost of function evaluations for the compliance problem (which requires solving the state equation to compute the value of J), the number of trials in satisfying the descent condition must be limited. Therefore, there is a tradeoff between attempting to choose larger step sizes to speed up convergence and the cost associated with the selection routine. As shown in the next section, we have found that fixing $\tau_n = 1$, which eliminates the cost of backtracking routine, generally leads to a stable and convergent algorithm. In some cases, however, the overall cost can be reduced by using larger step sizes.

Ignoring constant terms in $\mathbf{z}_n$ and rearranging, we can write (4.54) equivalently as

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}_n^{\mathsf{L}} \leq \mathbf{z} \leq \mathbf{z}_n^{\mathsf{U}}}{\operatorname{argmin}} \ \left\| \mathbf{z} - \mathbf{z}_{n+1}^* \right\|^2_{\mathbf{H}_n + \tau_n \mathbf{G}} \tag{4.66}$$

where the interim update $\mathbf{z}_{n+1}^*$ is the given by

$$\mathbf{z}_{n+1}^* = \mathbf{z}_n - \tau_n \left(\mathbf{H}_n + \tau_n \mathbf{G}\right)^{-1} \left[\nabla \tilde{J}(\mathbf{z}_n)\right] \tag{4.67}$$

With the appropriate choice of step size (satisfying any one of the conditions (4.59), (4.63), or (4.65)) and boundedness of $\mathbf{H}_n$, it can be shown that every limit point of the the sequence $\mathbf{z}_n$ generated by the algorithm is a critical point of $\tilde{J}$. For the particular case of quadratic regularization, it is evident from (4.67) that the algorithm reduces to the so-called scaled gradient projection algorithm, and the convergence proof can be found in [26]. A more general proof can be found in the review paper on proximal splitting method by [18] though the metric associated with the proximal term, i.e., $\|\mathbf{z} - \mathbf{z}_n\|^2_{\mathbf{H}_n + \tau_n \mathbf{G}}$ in (4.54), is fixed there.

As seen from (4.54) or (4.66), the forward-backward algorithm requires the solution to a sparse, strictly convex quadratic program subject to simple bound constraints which can be efficiently solved using a variety of methods, e.g., the active set method. Alternatively, the projection of $\mathbf{z}_{n+1}^*$ can be recast as a bound constrained sparse least squares problem and solved using algorithms in [2].

4.8 Two-metric projection variation

Next we discuss a variation of the splitting algorithm that simplifies the projection step (4.66) by augmenting the interim density (4.67). More specifically, we adopt a variant of the two-metric projection method [25, 79], in which the norm in (4.66) is replaced by the

usual Euclidean norm, and the scaling matrix $\mathbf{H}_n + \tau_n \mathbf{G}$ in the interim step (4.67) is made diagonal with respect to the active components of $\mathbf{z}_n$.

Let $I_n = I_n^{\mathsf{L}} \cup I_n^{\mathsf{U}}$ denote the set of active constraints where

$$I_n^{\mathsf{L}} = \left\{ k : [\mathbf{z}_n]_k \leq \delta_\rho + \epsilon \text{ and } \left[\nabla \tilde{J}(\mathbf{z}_n)\right]_k > 0 \right\} \tag{4.68}$$

$$I_n^{\mathsf{U}} = \left\{ k : [\mathbf{z}_n]_k \geq 1 - \epsilon \text{ and } \left[\nabla \tilde{J}(\mathbf{z}_n)\right]_k < 0 \right\} \tag{4.69}$$

Here ϵ is an algorithmic parameter (we fix it at 10^{-3} for the numerical results) that enlarges the set of active constraints in order to avoid the discontinuities that may otherwise arise [25]. Then

$$[\mathbf{D}_n]_{ij} \equiv \begin{cases} 0 & \text{if } i \neq j \text{ and } i \in I_n \text{ or } j \in I_n \\ [\mathbf{H}_n + \tau_n \mathbf{G}]_{ij} & \text{otherwise} \end{cases} \tag{4.70}$$

is a scaling matrix formed from $\mathbf{H}_n + \tau_n \mathbf{G}$ that is diagonal with respect to I_n and therefore removes the coupling between the active and free constraints. The operation in (4.70) essentially consists of zeroing out all the off-diagonal entries of $\mathbf{H}_n + \tau_n \mathbf{G}$ for the active components. Note that any other positive matrix with the same structure as $\mathbf{D}_n$ can be used. The new interim density is then defined as

$$\mathbf{z}_{n+1}^* = \mathbf{z}_n - \tau_n \mathbf{D}_n^{-1} \left[\nabla \tilde{J}(\mathbf{z}_n)\right] \tag{4.71}$$

and the next iterate is given by the Euclidian projection of this interim density onto the constraint set

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}_n^{\mathsf{L}} \leq \mathbf{z} \leq \mathbf{z}_n^{\mathsf{U}}}{\operatorname{argmin}} \left\| \mathbf{z} - \mathbf{z}_{n+1}^* \right\|^2 \tag{4.72}$$

which has an explicit solution

$$[\mathbf{z}_{n+1}]_k = \min\left(\max\left(\left[\mathbf{z}_n^{\mathsf{L}}\right]_k, \left[\mathbf{z}_{n+1}^*\right]_k\right), \left[\mathbf{z}_n^{\mathsf{U}}\right]_k\right), \quad k = 1, \ldots, m \tag{4.73}$$

Since $\mathbf{D}_n^{-1} \nabla \tilde{J}(\mathbf{z}_n)$ can be viewed as the gradient of $\tilde{J}$ with respect to the metric induced by $\mathbf{D}_n$, we can see that the present algorithm consisting of (4.71) and (4.72) utilizes two separate metrics for differentiation and projection operations. The significant computational advantage of carrying out the projection step with respect to the Euclidian norm is due to the particular separable structure of the constraint set. Compared to the forward-backward algorithm discussed before, at the cost of modifying the scaling matrix, the overhead associated with solving the quadratic program (cf. (4.66)) is eliminated.

As in the previous algorithm, one can show that $\mathbf{z}_n$ is a critical point of $\tilde{J}$ if and only if $\mathbf{z}_{n+1} = \mathbf{z}_n$ for all $\tau_n > 0$. Similarly, if $\mathbf{z}_n$ is not a stationary point, then for a sufficiently small step size, the next iterate decreases the value of the cost function, i.e., $\tilde{J}(\mathbf{z}_{n+1}) < \tilde{J}(\mathbf{z}_n)$.

The choice of τ_n can be again obtained from an Amijo-type condition along the projection arc (cf. [25]), namely,

$$\tilde{J}(\mathbf{z}_n) - \tilde{J}(\mathbf{z}_{n+1}) \geq \nu \mathbf{d}_n^T \nabla \tilde{J}(\mathbf{z}_n) \tag{4.74}$$

where the direction vector $\mathbf{d}_n$ is given by

$$[\mathbf{d}_n]_k = \begin{cases} [\mathbf{z}_n]_k - [\mathbf{z}_{n+1}]_k & k \in I_n \\ \left[\tau_n \mathbf{D}_n^{-1} \nabla \tilde{J}(\mathbf{z}_n)\right]_k & k \notin I_n \end{cases} \tag{4.75}$$

In the next section, we will compare the performance of the forward-backward algorithm consisting of (4.66) and (4.67) with the two-metric projection consisting of (4.71) and (4.73).

4.9 Numerical investigations

The model compliance minimization problem adopted here is the benchmark MBB beam problem, whose domain geometry and prescribed loading and boundary conditions are shown in Figure 3.2. Using appropriate boundary conditions, the symmetry of the problem is exploited to pose and solve the state equation only on half of the extended domain. The constituent material $\mathbf{C}$ is assumed to be isotropic with unit Young's modulus and Poisson ratio of 0.3. The volume penalty parameter is $\lambda = 200/|\Omega|$ where $|\Omega|$ is the area of the extended design domain. For all the results in this section, the lower bound on the density is set to $\delta_\rho = 10^{-3}$ and, unless otherwise stated, the SIMP penalty exponent is fixed at $p = 3$. A simple backtracking algorithm is used to determine the value of the step size parameter. Given constants $\tau_0 > 0$ and $0 < \sigma < 1$, the step size parameter in the nth iteration is given by

$$\tau_n = \sigma^{k_n} \tau_0 \tag{4.76}$$

where k_n is the smallest non-negative integer such that τ_n satisfies (4.65) or (4.74). In practice, this means that we begin with the initial step size τ_0 and reduce it by a factor of σ until descent conditions are satisfied. The descent parameter is set to $\nu = 10^{-3}$ and the backtracking parameter is $\sigma = 0.6$. Note that larger ν leads to a more severe descent requirement and subsequently smaller τ_n. Similarly, smaller σ reduces the step size parameter by a larger factor which can decrease the number of backtracking step. Note, however, that using small step sizes may lead to slow convergence of the algorithm.

Since each backtracking step involves evaluating the cost functional and therefore solving the state equation, as a measure of computational cost, we keep track of the total number of backtracking steps (i.e., $\sum_n k_n$) in addition to the total number of iterations. The

algorithm	$\mathbf{H}_n$	τ_0	# it.	# bt.	$\ell(\mathbf{u}_\rho)$	$R(\rho)$	$V(\rho)$	$\tilde{J}(\rho)$	E_1	E_2
FBS	identity	1	316	0	100.019	8.553	0.5120	210.965	9.962e-6	9.943e-5
FBS	identity	2	215	154	100.093	8.537	0.5114	210.914	9.178e-6	5.812e-5
FBS	reciprocal	1	186	0	99.937	8.594	0.5125	211.032	9.769e-6	9.363e-5
FBS	reciprocal	2	91	39	100.095	8.568	0.5117	211.008	4.926e-6	9.746e-5
TMP	identity	1	330	0	100.076	8.533	0.5117	210.951	9.958e-6	9.973e-5
TMP	identity	2	151	78	100.060	8.556	0.5116	210.938	9.639e-6	5.900e-5
TMP	reciprocal	1	179	0	99.943	8.592	0.5125	211.031	9.878e-6	9.453e-5
TMP	reciprocal	2	85	34	100.078	8.578	0.5117	210.999	9.043e-6	8.074e-5

Table 4.1: Summary of influence of various factors in the algorithm for the MBB problem with $\beta = 0.06$. The acronym FBS designates the forward-backward algorithm and TMP refers to the two-metric projection algorithm. Forth and fifth columns show the total number of iterations and backtracking steps. The remaining columns show the final value of compliance $\ell(\mathbf{u}_\rho)$, regularization term $R(\rho)$, volume fraction $V(\rho) = |\Omega|^{-1} \int_\Omega \rho \mathrm{d}\mathbf{x}$, the regularized objective $\tilde{J}(\rho)$, the relative change in cost function value E_1 and the error in satisfaction of the first order conditions of optimality E_2

Figure 4.4: Final density field for the MBB problem and $\beta = 0.06$ plotted in grayscale. This result was generated using the TMP algorithm with $\tau_0 = 2$ and $m_n = 1$

convergence criteria adopted here is based on the relative decrease in the objective function

$$E_1 = \frac{\left|\tilde{J}(\mathbf{z}_{n+1}) - \tilde{J}(\mathbf{z}_n)\right|}{\left|\tilde{J}(\mathbf{z}_n)\right|} \leq \epsilon_1 \tag{4.77}$$

and the satisfaction of the first order conditions of optimality according to

$$E_2 = \frac{\left\|\Pi[\mathbf{z}_{n+1} - \nabla\tilde{J}(\mathbf{z}_{n+1})] - \mathbf{z}_{n+1}\right\|}{\left\|\mathbf{z}_{n+1}\right\|} \leq \epsilon_2 \tag{4.78}$$

Here Π is the Euclidian projection onto the constraint set $[\delta_\rho, 1]^m$ defined by $[\Pi(\mathbf{y})]_i = \min\left(\max\left(\delta_\rho, [\mathbf{y}]_i\right), 1\right)$. Unless otherwise stated, we have selected $\epsilon_1 = 10^{-5}$ and $\epsilon_2 = 10^{-4}$.

We begin with the investigation of the behavior of two forms of the algorithm with different choice of parameters discussed in the previous section. In particular, we compare the forward-backward algorithm with the two-metric projection method and investigate the influence of the Hessian approximation. In addition to the choice of $\mathbf{H}_n$ defined by (4.57), we also consider a fixed scaling of the identity matrix

$$\mathbf{H}_n \equiv \alpha \mathbf{I}, \qquad n = 1, 2, \ldots \tag{4.79}$$

for which the algorithm becomes the basic forward-backward algorithm with the same proximal term in every iteration. The scaling coefficient α is set to $4\lambda A$ where A is the area of an element. This choice is made so that the step size parameter τ_n is the same order of magnitude as with reciprocal Hessian. The other parameter investigated here is the initial step size parameter τ_0 and we consider two choices $\tau_0 = 1$ and $\tau_0 = 2$. In all cases, the move limit is fixed at $m_n = 1$ for all n and thus $\mathcal{A}_n = \mathcal{A}$.

The model problem is the MBB beam discretized with a grid of 300×50 bilinear quad elements and Tikhonov regularization parameter is set to $\beta = 0.06$. The initial guess in all cases is taken to be uniform density field $\rho_h \equiv 1/2$. All the possible combinations of the above choices produce the same final topology, similar to the representative solution shown in Figure 4.4. *This shows the framework exhibits stable convergence to the same final solution and is relatively insensitive to various choices of algorithmic parameters for this level of regularization.* What is different, however, is the speed of convergence and the required computational effort as measured by the number of the backtracking steps, total number of iterations, and cost per iteration. The results are summarized in Table 4.1.

First we note that the initial step size $\tau_0 = 1$ does not lead to any backtracking steps which means that in each iteration the step size parameter is $\tau_n = 1$. By contrast, using the larger initial step size parameter $\tau_0 = 2$ requires backtracking steps to satisfy the descent condition but substantially reduces the total number of iterations. Moreover, in all cases, the constant Hessian (4.79) requires nearly twice as many iterations and backtracking steps compared to the "reciprocal" Hessian. *This highlights the fact that embedding the reciprocal approximation of compliance does indeed lead to faster convergence.* Overall, the best performance is obtained using the reciprocal approximation and larger initial step size parameter.

For this problem, the forward-backward algorithm and the two-metric projection method roughly have the same number of iterations and backtracking steps. However, the cost per iteration for the two-metric projection is significantly lower since the projection step is computationally trivial. Therefore, the two-metric projection is more efficient.

Next we investigate the performance of the algorithm for a smaller value of the regularization parameter which is expected to produce more complex topologies. For the next set of results, we set $\beta = 0.01$. In all cases considered, the forward-backward and the two-metric projection algorithms both give identical final topologies with roughly the same number of iterations and so we only report the results for the two-metric projection algorithm. Also, as demonstrated by the first study, the use of reciprocal approximation leads to better and faster convergence of the algorithm so we limit the remaining results to the "reciprocal"

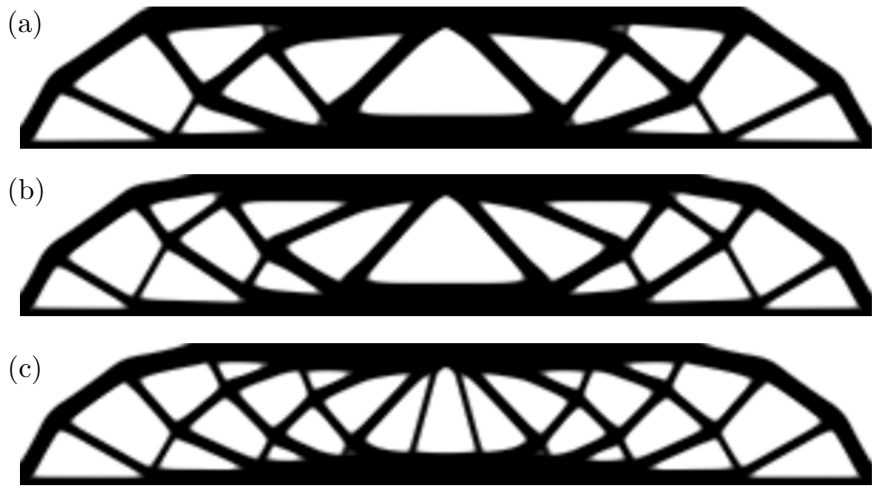


Figure 4.5: Final densities plotted in grayscale for the MBB problem and $\beta = 0.01$. The results are generated using the TMP algorithm with (a) $\tau_0 = 2$, $m_n = 1$ (b) $\tau_0 = 1$, $m_n = 1$ (c) $\tau_0 = 1$, $m_n = 0.03$

$\mathbf{H}_n$. The tolerance level $\epsilon_2 = 10^{-4}$ for satisfaction of the optimality condition is relatively stringent in this case due to the complexity of final designs (compared to $\beta = 0.06$) and leads to a large number of iterations with little change in density near the optimum. We therefore increase the tolerance to $\epsilon_2 = 2 \times 10^{-4}$ which gives nearly identical final topologies but with fewer iterations.

We examine the influence of the step size parameter and move limit, which unlike the previous case of large regularization parameter, can lead to different final solutions. We consider two possible initial step size parameters $\tau_0 = 1$ and $\tau_0 = 2$, as well as two choices for the move limit $m_n \equiv 1$ and $m_n \equiv 0.03$. Here we are using a fixed move limit m_n for all iteration n. It may be possible to devise a strategy to increase m_n in the later stages of optimization to improve convergence. The results are summarized in Table 4.2 and the final solutions are shown in Figure 4.5.

First note that with no move limit constraints, i.e., $m_n = 1$, the final solution with the more aggressive choice of initial step size parameter ($\tau_0 = 2$) is less complex and has fewer members compared to $\tau_0 = 1$, which as before does not require any backtracking steps. Note, however, that the more aggressive scheme in fact requires more iterations to converge. In the presence of move limits, there is no backtracking step with either choice of step size but the larger step size does reduce the total number of iterations. The final topologies are identical and have more members compared to the solutions obtained without the move limits. It is interesting to note that the overall iteration count is lowest for $\tau_0 = 2$ and $m_n = 0.03$ despite the limit on the change in density in each iteration. As noted earlier, the use of move limits can stabilize the convergence of the topology optimization problem.

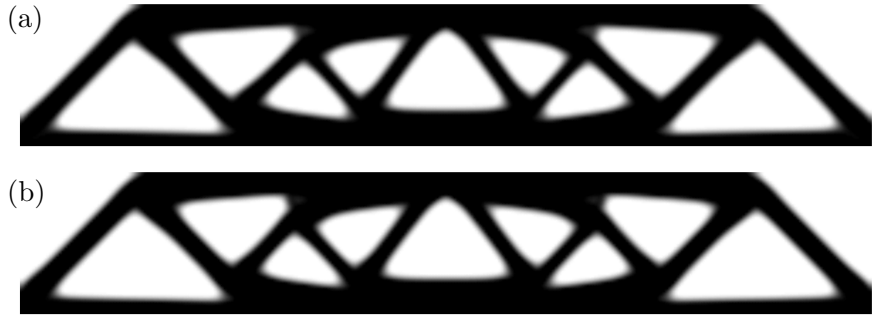


Figure 4.6: Final densities plotted in grayscale for the MBB problem with $\beta = 0.06$ and SIMP penalty exponent (a) $p = 4$ (b) $p = 5$

The overall trend that the move aggressive choice of parameters produce less complex final solutions is due to the fact that member formation occurs early on in the algorithm. The most aggressive algorithm ($\tau_0 = 2$, $m_n = 1$) still produces the best solution as measured by $\tilde{J}$ while the solution obtained enforcing the move limit $m_n = 0.03$ has the lowest value of compliance J (due to distribution of members and slightly higher volume fraction).

We note that aside from the higher degree of complexity, the optimal densities for $\beta = 0.01$ contain fewer intermediate values compared to the solution for $\beta = 0.06$. One measure of discreteness used in [141] is given by

$$M(\rho) = \frac{1}{|\Omega|} \int_{\Omega} 4\,(\rho - \delta_\rho)\,(1 - \rho)\,\mathrm{d}\mathbf{x} \tag{4.80}$$

which is equal to zero if ρ takes only values of δ_ρ and 1. For the solutions shown in Figure 4.5, $M(\rho)$ is equal to 6.98%, 7.64% and 8.90% from top to bottom, respectively. In contrast, the optimal density for $\beta = 0.06$ (cf. Figure 4.4) has a discreteness measure of 15.0%. By increasing the value of the SIMP exponent p, the optimal densities can be made more discrete. The results for $\beta = 0.06$ using $p = 4$ and $p = 5$ are shown in Figure 4.6. While the optimal topologies are nearly identical to that the solution for $p = 3$, the discrete measure is lower to 13.1% and 12.1%, respectively. Observe, however, that the layer of intermediate densities around the boundary cannot be completely eliminated even when p is increase to a very large value since the Tikhonov regularizer is singular in the discontinuous limit of density.

As shown in the previous section, the optimal solutions to the discrete problem converge to an optimal solution of the continuum problem as the finite element mesh is refined. We next demonstrate numerically that solutions produced by the present optimization algorithms appear to be stable with respect to mesh refinement. We do this for the case of $\beta = 0.01$ using the two-metric projection algorithm with $\tau_n \equiv 1$ where the final topology is

algorithm	τ_0	m_n	# it.	# bt.	$\ell(\mathbf{u}_\rho)$	$R(\rho)$	$V(\rho)$	$\tilde{J}(\rho)$	E_1	E_2
TMP	1	1	138	0	102.306	4.669	0.4740	201.779	6.989e-6	1.978e-5
TMP	2	1	169	62	102.716	4.075	0.4720	201.189	9.780e-6	1.679e-5
TMP	1	0.03	153	0	100.738	5.185	0.4855	203.014	7.217e-6	1.998e-4
TMP	2	0.03	98	0	100.568	5.173	0.4862	202.970	9.795e-6	1.566e-4

Table 4.2: Summary of the results for the MBB beam problem with $\beta = 0.01$

relatively complex and the algorithm is expected to be more sensitive. As shown in Figure 4.7, we solve the problem using finer grids consisting of 600×100 and 1200×200 bilinear square elements. The final density distribution is nearly identical indicating convergence of optimal densities in the L^p-norm.

We conclude this section with design of a compliant force inverter for which the cost functional is no longer self-adjoint and therefore, unlike compliance, the gradient field may take both negative and positive values in the domain. The objective of mechanism design is to identify a structure that maximizes the force exerted on a workpiece under the action of an external actuator. As illustrated in Figure 4.8, the force inverter transfers the input force of the actuator to a force at the prescribed output location in the opposite direction. We refer to the section 1.6 for the formulation of this problem. The cost functional, in the discrete setting, is given by

$$J(\mathbf{z}) = -\mathbf{L}^T\mathbf{U}(\mathbf{z}) + \lambda \mathbf{z}^T\mathbf{v} \tag{4.81}$$

where $[\mathbf{L}]_i = \int_{\Gamma_{S_1}} \mathbf{k}_1 \cdot \mathbf{N}_i \mathrm{d}s$ and $\mathbf{U}(\mathbf{z})$ solves and, as before, $\mathbf{U}(\mathbf{z})$ is the solution to $[\mathbf{K}(\mathbf{z}) + \mathbf{K}_s]\,\mathbf{U} = \mathbf{F}$. Here $\mathbf{K}_s$ is the stiffness matrix associated with linear springs $\mathbf{k}_1$ and $\mathbf{k}_2$. The gradient of J can be readily computed as $\nabla J(\mathbf{z}) = -\mathbf{P}^T\overline{\mathbf{E}}(\mathbf{z}) + \lambda\mathbf{v}$ where

$$\left[\overline{\mathbf{E}}(\mathbf{z})\right]_e = p\,[\mathbf{P}\mathbf{z}]_e^{p-1}\,\overline{\mathbf{U}}(\mathbf{z})^T\mathbf{k}_e\mathbf{U}(\mathbf{z}) \tag{4.82}$$

and $\overline{\mathbf{U}}(\mathbf{z})$ is the solution to the *adjoint* problem

$$[\mathbf{K}(\mathbf{z}) + \mathbf{K}_s]\,\overline{\mathbf{U}} = \mathbf{L} \tag{4.83}$$

It is evident that ∇J can take both positive and negative values. The main implication of this for the proposed algorithm is that the reciprocal approximation of the cost functional is not convex and so we cannot use its Hessian directly in the proximal term of the quadratic model. A simple alternative that we tested is to use (4.57) with the diagonal entries modified as

$$h_k(\mathbf{y}) = \left| \frac{2}{[\mathbf{y}]_k} \left[\mathbf{P}^T\overline{\mathbf{E}}(\mathbf{y})\right]_k \right| \tag{4.84}$$

Such an approximation has been previously explored by [83, 84] and is similar in spirit to approximations in Svanberg's Method of Moving Asymptotes [153]. We defer a more

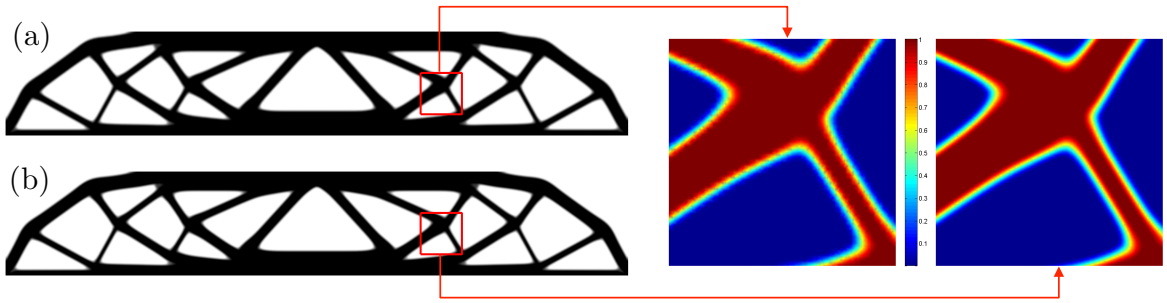


Figure 4.7: Results of the mesh refinement study with (a) 600×100 (b) 1200×200 elements.

detailed study of suitable approximation of the Hessian for general problems to our future work which, as illustrated here, must be based on *a priori* knowledge of the cost functional.

The compliant mechanism design is known to be more prone to getting trapped in suboptimal local minima. One such local minimum is $\rho_h \equiv \delta_\rho$ where the entire structure is eliminated and virtually no work is transferred between the input actuator and the output location. For this case, the value of the cost functional is roughly zero since there is no density variation and minimum volume of material. To avoid converging to this solution, we use a smaller step size parameter $\tau_n = 0.1$. We also begin with small volume penalty parameter of $\lambda = 0.02$ which is then increased to $\lambda = 0.15$ once the value of cost functional reaches a negative value. This point roughly corresponds to an intermediate density distribution in which the structure connects the input force to the output location. The final solution for $\beta = 3 \times 10^{-4}$, a grid of 160×160 quadrilateral elements, and the two-metric projection algorithm is shown in Figure 4.8. This solution required a total of 140 iterations.

4.10 Discussion and closing remarks

Since the splitting algorithm presented here is a first-order method, it is also appropriate to compare its performance to the gradient projection algorithm, which is among the most basic first-order methods for solving constrained optimization problems. The next iterate in the gradient projection method is simply the projection of the unconstrained gradient descent step onto the admissible space. In the absence of move limits and in the discrete setting, we have the following update expression

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}\in[\delta_\rho,1]^m}{\operatorname{argmin}} \left\| \mathbf{z} - \left[\mathbf{z}_n - \frac{\tau_n}{\alpha} \nabla \tilde{J}(\mathbf{z}_n) \right] \right\|^2 \tag{4.85}$$

where the scaling parameter $\alpha = 4\lambda A$ is defined as before in order to allow for a direct comparison with the forward-backward splitting in the case $\mathbf{H}_n = \alpha \mathbf{I}$. We determine the

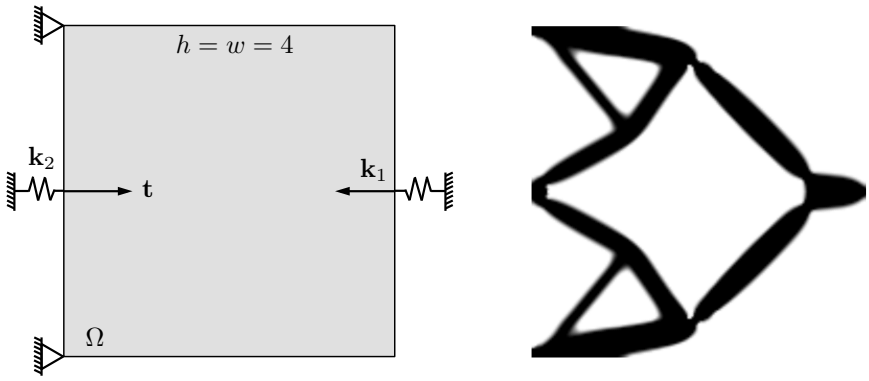


Figure 4.8: The design domain and boundary conditions for the force inverter problem (left) and the optimal topology (right). For this example, $\|\mathbf{k}_1\| = \|\mathbf{k}_2\|$

step size parameter τ_n in each iteration using the backtracking procedure (4.76) based on the Armijo-type descent condition (4.65). Note that due to the simple structure of the constraint set, computing the gradient $\nabla \tilde{J}$ constitutes the main computational cost of the gradient projection algorithm in each iteration. Table 4.3 summarizes the results for the MBB beam problem with $\beta = 0.06$ for two different choice of initial step size parameter τ_0. First observe that the step sizes are smaller compared to the forward-backward algorithm, a fact that can be seen from the equivalent expression for (4.85) given by

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}\in[\delta_\rho,1]^m}{\operatorname{argmin}} \; \tilde{J}(\mathbf{z}_n) + (\mathbf{z}-\mathbf{z}_n)^T \nabla \tilde{J}(\mathbf{z}_n) + \frac{1}{2\tau_n} \|\mathbf{z}-\mathbf{z}_n\|^2_{\alpha\mathbf{I}} \tag{4.86}$$

This shows that in each iteration, we construct a quadratic model for the composite objective $\tilde{J}$. By constrast, the quadratic model in (4.54) is only used for J and the regularization term appears exactly. Since $\nabla \tilde{J}$ has a larger Lipschitz constant compared to ∇J, it is therefore expected that τ_n must be smaller to ensure descent. It is also instructive to recall the informal derivation of the forward-backward algorithm in section 3.3 where the main difference with the gradient projection algorithm was the use of a semi-implicit (in place of an explicit) temporal discertization of the gradient flow equation. Note that the gradient projection algorithm converged to the same solution as before (cf. Figure 4.4) though in the case of $\tau_0 = 0.25$, the convergence was too slow and we terminated the algorithm after 1,000 iterations.

Since the Method of Moving Asymptotes [153] is the most widely used algorithm in the topology optimization literature, we also tested its performance using the same MBB problem. We followed the common practice and used the algorithm as a black-box optimization routine. In particular, we provided the algorithm with the gradient of composite objective $\tilde{J}$ and did not make any changes to the open source code provided by Svangberg[13]. MMA

[13]We remark that with a few exceptions, MMA is used in the same way by Borrvall in a review paper [28] where

algorithm	τ_0	# it.	# bt.	$\tilde{J}(\rho)$	E_2
GP	0.25	1000*	0	210.74	1.362e-4
GP	0.5	568	79	210.68	8.939e-5
MMA	–	1000*	0	213.39	1.913e-4

Table 4.3: Summary of the results for gradient projection and MMA algorithm for the MBB beam problem with $\beta = 0.06$. The asterisk indicates that the maximum allowed iteration count of 1,000 was reached before the convergence criteria was met

internally generates a separable convex approximation to $\tilde{J}$ using reciprocal-type expansions with appropriately defined and updated asymptotes. Though such approximations are suitable for the structural term, they may be inaccurate for the Tikhonov regularizer and the composite objective. As shown in Table 4.3, MMA did not converge (according to the convergence criteria described earlier) in 1,000 iterations before it was terminated. Furthermore, not only was the final value of the objective function larger than that obtained by gradient projection or either splitting algorithm, the final density was topologically different from the solution shown in the Figure 4.4.

The fact that the present splitting framework outperforms MMA should not be surprising. Unlike MMA, which is far more general and can handle a much broader class of problems [154], the present algorithm is tailored to the specific structure of (4.1) (or (4.48) in the discrete) and provides an ideal treatment of its constituents. First, the composite objective is the sum of two terms and algorithm deals with each term separately. The regularization term R is represented with a high degree fidelity since the resulting subproblem with its simple structure can be solved efficiently. The structural term J, while expensive to compute, contains many local minima and very fast convergence usually at best reaches a suboptimal local minimum. Moreover, J tends to be rather flat near stationary points and so one should not require a high level accuracy for satisfaction of the first order conditions of optimality. As a side remark, these characteristics indicate that second order methods do not pay off given their significantly higher computational cost per iteration[14]. The other drawback of using exact second order information is the storage requirements, quadratic in the size of the problem, which can be prohibitive for large-scale problems such as those encountered in practical applications of topology optimization. Therefore, first order methods are better suited for minimizing J.

In the splitting algorithm proposed here, we use additional knowledge about the behavior of J to construct accurate approximations using only first order information and minimal storage requirements. Furthermore, the two-metric approach allows for a computationally efficient treatment of the constraint set. In fact, the proposed approach is aligned with

he compares various regularizations schemes, including Tikhonov regularization.

[14]Computing exact Hessian information is especially expensive for PDE-constrained problem since every Hessian-vector product requires the solution to an adjoint system.

the renewed interest in first-order convex optimization algorithms for solving large-scale inverse problems in signal recovery, statistical estimation, and machine learning [176, 53, 33, 69]. Our rather restricted and narrow comparison with MMA is meant to motivate the virtue of developing such tailored algorithms. In the topology optimization community, frequently when new formulations or new physical problems are presented, the resulting numerical optimization problems are solved using blackbox algorithms. As a result, not only is the computational performance frequently poor but also the quality of the final solutions, depending on the choice of algorithmic parameters or continuation schemes, may be adversely affected. This shows that, aside from efficiency, robustness is also a major issue. Although the extremely high sensitivity to parameters to a large extent is intrinsic to the size, nonconvexity and nonsmoothness of optimal shape problems, we emphasize that it should be minimized as much as possible. Developing an appropriately-designed optimization algorithm that fits the structure of the problem at hand can be key to achieving this.

Chapter 5

Discretizations Based on Polygonal Finite Elements

For all the numerical examples presented in this thesis, the same computational mesh is used to define the discretization of the design field (e.g., densities or implicit functions) and the response field associated with the governing state equation (e.g., displacements). This approach is advantageous from a practical point of view since one only needs to keep track of the information for a single mesh despite the fact that (at least) two independent fields are present in the problem. However, it was mentioned in chapter 1 that this procedure can lead to certain numerical instabilities, most notably the appearance of checkerboard patterns in the optimal solutions to the compliance minimization problem. It is well-known that lower-order Lagrangian elements (e.g., linear triangles and bilinear quadrilaterals in two spatial dimensions), when used for the discretization of displacements, produce spurious minimum compliance designs in the absence of any regularization of the density field [62, 91, 142].

The appearance of these numerical instabilities, akin to those plaguing some finite element discretizations of mixed variational problems, is indicative of inadequate or poor approximation characteristics of the chosen spaces. In other words, they are associated with the poor numerical modeling of the response of the design. For example, the checkerboard pattern has an artificially high stiffness[1] when modeled by bilinear quadrilateral elements and therefore is economical in the optimization process in stiffness problems [62]. Figure 5.1 shows MBB beam solutions using piecewise constant discretization of the density field (with no additional restriction imposed) for different levels of mesh refinement. For the solutions on the left column, the displacement field was discretized using bilinear quadrilateral elements while regular hexagonal Wachspress elements (see section 5.1) were used for the middle column. For the quad mesh, the spurious checkerboard patterns are present for all meshes indicating the problem is not resolved with mesh refinement. By contrast, the polygonal solutions are free of such anomalies. This can be attributed to the fact that fine-scale patterns in a hexagonal tessellation, when modeled by Wachspress elements, are not artificially stiff. In [161], the numerical (homogenized) stiffness of some possible patterns were computed and shown to be close to the stiffness of penalized homogeneous distribution of material. The MBB solutions obtained from an unstructured polygonal discretization are also checkerboard-free as shown in the right column of Figure

[1] Physically, a checkerboard patch should have zero stiffness since it consists of eccentric point connections.

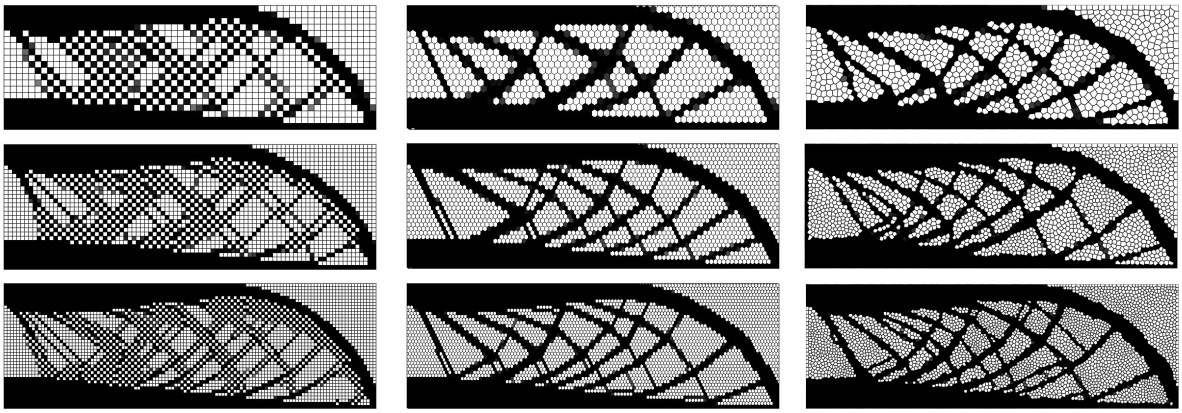


Figure 5.1: The solutions to the MBB beam problem using structured quadrilateral (left) structured hexagonal (middle) and unstructured polygonal (right) meshes

5.1. These observation serve as a motivation to study the use of polygonal finite elements as means to discretize the topology optimization problem. As discussed later, the chosen isoparametric formulation is a natural extension of standard linear triangles and bilinear quadrilaterals to all convex n-gons.

It is important to emphasize that the regularization mechanism in the various restriction formulations discussed so far in this thesis can eliminate the appearance of the fine-scale instabilities provided that the mesh is sufficiently refined. For example, in the density filtering formulation, the most rapid variation of an admissible density function is limited by the radius (support size) of the filtering kernel. Checkerboard or similar patterns (e.g., islands and layers in the case of piecewise linear density discretization) are excluded if the mesh resolves this length scale. Similarly, the norm of the density gradient or its total variation can be made arbitrarily large for such grid-scale oscillations by refining the mesh. However, given the fact that the stabilities are related to poor approximation of the state equation, *there remains some interest in obtaining stable finite element formulations without the need for imposing any further constraints on the design field.* We remark that for certain topology optimization problems arising in fluid mechanics (see section 5.6 for examples), the original continuum problem is well-posed and so there is no need for regularization of design field. Yet an appropriate discretization of the fluid flow equations must be chosen to ensure accurate analysis of the design. Certain mixed formulations, for example, are known to yield poor approximations of the velocity field (due to the so-called locking phenomenon) and incorrect pressure field (due to the presence of spurious modes) and are therefore not reliable for use in optimization. We show by means of numerical examples and investigation of the so-called Babuska-Brezzi condition that a stable mixed formulation for incompressible Stokes equation can be obtained using polygonal finite elements. In summary, it is advantageous in topology optimization (or more broadly speaking, PDE-

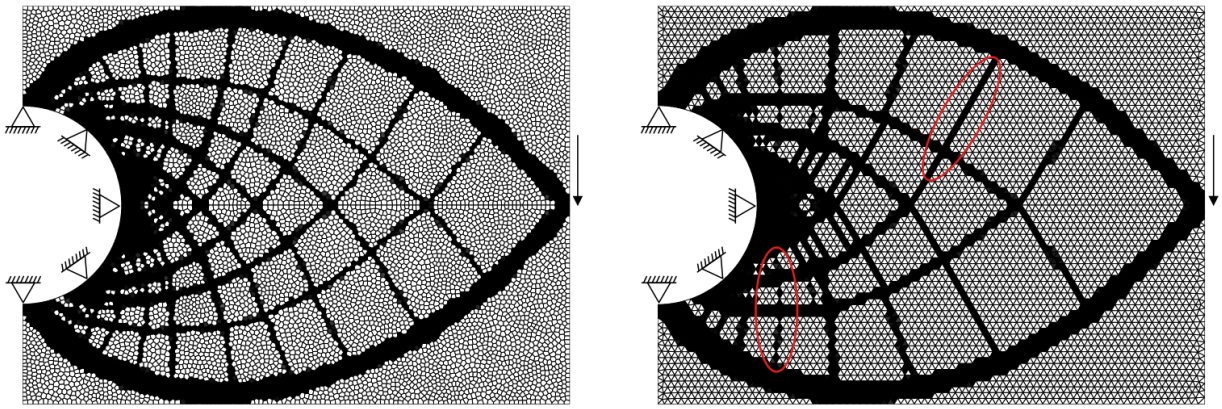


Figure 5.2: Solutions to the Michell cantilever beam problem based on (a) 10,000 polygonal elements and (b) 10,220 quadratic triangular elements. Both meshes are constructed to be symmetric about the horizontal axis

constrained optimization problems) to use a numerical method that best suits the physics of the problem in order to compute the response of the candidate designs as accurately and efficiently as possible. The use of polygonal discretizations, in this regard, seems to be appropriate for certain two-dimensional problems in solid and fluid mechanics problems.

We close this section with another observation regarding the influence of mesh geometry in the present context. In topology optimization literature, uniform structured meshes are often used since they reduce the computational cost associated with element calculations. As discussed in the appendix, when using an unstructured mesh, the invariant quantities such as the element stiffness matrices and the global stiffness matrix connectivity can be computed once and stored for use in subsequent optimization iterations, which in turn eliminates the overhead associated with repeated element calculations.

A problem less often noted is that the constrained geometry of structured meshes, in certain settings, can lead to mesh-dependent sub-optimal designs. For example, in [127], it is pointed out for the perimeter constraint formulation, a straightforward calculation of the total variation of density with a piecewise constant discretization on a regular grid "will encourage the structural edges to be parallel with the FE-edges" and thus "the method suffers from a slight *rotational mesh-dependence*." The consequence is that the finite element solutions, defined on such meshes, may not converge to a solution of the continuum problem[2]. The approach advocated in the above-mentioned paper is to replace the total variation functional by a non-isotropic version in order to make the discretization compatible with

[2]We must note that this rotational mesh-dependency is not present for the filtering or Tikhonov regularization schemes and the standard finite element discretizations are consistent. Aside the from the convergence proof for the Tikhonov regularization in chapter 4, we also refer to the results shown in Figure 3.5(a) and (b) which illustrate the final densities are not sensitive to the mesh geometry.

this new measure of perimeter. An alternative solution would be to use an unstructured mesh with higher degree of isotropy. We do not pursue this issue further in this thesis but we illustrate the influence of mesh geometry on the optimal layouts with an example. Consider the minimum compliance design for the so-called Michell cantilever beam, whose optimal frame layout consists of an orthogonal network of members [152]. In Figure 5.2, we compare the solutions obtain from piecewise constant density discretization (with no additional constraints) on a structured triangular mesh and an unstructured polygonal discretization. Both meshes are constructed to be symmetric about the horizontal axis at mid-depth and quadratic triangles were used to avoid checkerboard-type instabilities. Since the geometry of the mesh in this setting dictates the possible layout of material and orientation of members, one expects that the solution obtained on a less-biased mesh to be better. Indeed the polygonal mesh has more members and thus exhibits higher resolution and the general layout of members is in better agreement with the Michell solution. The members intersect at roughly 90° angles and are spaced more evenly and the principal stresses for the optimal design are aligned with the members according to the Michell layout theory (see also Figure 22 in [162]). The triangular mesh, on the other hand, suffers from the limitation of its geometry. Members that line up with the mesh must strictly conform to it, while others are poorly approximated (see the members marked in Figure 5.2).

5.1 Convex polygonal finite elements

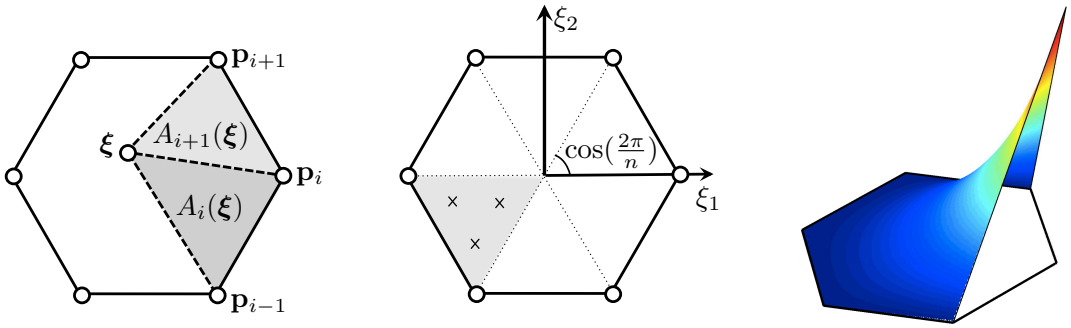
In this section, we present the element formulation for convex n-gons outlined in [149] which consists of Wachspress interpolation functions and standard isoparametric transformations. The resulting finite element approximation space is conforming on convex polygonal meshes, for example, those constructed using the Voronoi meshing algorithm developed in the next chapter. For $n = 3$ and $n = 4$, the element is identical to the commonly used constant strain triangle and isoparametric bilinear quadrilateral, respectively.

Consider a regular n-gon $V^n \subseteq \mathbb{R}^2$ with vertices located at

$$\mathbf{p}_i = \left(\cos \frac{2\pi i}{n}, \sin \frac{2\pi i}{n} \right), \quad i = 1, \ldots, n \tag{5.1}$$

The Wachspress interpolation function corresponding to node i is defined as [170, 149]

$$N_i(\boldsymbol{\xi}) = \frac{\alpha_i(\boldsymbol{\xi})}{\sum_{j=1}^{n} \alpha_j(\boldsymbol{\xi})} \tag{5.2}$$



(a) (b) (c)

Figure 5.3: (a) Illustration of the triangular areas $A_i(\boldsymbol{\xi}) := A(\mathbf{p}_{i-1}, \mathbf{p}_i, \boldsymbol{\xi})$ used to define the interpolant α_i (b) triangulation of the reference regular polygon and integration points defined on each triangle (c) Wachspress interpolation function for a regular hexagon

where α_i are defined by[3]

$$\alpha_i(\boldsymbol{\xi}) = \frac{A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})}{A(\mathbf{p}_{i-1}, \mathbf{p}_i, \boldsymbol{\xi}) A(\mathbf{p}_i, \mathbf{p}_{i+1}, \boldsymbol{\xi})} \tag{5.3}$$

Here A denotes the area of the triangle with vertices located at its arguments (see Figure 5.3(a)). Since the n-gon is regular, $A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})$ is the same for all i and thus can be factored out of expression (5.2). In particular, instead of (5.3), we can use

$$\alpha_i(\boldsymbol{\xi}) = \frac{1}{A(\mathbf{p}_{i-1}, \mathbf{p}_i, \boldsymbol{\xi}) A(\mathbf{p}_i, \mathbf{p}_{i+1}, \boldsymbol{\xi})} \tag{5.4}$$

in (5.2). Note that $\boldsymbol{\xi}$ must be restricted to the interior of V^n for the interpolants to be well-defined. However, the Wachspress functions, which are infinitely differentiable on the interior of V^n, can be extended continuously to $\overline{V^n}$ such that if $\boldsymbol{\xi}$ lies on the edge between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, we have

$$N_i(\boldsymbol{\xi}) = 1 - \frac{\|\boldsymbol{\xi} - \mathbf{p}_i\|}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}, \quad N_{i+1}(\boldsymbol{\xi}) = \frac{\|\boldsymbol{\xi} - \mathbf{p}_i\|}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}, \quad N_j(\boldsymbol{\xi}) = 0 \ \forall j \neq i, i+1 \tag{5.5}$$

This shows that Wachspress functions are linear along the edge of the polygon and satisfy the so-called Kronecker-delta property (see Figure 5.3(c)). Also immediate from above construction is that the Wachspress functions are non-negative and form a partition of unity. More specifically, for $\boldsymbol{\xi} \in \overline{V^n}$

$$N_i(\boldsymbol{\xi}) \geq 0, \qquad \sum_{i=1}^{n} N_i(\boldsymbol{\xi}) = 1 \tag{5.6}$$

[3]By convention, we set $\mathbf{p}_{n+1} = \mathbf{p}_1$ in this expression

One can in fact show for all $\boldsymbol{\xi} \in \overline{V^n}$ (see the appendix of [113]),

$$\boldsymbol{\xi} = \sum_{i=1}^{n} N_i(\boldsymbol{\xi}) \mathbf{p}_i \tag{5.7}$$

Given these properties, the Wachspress functions can be used to construct an isoparametric mapping from V^n to *any* strictly convex[4] n-gon W. If $\{\mathbf{x}_i\}_{i=1}^n$ denotes the location of the vertices of W, ordered counterclockwise, then the isoparametric map $\boldsymbol{\varphi} : \overline{V^n} \to \overline{W}$ defined by

$$\boldsymbol{\varphi}_W(\boldsymbol{\xi}) = \sum_{i=1}^{n} N_i(\boldsymbol{\xi}) \mathbf{x}_i \tag{5.8}$$

is one-to-one and onto [78]. Notice that by virtue of (5.5), $\boldsymbol{\varphi}_W$ maps the vertices and edges of V^n to the corresponding vertices and edges of W. Subsequently one can define the following interpolation functions over W using the isoparametric mapping

$$\tilde{N}_i(\mathbf{x}) = N_i\left(\boldsymbol{\varphi}_W^{-1}(\mathbf{x})\right), \quad \mathbf{x} \in \overline{W},\ i = 1, \ldots, n \tag{5.9}$$

It is straightforward to see that these interpolation functions satisfy the basic requirements for convergence[5] in finite elements [90]. In particular, they are smooth in W and can represent any linear field on W exactly (see Proposition 2 in section 3.3 of [90]). Furthermore, $\tilde{N}_i$ are also linear along the edges of W, which means that the resulting approximation space is C^0-conforming. More precisely, consider a smooth bounded domain $\Omega \subset \mathbb{R}^2$ and its tessellation $\{\Omega_e\}_{e=1}^m$ consisting of strictly convex polygons. Let n_e denote the number of edges of Ω_e and define $P\left(V^n\right) = \operatorname{span}\{N_i\}_{i=1}^n$. Then the finite element space

$$\left\{ u : \overline{\Omega} \to \mathbb{R} : u\big|_{\Omega_e} = \hat{u}_e \circ \boldsymbol{\varphi}_{\Omega_e}^{-1} \text{ for some } \hat{u}_e \in P\left(V^{n_e}\right) \text{ and } e = 1, \ldots, m \right\} \tag{5.10}$$

is contained in $C^0(\overline{\Omega})$. Notice that the local restriction $\hat{u}_e$ is linear on the edges of V^{n_e} and $\boldsymbol{\varphi}_{\Omega_e}$ preserves this linearly along the edges of Ω_e.

From a practical point of view, the calculations are done in the same manner as the usual isoparametric formulations. The interpolations functions are defined on the parent domain, in this case the set of reference n-gons, where the weak form integrals are mapped to and evaluated numerically. In particular, there is no need to construct the interpolation functions (5.9) in the physical domain. In this thesis, we adopt a simple procedure for numerical integration. For $n = 3$ and $n = 4$, we use the standard quadrature rules for triangles and quads and for $n \geq 5$, we divide V^n into n triangles (by connecting the

[4] By "strictly" convex, we mean that no three vertices of the polygon are collinear.

[5] We also refer the reader to the recent paper by Gillette et al. [80] with local error estimates for Wachspress and other barycentric interpolation functions on convex polygons.

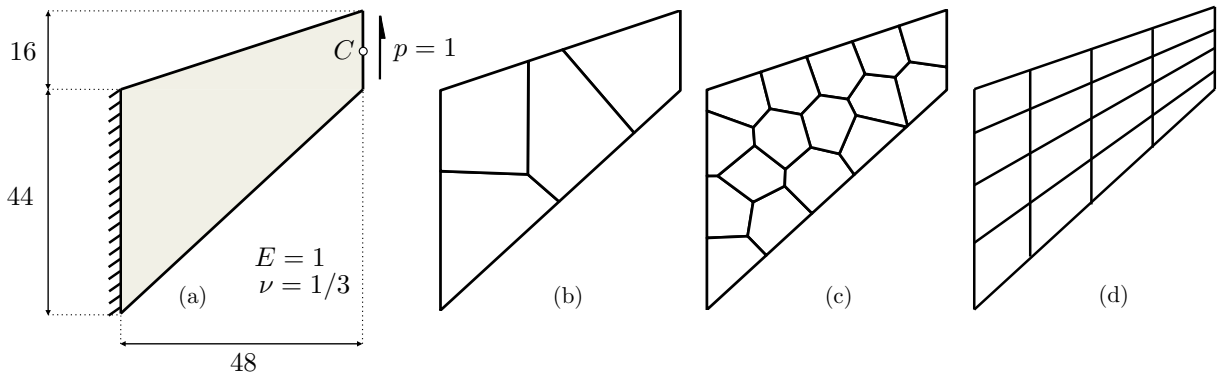


Figure 5.4: The geometry, boundary conditions, and material properties for Cook's problem (b) polygonal mesh with 4 elements (c) polygonal mesh with 16 elements (d) typical quadrilateral mesh with 16 elements (each edge is divided evenly)

centroid to the vertices) and use the well-known quadrature rules on each triangle (see Figure 5.3(b)). However, this integration procedure is not optimal and recently specific quadratic rules with higher accuracy have been developed for polygonal domains (see, for example, [115, 116]) .

5.2 A benchmark problem in elasticity

We assess the performance of the polygonal finite elements using a benchmark problem from linear elasticity. For standard patch tests and convergence studies, we refer the reader to [147, 156] where the formulation is shown to exhibit the expected rates of convergence for linearly precise elements.

Figure 5.4(a) shows the set up for the so-called Cook's problem [54, 180], which consists of a tapered swept panel subjected to uniform shear loading. The quantity of interest is the tip deflection at mid-depth (point C), which was computed on several quadrilateral and polygonal discretizations, some of which are shown in Figure 5.4. Note that the refinement for quadrilateral meshes is progressive (i.e., the finer meshes are embedded in the coarser ones), while the polygonal meshes are constructed independently. The results were compared to the reference value of 23.96, reported in reference [180]. The values for deflection and absolute error are plotted in Figure 5.5 and provided in Table 5.1.

As expected, convergence to the exact solution is monotonic for both types of elements. However, polygonal elements are not as stiff as the quadrilateral elements and produce better results, especially with coarser meshes. It is interesting to observe that even though the polygonal mesh with four elements is made up of three quadrilaterals and only one pentagon, it gives a significantly more accurate deflection value than the corresponding quadrilateral mesh. In fact, the accuracy obtained on this mesh is comparable to that of

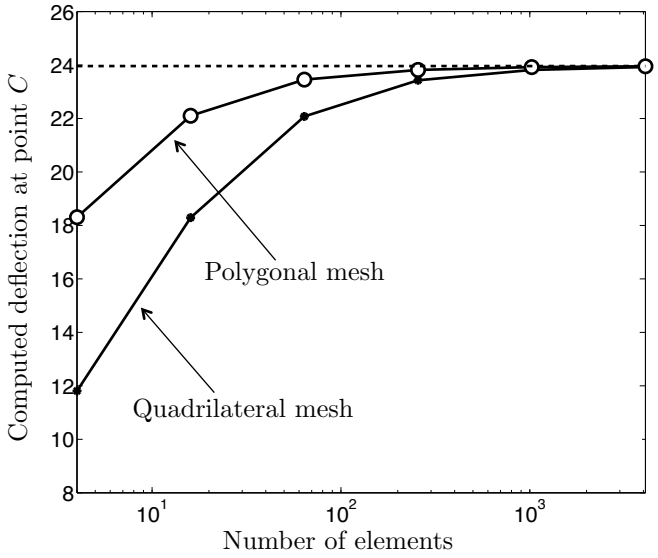


Figure 5.5: Illustration of the convergence of numerical results for Cook's problem

the quadrilateral mesh with 16 elements. Since numerical instabilities such as checkerboard patterns are caused by their artificial stiffness in the finite element approximation, it is reasonable that polygonal elements are not susceptible to such pathologies.

	Polygonal			Quadrilateral		
# elements	# nodes	deflection	% error	# nodes	deflection	% error
4	10	18.3081	23.59	9	11.8090	50.71
16	34	22.1011	7.758	25	18.2902	23.66
64	130	23.4535	2.114	81	22.0781	7.854
256	514	23.8177	0.594	289	23.4303	2.211
1024	2050	23.9235	0.152	1089	23.8176	0.594
4096	8194	23.9527	0.031	4225	23.9245	0.148

Table 5.1: Summary of results for Cook's problem

5.3 Mixed variational problems and stability

In this section, we investigate the stability of polygonal finite elements for mixed variational problems. In particular, we focus on the incompressible Stokes flow problem[6] where we show that a low-order velocity-pressure discretization using polygonal elements leads to a stable finite element formulation. Our interest in the issue of stability for the Stokes

[6]This problem is identical to incompressible elasticity with a reinterpretation of the physical meaning of the variables.

problem is twofold. First, this formulation, featuring piecewise constant pressure fields, is conceptually simple and computationally efficient. In the next section, we show examples of topology optimization problems with Stokes flow as the governing state equation where the design field as well as the velocity and pressure fields are discretized on the same mesh. Moreover, as discussed in the introduction of this chapter, checkerboard instabilities observed in topology optimization are reminiscent of similar instabilities that occur in mixed finite element discretizations of the Stokes problem. This connection is further established in [91] where it is shown that the incremental linearization of the stationarity conditions for the minimum compliance problem leads to a system that has a similar structure to that of the Stokes problem.

Let Ω denote an open, bounded and connected set in $\mathbb{R}^d$ with a Lipschitz continuous boundary. The incompressible Stokes flow problem is given by [63]

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega \\ \operatorname{div}\mathbf{u} &= \mathbf{0} && \text{in } \Omega \\ \mathbf{u} &= \mathbf{0} && \text{on } \partial\Omega \end{aligned} \tag{5.11}$$

where Δ is the vector Laplacian, $\mathbf{u}$ and p are the velocity and pressure fields, respectively, $\mathbf{f} \in L^2(\Omega)^d$ is the applied body force, and $\mu > 0$ is the dynamic viscosity of the fluid. Defining the velocity and pressure spaces

$$\mathcal{V} = H_0^1(\Omega)^d \quad \text{and} \quad \mathcal{Q} = L_0^2(\Omega) := \left\{ q \in L^2(\Omega) : \int_\Omega q \mathrm{d}\mathbf{x} = 0 \right\} \tag{5.12}$$

the mixed variational form of (5.11) consists of finding $(\mathbf{u}, q) \in \mathcal{V} \times \mathcal{Q}$ such that[7]

$$a(\mathbf{u}, \mathbf{v}) + b(p, \mathbf{v}) = \ell(\mathbf{v}), \qquad \forall \mathbf{v} \in \mathcal{V} \tag{5.13}$$

$$b(q, \mathbf{u}) = 0, \qquad \forall q \in \mathcal{Q} \tag{5.14}$$

where

$$a(\mathbf{u}, \mathbf{v}) = \mu \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} \mathrm{d}\mathbf{x}, \quad b(p, \mathbf{v}) = -\int_\Omega p \operatorname{div}\mathbf{u} \mathrm{d}\mathbf{x}, \quad \ell(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \mathrm{d}\mathbf{x} \tag{5.15}$$

A sufficient set of conditions for the well-posedness of the abstract variational problem (5.13)–(5.14) is that bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ and the linear form $\ell(\cdot)$ are continuous,

[7] For the sake of simplicity, we have assumed that only homogenous velocity boundary conditions are imposed in (5.11). However, the same abstract variational problem can be obtained for inhomogenous boundary conditions by "lifting" the boundary conditions and changing variables [73]. The theoretical results also hold when the velocity boundary conditions are imposed on $\Gamma_D \subsetneq \partial\Omega, |\Gamma_D| \neq 0$ since $\|\nabla\mathbf{v}\|_{0,1,\Omega}$ will again define a norm for the velocity space $\{\mathbf{v} \in H^1(\Omega) : \mathbf{v}|_{\Gamma_D} = \mathbf{0}\}$.

$a(\cdot,\cdot)$ is coercive on $\mathcal{V}^0 := \{\mathbf{v} \in \mathcal{V} : b(q,\mathbf{v}) = 0, \forall q \in \mathcal{Q}\}$, i.e.,

$$\exists \alpha > 0 \quad \text{such that} \quad a(\mathbf{v},\mathbf{v}) \geq \alpha \left\|\mathbf{v}\right\|_{\mathcal{V}}^2, \ \forall \mathbf{v} \in \mathcal{V}^0 \tag{5.16}$$

and there exists a constant $\beta > 0$ such that

$$\inf_{q\in\mathcal{Q}} \sup_{\mathbf{v}\in\mathcal{V}} \frac{b(q,\mathbf{v})}{\left\|q\right\|_{\mathcal{Q}} \left\|\mathbf{v}\right\|_{\mathcal{V}}} \geq \beta \tag{5.17}$$

The last condition is known as the Ladyzhenskaya-Babuska-Brezzi (LBB) condition.

In the Stokes problem, the natural norm associated with the velocity space is $\left\|\mathbf{v}\right\|_{\mathcal{V}} = \left\|\nabla\mathbf{v}\right\|_{0,1,\Omega}$[8] and so it can be readily seen that $a(\cdot,\cdot)$ is continuous and coercive on $\mathcal{V}$ (and subsequently on $\mathcal{V}^0$). Continuity of $b(\cdot,\cdot)$ and $\ell(\cdot)$ are also easy to verify. Finally, the condition (5.17) follows from the fact that the divergence operator is a surjection from $H_0^1(\Omega)^d$ onto $L_0^2(\Omega)$ [81].

Considering the finite element subspaces $\mathcal{V}_h \subset \mathcal{V}$ and $\mathcal{Q}_h \subset \mathcal{Q}$, the Galerkin approximation of the above variational problem consists of seeking $(\mathbf{u}_h, p_h) \in \mathcal{V}_h \times \mathcal{Q}_h$ such that

$$a(\mathbf{u}_h,\mathbf{v}_h) + b(p_h,\mathbf{v}_h) = \ell(\mathbf{v}_h), \qquad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{5.18}$$

$$b(q_h,\mathbf{u}_h) = 0, \qquad \forall q_h \in \mathcal{Q}_h \tag{5.19}$$

The approximate problem (5.18)–(5.19) is well-posed if, in addition to the above continuity and coercivity requirements[9],

$$\beta_h := \inf_{q_h\in\mathcal{Q}_h} \sup_{\mathbf{v}_h\in\mathcal{V}_h} \frac{b(q_h,\mathbf{v}_h)}{\left\|q_h\right\|_{\mathcal{Q}} \left\|\mathbf{v}_h\right\|_{\mathcal{V}}} > 0 \tag{5.20}$$

which is nothing but the discrete version of the LBB condition (5.17). Moreover, under these conditions, the finite element solutions pair $(\mathbf{u}_h, p_h)$ satisfies the following error estimates [35, 73]

$$\left\|\mathbf{u}-\mathbf{u}_h\right\|_V \leq \left(1+\frac{M}{\alpha}\right)\left(1+\frac{K}{\beta_h}\right) \inf_{\mathbf{v}_h\in\mathcal{V}_h} \left\|\mathbf{u}-\mathbf{v}_h\right\|_{\mathcal{V}} + \frac{K}{\alpha} \inf_{q_h\in\mathcal{Q}_h} \left\|p-q_h\right\|_{\mathcal{Q}} \tag{5.21}$$

$$\begin{aligned}\left\|p-p_h\right\|_Q \leq\ & \frac{M}{\beta_h}\left(1+\frac{M}{\alpha}\right)\left(1+\frac{K}{\beta_h}\right) \inf_{\mathbf{v}_h\in\mathcal{V}_h} \left\|\mathbf{u}-\mathbf{v}_h\right\|_{\mathcal{V}} \\ & + \left(1+\frac{K}{\beta_h}+\frac{MK}{\alpha\beta_h}\right) \inf_{q_h\in\mathcal{Q}_h} \left\|p-q_h\right\|_{\mathcal{Q}}\end{aligned} \tag{5.22}$$

[8] The norm associated with $\mathcal{Q}$ is of course the L^2-norm $\left\|q\right\|_{\mathcal{Q}} = \left\|q\right\|_{0,1,\Omega}$

[9] Observe that for the Stokes problem $a(\cdot,\cdot)$ is coercive on all of $\mathcal{V}$. Subsequently, it is also coercive on $\mathcal{V}_h^0 := \{\mathbf{v}_h \in \mathcal{V}_h : b(q_h,\mathbf{v}_h) = 0, \forall q \in \mathcal{Q}_h\}$. However, in general, coercivity of $a(\cdot,\cdot)$ on $\mathcal{V}^0$ does not imply its coercivity on $\mathcal{V}_h^0$ since we may have $\mathcal{V}_h^0 \not\subseteq \mathcal{V}^0$. In such cases, the latter must be verified independently for the given spaces $\mathcal{V}_h$ and $\mathcal{Q}_h$.

for some positive constants M and K[10].

With the typical choice of finite element spaces, standard interpolation errors show that the distances $\inf_{\mathbf{v}_h \in \mathcal{V}_h} \|\mathbf{v} - \mathbf{v}_h\|_{\mathcal{V}}$ and $\inf_{q_h \in \mathcal{Q}_h} \|p - q_h\|_{\mathcal{Q}}$ vanish under mesh refinement as $h \to 0$ (see, for example, (4.30)). The above estimates then prove convergence of the finite element solutions *provided that β_h remains bounded away from zero* (meaning that $\beta_h \geq \beta_0$ for some constant $\beta_0 > 0$ and all h). In fact, it is desirable that β_h becomes independent of h so as to achieve an optimal rate of convergence. In this case, the distance between $(\mathbf{u}, p)$ and $(\mathbf{u}_h, p_h)$ is on the order of the distance between $(\mathbf{u}, p)$ and its best approximation in $\mathcal{V}_h \times \mathcal{Q}_h$. Otherwise, if $\beta_h \to 0$ with h, the finite element formulation is said to exhibit *locking*. Intuitively, locking occurs when, given a finite element pressure space $\mathcal{Q}_h$, the velocity space $\mathcal{V}_h$ is not sufficiently rich to both satisfy the weak incompressibility constraint (5.14) and approximate the flow equation (5.13). Mesh refinement does not alleviate the problem since it also enriches the pressure space $\mathcal{Q}_h$. Therefore, it is important to recognize that preventing locking involves the appropriate selection of $\mathcal{V}_h$ with respect to the given choice of pressure discretization.

Aside from locking, the other important issue related to stability of the mixed finite element formulations is the appearance of spurious pressure modes. The pair of spaces $\mathcal{V}_h$ and $\mathcal{Q}_h$ admits a *spurious pressure mode* if there exists $\tilde{p}_h \in \mathcal{Q}_h \backslash \{0\}$ such that

$$b(\tilde{p}_h, \mathbf{v}_h) = 0 \quad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{5.23}$$

If pressure modes are present, then the LBB condition (5.20) cannot be satisfied (the inf-sup quantity is simply zero) and so the finite element problem is not well-posed. Observe that if $(\mathbf{u}_h, p_h)$ is the solution to (5.18)–(5.19), then $(\mathbf{u}_h, p_h + s\tilde{p}_h)$ is also a solution for any $s \in \mathbb{R}$ and a spurious pressure mode $\tilde{p}_h$. Conversely, for finite-dimensional spaces $\mathcal{V}_h$ and $\mathcal{Q}_h$, the violation of the LBB condition implies existence of spurious modes. Even though it is possible in some instances to obtain good approximations to $\mathbf{u}_h$ in presence of spurious modes[11], such formulations are in general deemed unreliable. We also note that the appearance pressure modes is problem-dependent (for the same velocity-pressure element, the pressure mode may or may not exist depending on the boundary conditions of the problem) while locking is intrinsic to the degree of interpolation of velocity and pressure fields.

[10] These constants are in fact the norms associated with the bilinear forms $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$. Notice that in these estimates, we have used the fact that $a(\cdot, \cdot)$ is α-coercive on $\mathcal{V}_h^0$ (see the remark in the previous footnote).

[11] There are certain improved error estimates, most notably for bilinear-velocity constant-pressure element, which show that despite the failure to satisfy the LBB condition (due to both locking and presence of pressure modes), the approximate velocity solution in some cases can be accurate [35].

5.4 Numerical investigation using the inf-sup test

Given a finite element formulation, i.e., a particular construction of space $\mathcal{V}_h$ and $\mathcal{Q}_h$, it is often difficult to establish the satisfaction of the LBB condition and the stability of the formulation as $h \to 0$ (that β_h remains bounded away from zero). In [45], a numerical test is advocated for assessing the stability of FE formulations. By presenting results for various known stable and unstable formulation, this test is shown to be a reliable substitute for analytical verification of these stability criteria. We use this procedure to establish the stability of the proposed finite element formulation based on polygonal discretizations.

Let us first define the space of spurious pressure modes

$$\mathcal{Q}_h^0 = \{q_h \in \mathcal{Q}_h : b(q_h, \mathbf{v}_h) = 0, \forall \mathbf{v}_h \in \mathcal{V}_h\} \tag{5.24}$$

The so-called inf-sup test consists of computing

$$\tilde{\beta}_h := \inf_{q_h \in \left(\mathcal{Q}_h^0\right)^\perp} \sup_{\mathbf{v}_h \in \mathcal{V}_h} \frac{b(q_h, \mathbf{v}_h)}{\|q_h\|_{\mathcal{Q}} \|\mathbf{v}_h\|_{\mathcal{V}}} \tag{5.25}$$

for a sequence of progressively finer discretizations for suitably chosen benchmark problems. If the computed quantity $\tilde{\beta}_h$ approaches zeros with h, then the formulation is expected to be unstable.

Observe that compared to β_h in (5.20), the pressure space $\mathcal{Q}_h$ in the above expression is replaced by the orthogonal complement of the space of pressure modes $(\mathcal{Q}_h^0)^\perp$. If no pressure modes exists, i.e., $\mathcal{Q}_h^0 = \{0\}$, then $(\mathcal{Q}_h^0)^\perp = \mathcal{Q}_h$ and so $\tilde{\beta}_h$ is the same as the inf-sup quantity β_h. Otherwise, we know from previous discussion that β_h and the FE problem is not well-posed. The advantage of working with the restricted space is that, as shown below, it makes the numerical evaluation of (5.25) possible. Moreover, the proposed test is mainly concerned with the convergence of the finite element formulation as the mesh is refined. Note that for a given mesh, one can directly check for the existence of pressure modes by computing the rank of the matrix associated with $b(\cdot, \cdot)$. At any rate, as we shall, the test is also capable of detecting the presence of spurious pressure modes for the considered meshes.

To see how $\tilde{\beta}_h$ can be numerically evaluated, let us denote by Π_h the projection onto $\mathcal{Q}_h$. For any $g \in L^2(\Omega)$, the projection $\Pi_h(g)$ satisfies

$$\int_\Omega q_h \Pi_h(g) \, \mathrm{d}\mathbf{x} = \int_\Omega q_h g \mathrm{d}\mathbf{x}, \quad \forall q_h \in \mathcal{Q}_h \tag{5.26}$$

With this definition, it is straightforward to show that[12]

[12]Note that for any $q_h \in \mathcal{Q}_h^0$, we have $\int_\Omega q_h \Pi_h(\operatorname{div} \mathbf{w}_h) \, \mathrm{d}\mathbf{x} = \int_\Omega q_h \operatorname{div} \mathbf{w}_h \mathrm{d}\mathbf{x} = 0$ for all $\mathbf{w}_h \in \mathcal{V}_h$

$$\left(\mathcal{Q}_h^0\right)^{\perp} = \{\Pi_h\left(\operatorname{div}\mathbf{w}_h\right) : \mathbf{w}_h \in \mathcal{V}_h\} \tag{5.27}$$

Therefore, we can rewrite (5.25) alternatively as

$$\tilde{\beta}_h = \inf_{\mathbf{w}_h\in\mathcal{V}_h}\sup_{\mathbf{v}_h\in\mathcal{V}_h}\frac{b\left(\Pi_h\left(\operatorname{div}\mathbf{w}_h\right),\mathbf{v}_h\right)}{\left\|\Pi_h\left(\operatorname{div}\mathbf{w}_h\right)\right\|_{\mathcal{Q}}\left\|\mathbf{v}_h\right\|_{\mathcal{V}}} \tag{5.28}$$

Moreover, from (5.26), we have

$$b\left(\Pi_h\left(\operatorname{div}\mathbf{w}_h\right),\mathbf{v}_h\right) = \int_\Omega \Pi_h\left(\operatorname{div}\mathbf{w}_h\right)\operatorname{div}\mathbf{v}_h\mathrm{d}\mathbf{x} = \int_\Omega \Pi_h\left(\operatorname{div}\mathbf{w}_h\right)\Pi_h\left(\operatorname{div}\mathbf{v}_h\right)\mathrm{d}\mathbf{x} \tag{5.29}$$

which gives the symmetric expression

$$\tilde{\beta}_h = \inf_{\mathbf{w}_h\in\mathcal{V}_h}\sup_{\mathbf{v}_h\in\mathcal{V}_h}\frac{\int_\Omega \Pi_h\left(\operatorname{div}\mathbf{w}_h\right)\Pi_h\left(\operatorname{div}\mathbf{v}_h\right)\mathrm{d}\mathbf{x}}{\left\|\Pi_h\left(\operatorname{div}\mathbf{w}_h\right)\right\|_{\mathcal{Q}}\left\|\mathbf{v}_h\right\|_{\mathcal{V}}} \tag{5.30}$$

Let us denote by $\{\mathbf{N}_i\}_{i=1}^{n_\mathbf{u}}$ the set of basis functions for the velocity space $\mathcal{V}_h$ and define the following matrices associated with the terms in (5.30):

$$[\mathbf{S}_h]_{ij} = \int_\Omega \nabla\mathbf{N}_i : \nabla\mathbf{N}_j\mathrm{d}\mathbf{x}, \quad [\mathbf{G}_h]_{ij} = \int_\Omega \Pi_h\left(\operatorname{div}\mathbf{N}_i\right)\Pi_h\left(\operatorname{div}\mathbf{N}_j\right)\mathrm{d}\mathbf{x} \tag{5.31}$$

Observe that $\mathbf{S}_h$ is positive definite and $\mathbf{G}_h$ is positive semi-definite. We have the following relation for $\tilde{\beta}_h$

$$\tilde{\beta}_h = \inf_{\mathbf{W}\in\mathbb{R}^{n_\mathbf{u}}}\sup_{\mathbf{V}\in\mathbb{R}^{n_\mathbf{u}}}\frac{\mathbf{W}^T\mathbf{G}_h\mathbf{V}}{\left(\mathbf{W}^T\mathbf{G}_h\mathbf{W}\right)^{1/2}\left(\mathbf{V}^T\mathbf{S}_h\mathbf{V}\right)^{1/2}} \tag{5.32}$$

From this and after some algebra, one can show that

$$\tilde{\beta}_h = \sqrt{\lambda_k} \tag{5.33}$$

where λ_k is the smallest nonzero eigenvalue for the following eigenvalue problem

$$\mathbf{G}_h\mathbf{V} = \lambda\mathbf{S}_h\mathbf{V} \tag{5.34}$$

From a practical point of view, while the calculation of $\mathbf{S}_h$ is straightforward, computing $\mathbf{G}_h$ involves projection onto the pressure space and is more involved. In the case of piecewise constant pressure space, we can derive a simple expression for $\mathbf{G}_h$. Let $\{\Omega_e\}_{e=1}^{n_p}$ denote a tessellation of the domain Ω. The basis for the piecewise constant pressure space $\mathcal{Q}_h$ is given by $\{\chi_{\Omega_e}\}_{e=1}^{n_p}$ and so we have the following simple expression for the projection operator

$$\Pi_h(g) = \sum_{e=1}^{n_p} \frac{1}{|\Omega_e|} \left(\int_{\Omega_e} g \mathrm{d}\mathbf{x} \right) \chi_{\Omega_e} \tag{5.35}$$

In turn, this yields

$$[\mathbf{G}_h]_{ij} = \sum_{e=1}^{n_p} \frac{1}{|\Omega_e|} \left(\int_{\Omega_e} \operatorname{div} \mathbf{N}_i \mathrm{d}\mathbf{x} \right) \left(\int_{\Omega_e} \operatorname{div} \mathbf{N}_j \mathrm{d}\mathbf{x} \right) \tag{5.36}$$

As mentioned before, the number of spurious pressure modes can also obtained from the eigenvalue problem (5.34). If there are $k-1$ zero eigenvalues, then there are precisely $\max\left(k-1-(n_{\mathbf{u}}-n_p), 0\right)$ spurious pressure modes possible. We note that $\tilde{\beta}_h > 0$ in this case even though $\beta_h = 0$ and the LBB condition is not satisfied.

We consider two benchmark problems both posed on the unit square $\Omega = [0,1]^2$. In the first example, as in (5.11), the homogenous velocity boundary conditions are imposed on the entire boundary $\partial\Omega$. The second example is borrowed from [45] where the velocity at $\mathbf{x} = (0,0)$ and its horizontal component along the left edge is set to zero. The velocity field is discretized by means the isoparametric Wachpress interpolation functions presented in the last section (cf. (5.10)) and an element-wise constant pressure field is used. We consider three types of meshes. First is the regular square grid for which the velocity element is the usual bilinear square element. The second is uniform hexagonal tessellations similar to the one shown in Figure 5.1. The last one is an unstructured mesh consisting of convex polygons generated using the Voronoi algorithm in chapter 6. For each example and mesh type, the quantity $\tilde{\beta}_h$ was computed on four or five progressively finer meshes. The results are plotted in Figure 5.6. As expected, for the bilinear quads, $\tilde{\beta}_h$ decays with mesh refinement. However, this value is nearly constant, beyond a certain level of refinement, for both types of polygonal meshes. Moreover, two checkerboard pressure modes were detected for every square grid in the first example. Owing to the nature of boundary conditions, no pressure modes were permitted for the same meshes in the second example. Both uniform hexagonal and unstructured polygonal meshes were free of any spurious pressure modes for both problems[13].

5.5 Topology optimization for fluid flow

In this section, we will discuss the formulation of topology optimization in Stokes flow problems. As usual, let $\Omega \subseteq \mathbb{R}^d$ denote the extended design domain and $\mathbf{g}$ be a given smooth velocity field defined on $\partial\Omega$ such that $\int_\Omega \mathbf{g} \cdot \mathbf{n} \mathrm{d}s = 0$. Moreover, let Γ_g denote the portion of the boundary where $\mathbf{g}$ is nonzero. We consider $\omega \subseteq \Omega$ to be an admissible shape

[13]The only exception was the polygonal mesh with 4 elements which, as a result of CVT iterations, consisted of four quads!

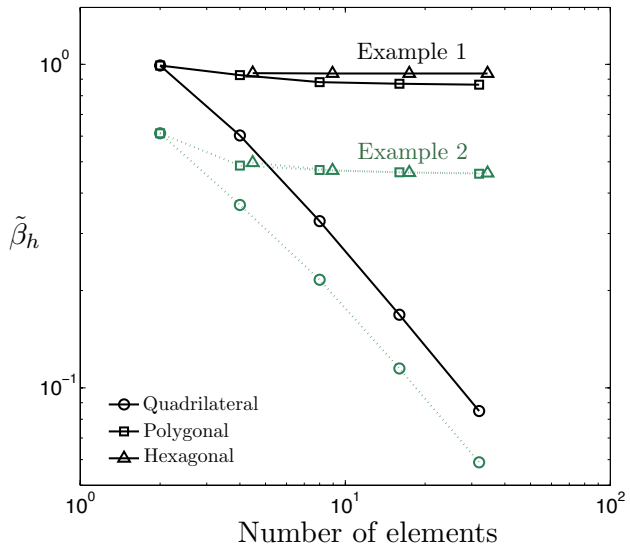


Figure 5.6: Results of the inf-sup test for the example problems

if $\partial\omega \supseteq \Gamma_g$ and it is sufficiently smooth so that the incompressible Stokes equations

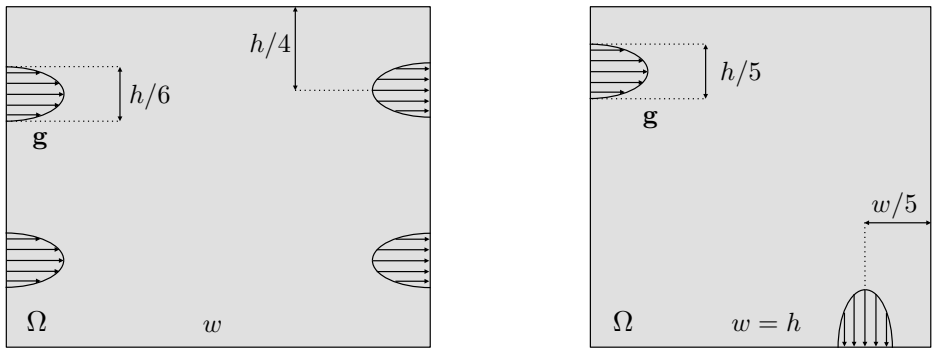
$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \mathbf{0} \quad && \text{in } \omega \\ \operatorname{div}\mathbf{u} &= 0 \quad && \text{in } \omega \\ \mathbf{u} &= \mathbf{g} \quad && \text{on } \partial\omega \cap \partial\Omega \\ \mathbf{u} &= \mathbf{0} \quad && \text{on } \partial\omega \backslash \partial\Omega \end{aligned} \tag{5.37}$$

admit a unique and stable solution $(\mathbf{u}_\omega, p_\omega)$. A benchmark optimal shape problem consists of finding an admissible ω that minimizes the dissipated power subject to a constraint on its volume. In particular we seek to minimize the following cost functional:

$$J(\omega) = \frac{1}{2}\int_\omega \mu\nabla\mathbf{u}_\omega : \nabla\mathbf{u}_\omega \mathrm{d}\mathbf{x} + \lambda\,|\omega| \tag{5.38}$$

It can be shown that minimizing the first term amounts to minimizing the average pressure drop between the inlet and the outlet. The second term, with $\lambda > 0$, is of course a penalty on the volume of the shape.

Similar to our treatment of optimal shape problems in elasticity, we follow a two-step procedure to reformulate this problem into one of sizing optimization with the control field defined on the entire extended domain Ω. The first step consists of approximating the governing state equation with a boundary value problem defined on Ω in analogous manner to the Ersatz approach. For a fixed shape ω, one can show that the solution $(\mathbf{u}_\omega^\varepsilon, p_\omega^\varepsilon)$ to the generalized Stokes problem given by



(a) (b)

Figure 5.7: The domain and boundary conditions for the benchmark optimal flow problems (a) double pipe (b) pipe bend. For both examples, the boundary velocity $\mathbf{g}$ on each portion of Γ_g has parabolic profile as indicated in the figures.

$$\begin{aligned} -\mu\Delta\mathbf{u} + \frac{1}{\varepsilon}(1-\chi_\omega)\,\mathbf{u} + \nabla p = \mathbf{0} \qquad & \text{in } \Omega \\ \operatorname{div}\mathbf{u} = 0 \qquad & \text{in } \Omega \\ \mathbf{u} = \mathbf{g} \qquad & \text{on } \partial\Omega \end{aligned} \tag{5.39}$$

when restricted to ω, is a good approximation to $(\mathbf{u}_\omega, p_\omega)$ for sufficiently small $\varepsilon > 0$ [76]. Intuitively, the new dissipative term in this generalized Stokes system forces $\mathbf{u}_\omega^\varepsilon$ to vanish in $\Omega\backslash\omega$ when the "permeability" coefficient ε is close to zero. In contrast to the Ersatz approximation in elasticity, observe that the viscous term with the highest order derivatives is left intact in this approximation.

As a surrogate to the minimization of (5.38), we fix $0 < \varepsilon \ll 1$ and consider the following optimization problem

$$\inf_{\chi\in L^\infty(\Omega;\{0,1\})} J(\chi) = \frac{1}{2}\int_\Omega \left[\mu\nabla\mathbf{u}_\chi : \nabla\mathbf{u}_\chi + \frac{1}{\varepsilon}(1-\chi)\,\mathbf{u}_\chi\cdot\mathbf{u}_\chi\right]\mathrm{d}\mathbf{x} + \lambda\int_\Omega \chi\mathrm{d}\mathbf{x} \tag{5.40}$$

where $\mathbf{u}_\chi$ is the solution to (5.39) with $\chi_\omega = \chi$. Few remarks are in order regarding the optimization problem (5.40). Unlike the related compliance minimization problem in elasticity, (5.40) is well-posed in that it admits minimizers in $L^\infty(\Omega;\{0,1\})$ [76]. Therefore, there is no need to impose additional regularity conditions on the characteristic functions. This is in part a consequence of the fact that the design field appears in the coefficient of the lower order term in the governing state equation (5.39). On physical grounds, the favorability of highly oscillatory shapes in the compliance problem is related to the stiffening effect of fine mixtures (cf. counterexample in section 2.3). By contrast, such arrangements in the fluid flow problem lead to larger viscous dissipation and are naturally excluded in the optimal regime.

Moreover, we note that the mere fact that (5.39) is an approximation to (5.37) by itself

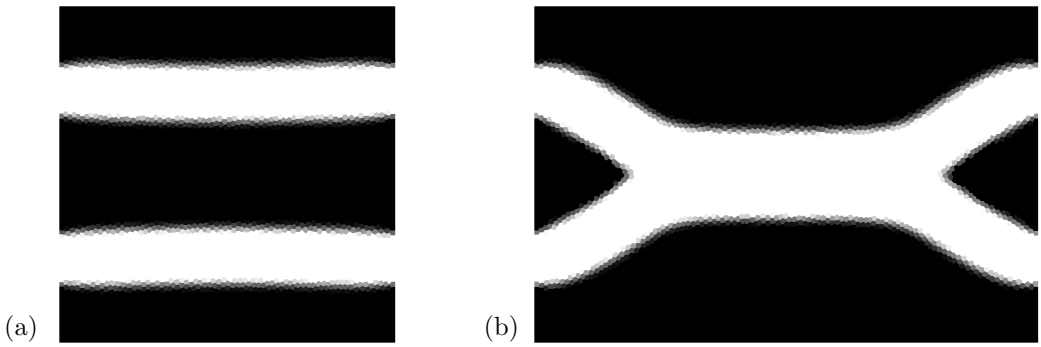


Figure 5.8: Double pipe solutions for (a) $w = h$ (b) $w = 1.5h$ where w is the width of the domain and h is its height. The prescribed volume fraction for this problem is $\overline{v} = 1/3$

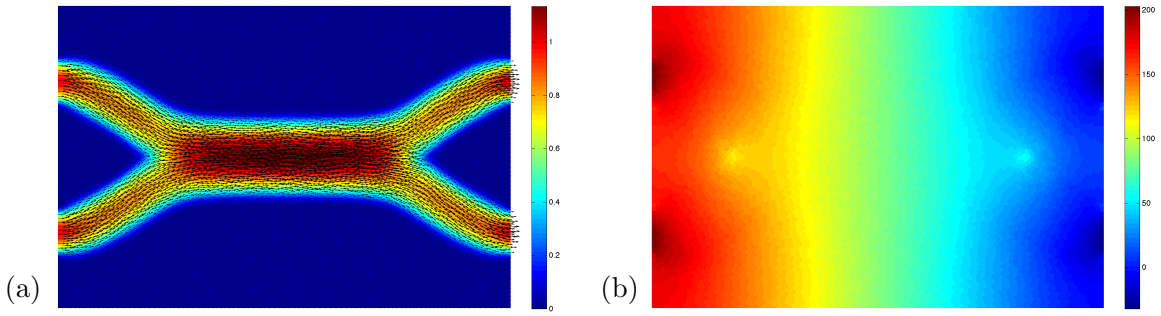


Figure 5.9: (a) Velocity field (magnitude plotted in the background) (b) pressure field for the double pipe solution with $w = 1.5h$

does not justify the approximation of (5.38) by (5.40). Just as in the case of elasticity, the space of admissible shapes associated with the latter problem is significantly larger than that of the former. Indeed, one needs a result similar to what is presented in section 2.10 for the Ersatz approximation.

The second step in the deriving a sizing optimization problem is to replace the space of characteristic functions $L^\infty(\Omega; \{0, 1\})$ by the larger space $L^\infty(\Omega; [0, 1])$. Interestingly, this relaxation does not require any additional modification of the problem (e.g., introducing penalization of intermediate fields) since one can show that there exists a characteristic function minimizing J in the large space $L^\infty(\Omega; [0, 1])$ [30, 76]. This result implies that enlarging the space of admissible designs does not change the value of the optimization problem even though it makes the problem amenable to gradient-based optimization algorithms.

The final sizing optimization problem is given by

$$\min_{\rho\in L^\infty(\Omega;[0,1])} \frac{1}{2}\int_\Omega \left[\mu\nabla\mathbf{u}_\rho : \nabla\mathbf{u}_\rho + \kappa(\rho)\mathbf{u}_\rho\cdot\mathbf{u}_\rho\right]\mathrm{d}\mathbf{x} + \lambda\int_\Omega \rho\mathrm{d}\mathbf{x} \tag{5.41}$$

where $\mathbf{u}_\rho$ is a solution to

$$\begin{aligned} -\mu\Delta\mathbf{u} + \kappa(\rho)\mathbf{u} + \nabla p &= \mathbf{0} \quad && \text{in } \Omega \\ \operatorname{div}\mathbf{u} &= 0 && \text{in } \Omega \\ \mathbf{u} &= \mathbf{g} && \text{on } \partial\Omega \end{aligned} \tag{5.42}$$

and the inverse permeability coefficient $\kappa(\rho) = \varepsilon^{-1}(1-\rho)$.

It is interesting to note that Borrvall and Petersson [30] arrive at the same sizing problem in two dimensions with

$$\kappa(\rho) = \frac{5\mu}{2\rho^2} \tag{5.43}$$

by considering the three-dimensional Stokes problem under plane flow assumptions wherein ρ represent the distance between two encapsulating plates. The dissipative term $\kappa(\rho)$ in that setting is due to the out-of-plane shear effects. Naturally, the fluid movement is restricted in the regions where ρ is close to zero. In this context $\int_\Omega \rho\mathrm{d}\mathbf{x}$ is the total volume of the fluid in the domain, and so their derivation gives some physical meaning to the design field ρ. The authors, however, recognize that the sizing formulation can also be used in three dimensional problems and with interpolation functions for $\kappa(\rho)$ that are more suited for numerical calculations. In fact, they advocate the use of

$$\kappa(\rho) = \frac{1}{\varepsilon}\frac{q\,(1-\rho)}{q+\rho} \tag{5.44}$$

along with a continuation procedure consisting of gradually increasing q. It is easy to see that this rational function approaches the linear function in (5.42) as $q \to \infty$. Also, referring to (5.43), ε is set to $4\times 10^{-5}/\mu$ which corresponds to minimum distance of 0.01 between the encapsulating plates. We note that the authors in [30] also use a small positive lower bound on $\kappa(\rho)$ (at $\rho = 1$) but it is not clear why this is needed either from a theoretical or a computational point of view.

5.6 Numerical results

We consider two benchmark optimal flow problems from [30]. For all the numerical results, the extended domain is discretized using unstructured convex polygonal meshes. As in the last section, the velocity field is defined using the isoparametric Wachpress functions while the pressure field is piecewise constant. A piecewise constant discretization is also used for the design space similar to section 1.4. Following [30], the inverse permeability is given

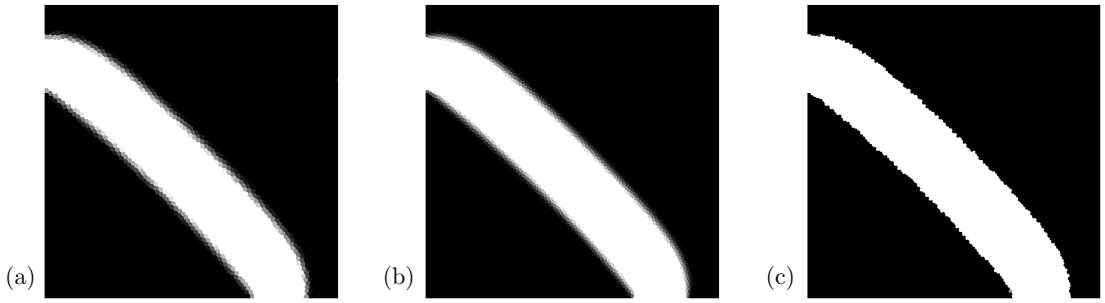


Figure 5.10: Pipe bend solutions with (a) 5,000 elements (b) 10,000 elements (c) same as part (b) but with the final value of $q = 10$ for the penalty parameter in $\kappa(\rho)$. The prescribed volume fraction for this problem is $\overline{v} = 0.08\pi$

by (5.44) and q is gradually increased from initial value of 0.01 to terminal value of 0.1. The resulting optimization algorithm, with the volume constraint explicitly enforced for the prescribed volume fraction $\overline{v}$, is solved using the Method of Moving Asymptotes [153].

The first example is the double pipe problem which consists of two velocity inlets and outlets as shown in Figure 5.7(a). The flow velocity profile is parabolic and orthogonal to the domain boundary in each case. In particular, we have $\mathbf{g} = \mathbf{0}$ except for the locations indicated in the figure and $\mathbf{g} = g\mathbf{n}$ on Γ_g where g is parabolic with unit amplitude. Depending on the aspect ratio of Ω, the optimal shape consists of two separate pipes (cf. Figure 5.8(a)) or one connected pipe transporting the fluid. Note that design field ρ is plotted in greyscale where $\rho = 0$ is shown in black (corresponding to the solid region where flow is restricted) and $\rho = 1$ is indicated in white. We can see from Figure 5.8(b) that for the wider domain, a single wider pipe is more advantageous even though it requires the fluid to travel a longer path. By contrast, in the square domain, the optimal design consists of two straight pipes. Of course, due to the unstructured nature of the polygonal mesh, the optimal solution is not completely straight or symmetric.

The pressure and velocity fields for the single pipe solution are plotted in Figure 5.9. We can see that owing to the small permeability coefficient, the flow velocity is small in the region where $\rho = 0$ and the zero flow condition along the boundary of pipe is well represented. Morever, the pressure field is smooth and free of any spurious modes.

The next example is the pipe bend problem whose domain and boundary conditions are shown in Figure 5.7(b). While one might expect a quarter torus as the optimal shape, the solution shown in Figure 5.10 is an almost straight pipe connecting the inlet and the outlet, which, as explained in [30], is reasonable for the fluid modeled by the Stokes flow. We also solve the problem on a finer mesh to exhibit the convergence of the design field without the need for any regularization. Both solutions are binary and nearly represent the same shape. A proof for the convergence of finite element solutions can be found in [30] and is a

consequence of the fact that weak convergence of the design field is sufficient to establish convergence (on a subsequence) of the associated velocity fields. The same mesh used for Figure 5.10(b) was used to investigate the effects of the penalty parameter q. The solution shown in Figure 5.10(c) is generated using the terminal value of $q = 10$. As predicted by the theory, the final sizing function is almost completely binary though the outline of corresponding shape is similar to that of Figure 5.10(b).

Chapter 6

Polygonal Mesh Generation Using Voronoi Diagrams

In this chapter, we discuss a mesh generation algorithm based on the concept of Voronoi diagrams that can be used to discretize arbitrary two-dimensional domains using convex polygons. The main ingredients of the proposed mesh generator are the implicit representation of the domain and the use of Centroidal Voronoi diagrams for its discretization. The signed distance function contains all the essential information about the meshing domain needed in this algorithm. As pointed out in [125], this implicit description provides great flexibility to construct a relatively large class of domains with algebraic expressions. A discretization of the domain is constructed from a Centroidal Voronoi tessellation (CVT) that incorporates an approximation to its boundary. The approximation is obtained by including the set of reflections of the seeds associated with the Voronoi tessellations [27, 179]. The Lloyd's method is used to establish a uniform (optimal) distribution of seeds and thus a high quality mesh [162]. We remark that CVTs have been previously used for generation and analysis of triangular discretizations (see, for example, [66, 67, 93]) and, in some cases, superconvergence of numerical solutions has been observed [89].

6.1 Distance functions and implicit geometries

Let Ω be a subset of $\mathbb{R}^2$ with smooth boundary. The *signed distance function* associated with Ω is the mapping $d_\Omega : \mathbb{R}^2 \to \mathbb{R}$ defined by:

$$d_\Omega(\mathbf{x}) = s_\Omega(\mathbf{x}) \min_{\mathbf{y}\in\partial\Omega} \|\mathbf{x}-\mathbf{y}\| \tag{6.1}$$

where $\partial\Omega$ denotes the boundary of Ω, $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^2$ (so $\|\mathbf{x}-\mathbf{y}\|$ here is the distance between $\mathbf{x}$ and point $\mathbf{y}$ on the boundary of the domain), and the sign function is given by:

$$s_\Omega(\mathbf{x}) = 1 - 2\chi_\Omega(\mathbf{x}) = \begin{cases} -1, & \mathbf{x}\in\Omega \\ 1, & \mathbf{x}\in\mathbb{R}^2\backslash\Omega \end{cases} \tag{6.2}$$

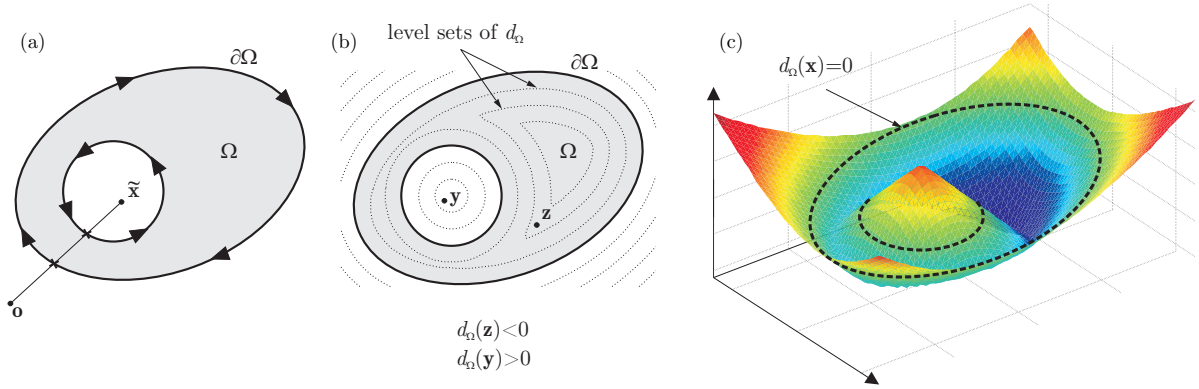


Figure 6.1: (a) Explicit parametrization of domain boundary: the ray connecting point $\tilde{\mathbf{x}}$ to point $\mathbf{o}$, known to lie outside the domain, intersects $\partial\Omega$ an even number of times, indicating $\tilde{\mathbf{x}} \notin \Omega$ (b) Implicit representation of the domain: the sign of the distance function $d_\Omega(\mathbf{x})$ determines if $\mathbf{x}$ lies inside the domain (c) Surface plot of the signed distance function: note that $\partial\Omega$ is given by the zero level set of d_Ω

Thus, if $\mathbf{x}$ lies inside the domain Ω, $d_\Omega(\mathbf{x})$ is minus the distance of $\mathbf{x}$ to the closest boundary point. The following characterizations are immediate from this definition:

$$\overline{\Omega} = \left\{\mathbf{x} \in \mathbb{R}^2 : d_\Omega(\mathbf{x}) \leq 0\right\}, \quad \partial\Omega = \left\{\mathbf{x} \in \mathbb{R}^2 : d_\Omega(\mathbf{x}) = 0\right\} \tag{6.3}$$

In the proposed meshing algorithm, we need to determine if a candidate point (seed) $\mathbf{x}$ lies in the interior of domain Ω. With an explicit representation of Ω, based on parametrization of its boundary, this may be difficult as it requires counting the number of times a ray connecting $\mathbf{x}$ and some exterior point intersects the boundary [120]. Given the signed distance function, we recover this information by evaluating the sign of $d_\Omega(\mathbf{x})$ (see Figure 6.1).

Other useful information about the domain geometry is provided by the signed distance function. Its gradient, ∇d_Ω, gives the *direction* to the nearest boundary point. If Ω has smooth boundary and $\mathbf{x} \in \partial\Omega$, then $\nabla d_\Omega(\mathbf{x})$ is the unit vector normal to the boundary. In general, for almost every point $\mathbf{x} \in \mathbb{R}^2$, we have:

$$\|\nabla d_\Omega(\mathbf{x})\| = 1 \tag{6.4}$$

It is possible that the distance function exhibits kinks even when $\partial\Omega$ is smooth. In particular, if $\mathbf{x}$ is equidistant to more than one point of $\partial\Omega$, then $\nabla d_\Omega(\mathbf{x})$ fails to exist. As illustrated in Figure 6.2(b), the variation of the distance function changes depending on which boundary point is approached. In such a case, numerical differentiation may result in $\|\nabla d_\Omega(\mathbf{x})\| \neq 1$.

In the proposed algorithm, we use the property of the gradient to find the reflection of

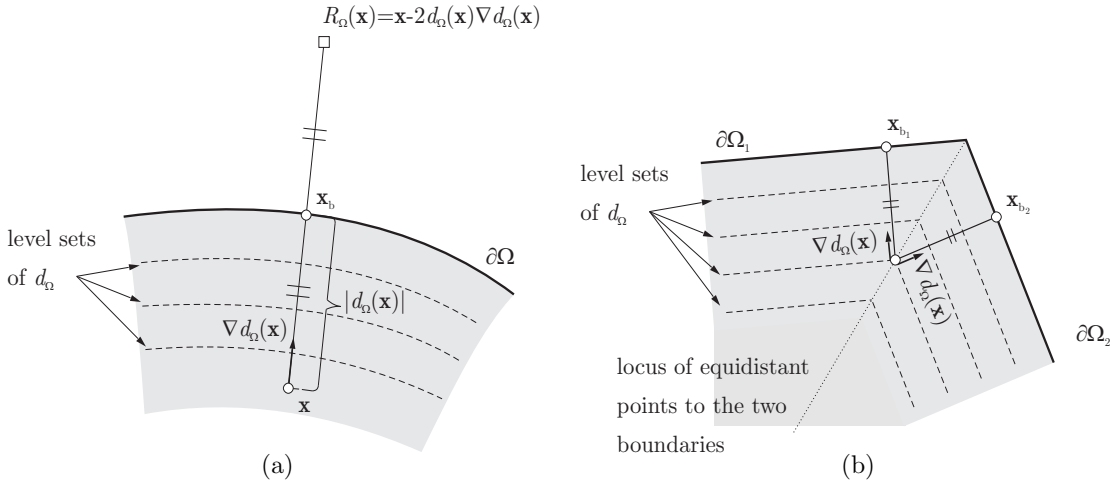


Figure 6.2: (a) For $\mathbf{x} \in \mathbb{R}^2$, the direction to the closest boundary point, $\mathbf{x}_b$, is given by $\nabla d_\Omega(\mathbf{x})$, which can be used to compute the reflection $R_\Omega(\mathbf{x})$ (b) The distance function exhibit kinks at points that are equidistant to more than one boundary point. Here $\nabla d_\Omega(\mathbf{x})$ denotes the one-sided gradient at such a point $\mathbf{x}$

$\mathbf{x}$ about the closest boundary point. Denoting the reflection by $R_\Omega(\mathbf{x})$, we have (cf. Figure 6.2(a)):

$$R_\Omega(\mathbf{x}) = \mathbf{x} - 2d_\Omega(\mathbf{x})\nabla d_\Omega(\mathbf{x}) \tag{6.5}$$

Note that this expression is valid for both interior and exterior points, i.e., for any $\mathbf{x} \in \mathbb{R}^2$.

We can see from the discussion so far that when Ω is characterized by its signed distance function, a great deal of useful information about Ω can be readily extracted. An essential task then is to construct d_Ω for a given domain Ω that we wish to discretize. For many simple geometries, the signed distance function can be readily identified. For example, if Ω is a circle of radius r centered at point $\mathbf{x}_o$, its distance function is given by:

$$d_\Omega(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_o\| - r \tag{6.6}$$

Moreover, set operations such as union, intersection, and complementation can be used to piece together and combine different geometries. Given domains Ω_1 and Ω_2, the expressions

$$\begin{aligned} d_{\Omega_1 \cup \Omega_2}(\mathbf{x}) &= \min\left(d_{\Omega_1}(\mathbf{x}), d_{\Omega_2}(\mathbf{x})\right) \\ d_{\Omega_1 \cap \Omega_2}(\mathbf{x}) &= \max\left(d_{\Omega_1}(\mathbf{x}), d_{\Omega_2}(\mathbf{x})\right) \\ d_{\mathbb{R}^2 \setminus \Omega_1}(\mathbf{x}) &= -d_{\Omega_1}(\mathbf{x}) \end{aligned} \tag{6.7}$$

capture the "sign" property of the distance function for the combined geometry, as illustrated in Figure 6.3. However, we note that the "distance" property may not necessarily hold

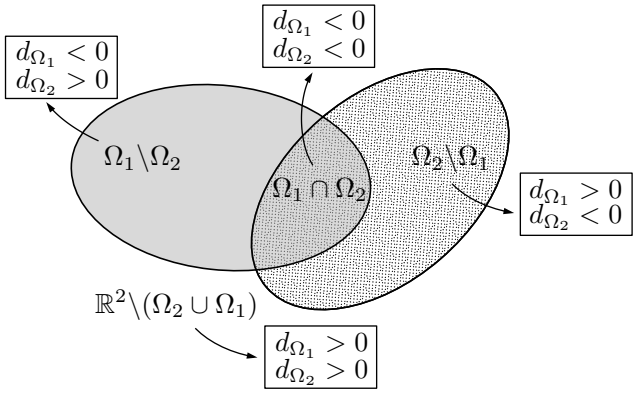


Figure 6.3: Correspondence between set operations and the sign of distance functions

everywhere[1]. A reference commonly cited in conjunction with these equations is the work of [132] but there the implicit functions do not carry the distance property. In our meshing algorithm, we require access to the distance functions of the constituent domains, in part, to address this issue.

Transformations such as rotation and translation can also be incorporated to obtain desired geometries. For example, if $\mathbf{T}_\theta$ is the matrix for rotation by angle θ about the origin, the signed distance function for rotated domain Ω_θ is given by:

$$d_{\Omega_\theta}(\mathbf{x}) = d_\Omega(\mathbf{T}_\theta^{-1}\mathbf{x}) \tag{6.8}$$

Also, a signed distance function can be obtained from the level sets of a given implicit function by solving a nonlinear system of equations (cf. [125]) or more generally using marching algorithms (see, for example, [151, 138, 120, 182]). We will revisit the issue of computing distance functions in relation to our meshing algorithm and later through examples.

6.2 Voronoi diagrams, CVTs, and Lloyd's algorithm

The concept of Voronoi diagrams plays a central role in the proposed algorithm. Given a set of n distinct points or *seeds* $\mathbf{P}$, the *Voronoi tessellation*[2] of the domain $\Delta \subseteq \mathbb{R}^2$ is defined by:

$$\mathcal{T}(\mathbf{P}; \Delta) = \{V_\mathbf{y} \cap \Delta : \mathbf{y} \in \mathbf{P}\} \tag{6.9}$$

[1]For example, consider $\Omega_1 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 < 0\}$ and $\Omega_2 = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 < 0\}$. The formula $d_{\Omega_1 \cup \Omega_2}(\mathbf{x}) = \min(d_{\Omega_1}(\mathbf{x}), d_{\Omega_2}(\mathbf{x}))$ has incorrect distance "value" in the third quadrant, i.e., for $x_1 < 0$, $x_2 < 0$. In this region, the closest boundary point is the *new* corner $\mathbf{x} = (0, 0)$ formed by the union operation.

[2]A *tessellation* or *tiling* of Δ is a collection of open sets S_i such that $\cup_i \overline{S}_i = \overline{\Delta}$ and $S_i \cap S_j = \emptyset$ if $i \neq j$.

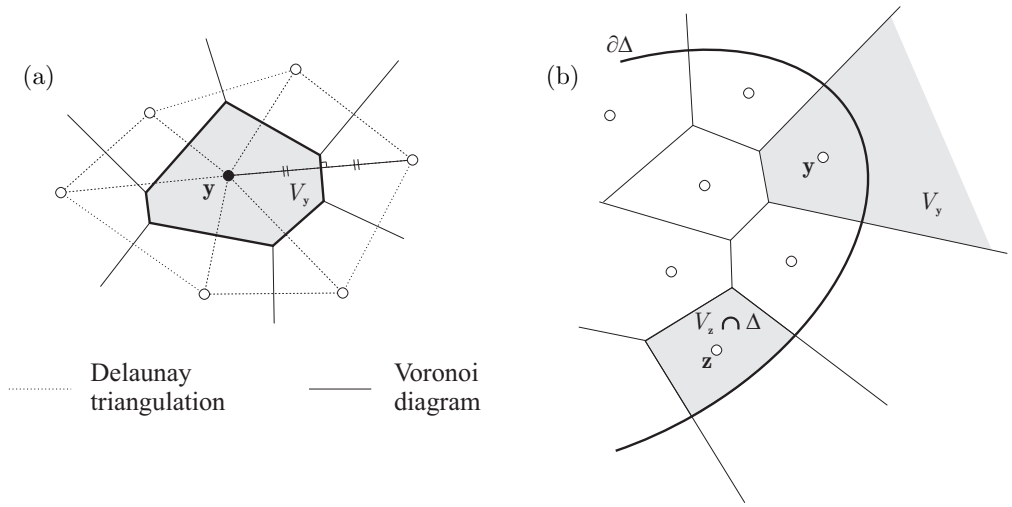


Figure 6.4: (a) Voronoi diagram and its dual, the Delaunay triangulation (b) Illustrating the difference between $V_{\mathbf{y}}$, defined in Equation (6.10) as the Voronoi cell, and $V_{\mathbf{y}} \cap \Delta$, as the regions making up the Voronoi tessellation of Δ (cf. Equation 6.9)

where $V_{\mathbf{y}}$ is the *Voronoi cell* associated with point $\mathbf{y}$:

$$V_{\mathbf{y}} = \left\{ \mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x} - \mathbf{y}\| < \|\mathbf{x} - \mathbf{z}\|, \forall \mathbf{z} \in \mathbf{P} \backslash \{\mathbf{y}\} \right\} \tag{6.10}$$

Therefore $V_{\mathbf{y}}$ consists of points in the plane closer to $\mathbf{y}$ than any other point in $\mathbf{P}$. To simplify the notation for the meshing algorithm, we are defining the cells over the entire $\mathbb{R}^2$, while it is common in the literature to define $V_{\mathbf{y}} \cap \Delta$ to be the Voronoi cell (see Figure 6.4).

The properties of Voronoi diagrams have been studied extensively and we refer the reader to review paper [15] on the topic. One relevant property in two dimensions is that if a Voronoi cell is bounded, it is necessarily a convex polygon since it is formed by finite intersection of half-planes (each of which is a convex set). Hence, as we shall see in the next section, the meshing algorithm produces discretizations consisting only of *convex* polygons. This is pertinent to the isoparametric formulation for polygonal finite elements which requires convexity of all the elements in the mesh [149, 148, 162].

The regularity of Voronoi diagrams is determined entirely by the distribution of the generating point set. A random or quasi-random set of generators may lead to a discretization not suitable for use in finite element analysis. Therefore, we restrict our attention to a special class of Voronoi tessellations that enjoy a higher level of regularity. A Voronoi tessellation $\mathcal{T}(\mathbf{P}; \Delta)$ is *centroidal* if for every $\mathbf{y} \in \mathbf{P}$:

$$\mathbf{y} = \mathbf{y}_c \qquad \text{where} \qquad \mathbf{y}_c := \frac{\int_{V_{\mathbf{y}} \cap \Delta} \mathbf{x} \mu(\mathbf{x}) \mathrm{d}\mathbf{x}}{\int_{V_{\mathbf{y}} \cap \Delta} \mu(\mathbf{x}) \mathrm{d}\mathbf{x}} \tag{6.11}$$

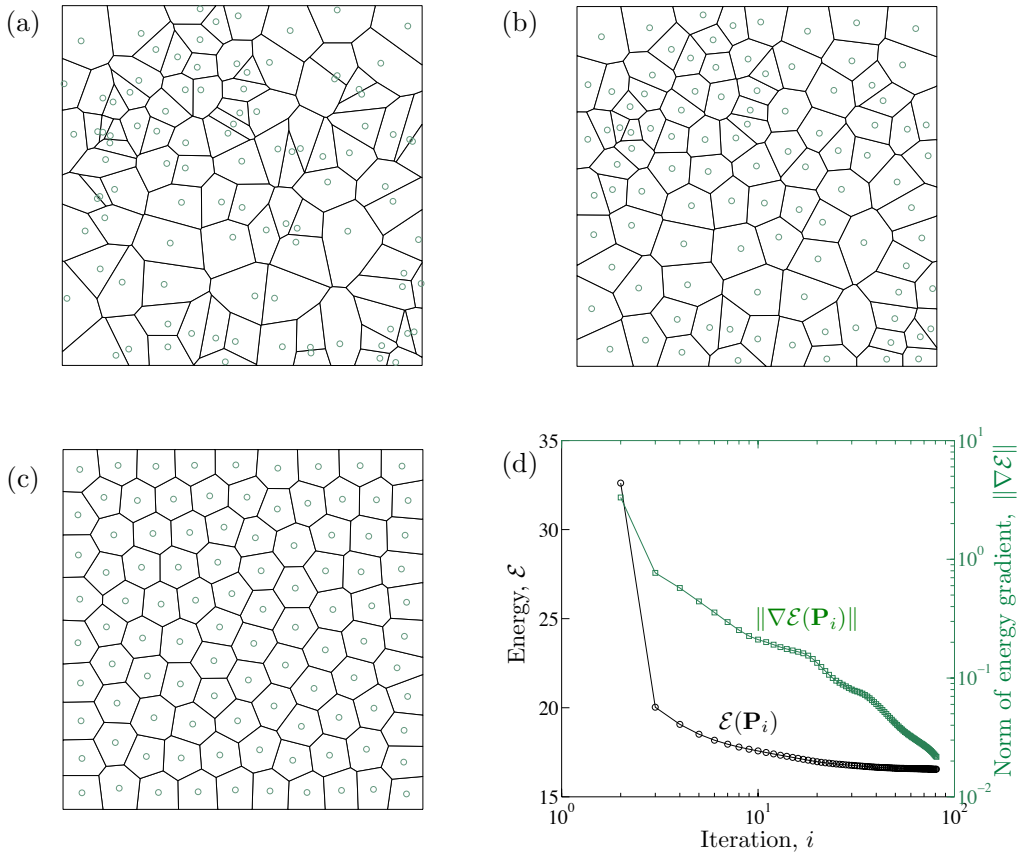


Figure 6.5: (a) Random initial point set $\mathbf{P}_1$ and the corresponding Voronoi diagram (b) First iteration of Lloyd's method: the Voronoi diagram generated by $\mathbf{P}_2 = \mathbf{L}(\mathbf{P}_1)$, i.e., the centroids of the Voronoi cells of $\mathbf{P}_1$ (c) Distribution of seeds and the diagram after 80 iterations (d) Monotonic convergence of the energy functional and decay in the norm of its gradient

and $\mu(\mathbf{x})$ is a given density function defined over Δ. Hence, in a Centroidal Voronoi tessellation (CVT), each generating point $\mathbf{y}$ coincides with the centroid $\mathbf{y}_c$ of the corresponding region (i.e., $V_{\mathbf{y}} \cap \Delta$).

An alternative variational characterization of a CVT is based on the deviation of each Voronoi region from its generating seed, measured by the following *energy* functional:

$$\mathcal{E}(\mathbf{P}; \Delta) = \sum_{\mathbf{y} \in \mathbf{P}} \int_{V_{\mathbf{y}}(\mathbf{P}) \cap \Delta} \mu(\mathbf{x}) \, \|\mathbf{x} - \mathbf{y}\|^2 \, \mathrm{d}\mathbf{x} \tag{6.12}$$

Note that the energy depends on points in $\mathbf{P}$ not only through the appearance of $\mathbf{y}$ in the integral but also the Voronoi cells in domain of the integral. Critical points of $\mathcal{E}(\mathbf{P}; \Delta)$ are point sets that generate CVTs since the gradient of energy functional with respect to a given $\mathbf{y} \in \mathbf{P}$ is given by [65, 110]:

$$\nabla_{\mathbf{y}}\mathcal{E} = 2m_{\mathbf{y}}\left(\mathbf{y} - \mathbf{y}_c\right) \quad \text{where} \quad m_{\mathbf{y}} = \int_{V_{\mathbf{y}}\cap\Delta} \mu(\mathbf{x})\mathrm{d}\mathbf{x} \tag{6.13}$$

Clearly $\nabla_{\mathbf{y}}\mathcal{E} = \mathbf{0}$ when relation (6.11) holds. Moreover, CVTs can be further classified based on the minimization of the energy functional. The CVTs corresponding to saddle points of $\mathcal{E}$ are called *unstable* while local and global minimizers (for fixed number of seeds) of the energy functional are known as *stable* and *optimal* CVTs, respectively [68, 110]. The CVTs in the latter groups form a more compact tessellation of the domain and due to this property find many applications in areas other than mesh generation—see [65] for a survey on the topic.

A simple but powerful method for computing CVTs is the *Lloyd's algorithm*, which iteratively replaces the given generating seeds by the centroids of the corresponding Voronoi regions. Lloyd's algorithm can be thought of as a fixed point iteration for the mapping $\mathbf{L} = (\mathbf{L}_{\mathbf{y}})_{\mathbf{y}\in\mathbf{P}}^{T} : \mathbb{R}^{n\times 2} \to \mathbb{R}^{n\times 2}$ where each component function is given by:

$$\mathbf{L}_{\mathbf{y}}(\mathbf{P}) = \frac{\int_{V_{\mathbf{y}}(\mathbf{P})\cap\Delta} \mathbf{x}\mu(\mathbf{x})\mathrm{d}\mathbf{x}}{\int_{V_{\mathbf{y}}(\mathbf{P})\cap\Delta} \mu(\mathbf{x})\mathrm{d}\mathbf{x}} \tag{6.14}$$

Therefore, $\mathbf{L}$ maps the point set $\mathbf{P}$ to the set of centroids of the Voronoi cells in $\mathcal{T}(\mathbf{P};\Delta)$. Given an initial point set $\mathbf{P}_1$, the Lloyd's method produces point set $\mathbf{P}_{k+1} = \mathbf{L}(\mathbf{P}_k)$ at the kth iteration. From the above relation, it is clear that a fixed point of this map, i.e., one that satisfies $\mathbf{P} = \mathbf{L}(\mathbf{P})$, forms a CVT. In [64], it is shown that the energy functional decreases in consecutive iterations of Lloyd's algorithm, that is,

$$\mathcal{E}(\mathbf{P}_{i+1};\Delta) \leq \mathcal{E}(\mathbf{P}_i;\Delta) \tag{6.15}$$

which means that the Lloyd's algorithm can be viewed as a descent method for the energy functional. This property is illustrated in Figure 6.5.

As discussed later, Lloyd's algorithm is incorporated in the proposed meshing scheme to construct more uniform polygonal meshes. We assess the performance of the Lloyd's algorithm by comparing the quality and uniformity of the resulting CVT meshes to meshes obtained from random and quasirandom placement of seeds. The latter approach is recommenced in [27, 179] where the authors obtain more uniform Voronoi tessellations by enforcing a minimum allowable distance between the interior seeds. This minimum separation eliminates the bunching up of the random seeds, and can also be used to generate graded meshes. However, the approach may produce distorted elements not suitable for use in finite element analysis. We consider the discretization of a square domain with a circular hole using random and quasi-random seed placement, and the proposed procedure. The resulting meshes shown in Figures 6.6, 6.7 and 6.8 consists of $n = 1000$ elements. In the

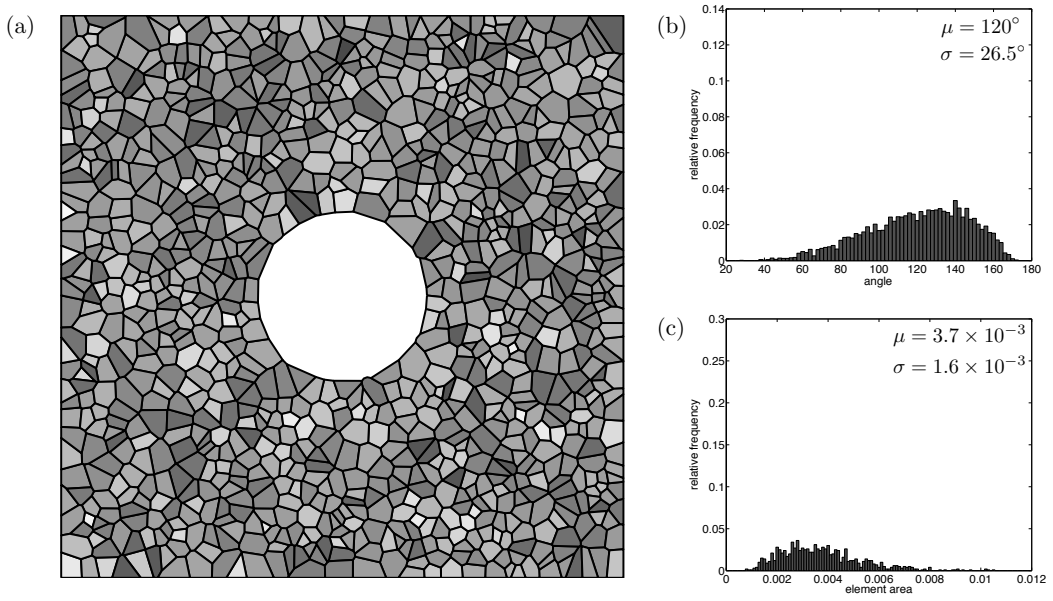


Figure 6.6: Random seed placement (a) resulting mesh with coefficient of variation of edge lengths plotted in gray-scale (b) histogram of interior angles of the mesh (c) histogram of element areas

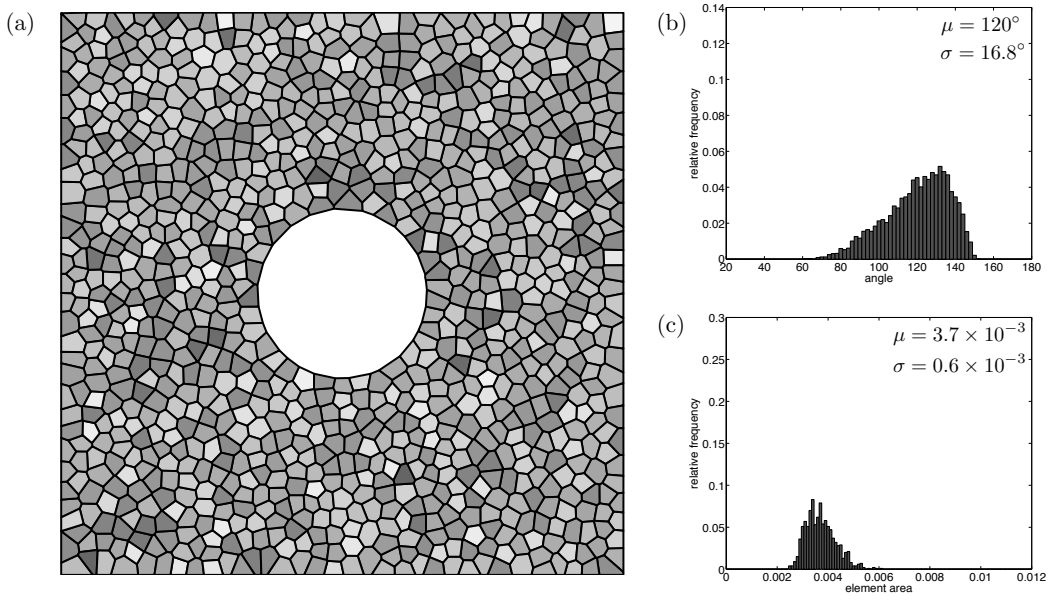


Figure 6.7: Quasi-random seed placement (a) resulting mesh with coefficient of variation of edge lengths plotted in gray-scale (b) histogram of interior angles of the mesh (c) histogram of element areas

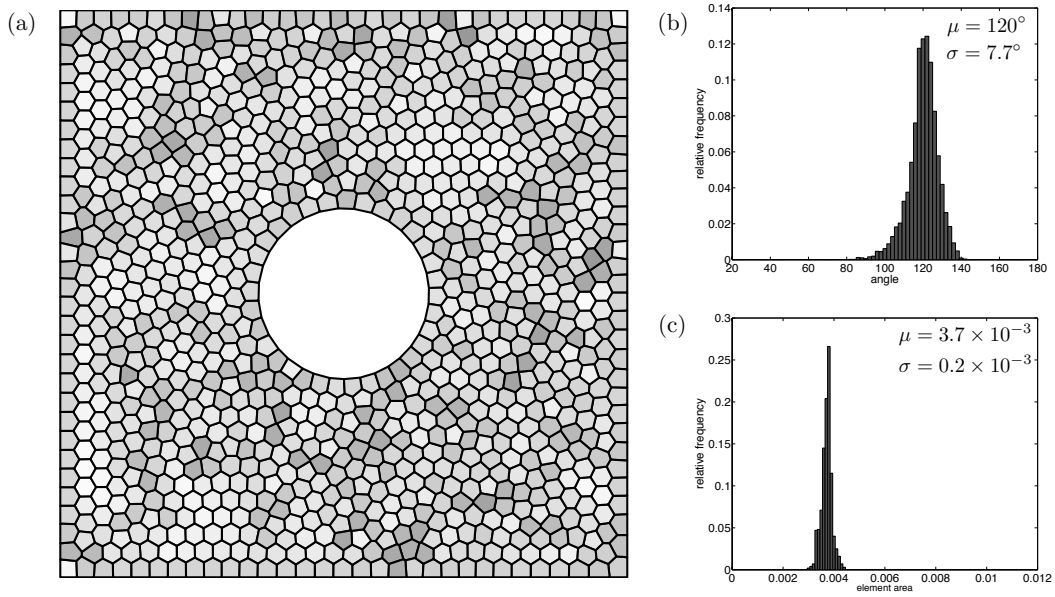


Figure 6.8: CVT mesh generation (a) mesh with coefficient of variation of edge lengths plotted in gray-scale (b) histogram of interior angles of the mesh (c) histogram of element areas

case of quasi-random discretization, we use a minimum allowable distance of $\sqrt{0.68ab/n}$, as prescribed by Bolander and Saito [27] for a rectangular domain with dimensions a and b. The CVT mesh was constructed with 100 iterations of Lloyd's algorithm. As a measure of mesh quality, the coefficient of variation of edge lengths for each element is plotted in gray-scale. The coefficient of variation for a regular polygon is zero (indicated as white in the figures) since all the edges have the same length. We observe that the CVT mesh is superior in terms of element quality. The histograms of interior angle (at the triple joints) and element area are also shown in these figures. These plots indicate that Lloyd's iterations drive the Voronoi mesh towards a regular hexagonal packing. Moreover, we note that the CVT mesh is significantly more uniform in size over the domain.

6.3 Voronoi meshing algorithm

Before discussing the details of the proposed meshing algorithm, we illustrate the main ideas based on the concepts developed so far. As shown by Bolander et al. [27, 179], a polygonal discretization can be obtained from the Voronoi diagram of a given set of seeds and their reflections.

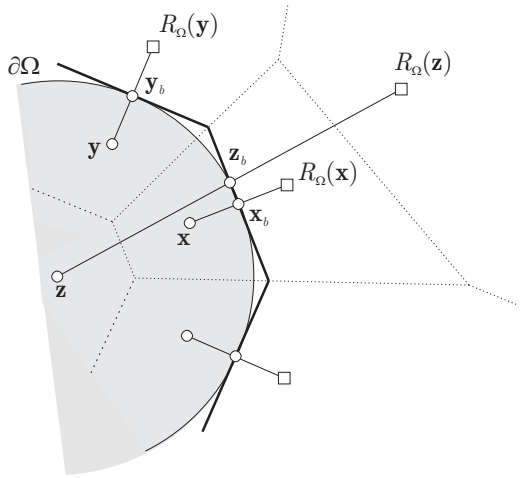


Figure 6.9: Illustration of the meshing approach: the Voronoi edges shared between seeds and their reflection approximate the boundary of the domain. Note that the reflections of the interior seeds "far" from the boundary (e.g. point $\mathbf{z}$ in the figure) do not contribute to the final mesh

Explanation of the approach

Assume $\Omega \subseteq \mathbb{R}^2$ is a bounded *convex* domain with smooth boundary and $\mathbf{P}$ is a given set of distinct seeds in Ω. To construct a polygonal discretization of Ω, we first reflect each point in $\mathbf{P}$ about the *closest* boundary point of Ω and denote the resulting set of points by $R_\Omega(\mathbf{P})$:

$$R_\Omega(\mathbf{P}) := \{R_\Omega(\mathbf{y}) : \mathbf{y} \in \mathbf{P}\} \tag{6.16}$$

Convexity of Ω ensures that all of the reflected points lie outside of Ω. We then construct the Voronoi diagram of the plane by including the original point set as well as its reflection. In other words, we compute $\mathcal{T}(\mathbf{P} \cup R_\Omega(\mathbf{P}); \mathbb{R}^2)$. If the Voronoi cells of a point $\mathbf{y}$ and its reflection have a common edge, i.e., if $V_\mathbf{y} \cap V_{R_\Omega(\mathbf{y})} \neq \emptyset$, then this edge is tangent to $\partial\Omega$ at the $\mathbf{y}_b$ (see Figure 6.9). Therefore, these edges form an approximation to the domain boundary and a reasonable discretization of Ω is given by the collection of Voronoi cells corresponding to the points in $\mathbf{P}$. For a given point set $\mathbf{P}$, such a discretization is uniquely defined and is denoted by $\mathcal{M}_\Omega(\mathbf{P})$. Thus, we have:

$$\mathcal{M}_\Omega(\mathbf{P}) = \left\{V_\mathbf{y} \in \mathcal{T}(\mathbf{P} \cup R_\Omega(\mathbf{P}); \mathbb{R}^2) : \mathbf{y} \in \mathbf{P}\right\} \tag{6.17}$$

We further note that the convexity of Ω implies that the boundary edges lie on the exterior of the domain and so the discretization $\mathcal{M}_\Omega(\mathbf{P})$ covers Ω.

Clearly a better approximation is obtained if the points in $\mathbf{P}$ are distributed more "evenly" in Ω. In our algorithm, we will incorporate Lloyd's iterations to obtain a point set $\mathbf{P}$ that produces a CVT. Since optimal CVTs consist of Voronoi cells that are congruent to a basic cell (and thus are uniform in size) [68], it is expected $\cup_{V \in \mathcal{M}_\Omega(\mathbf{P})} V \approx \Omega$ especially

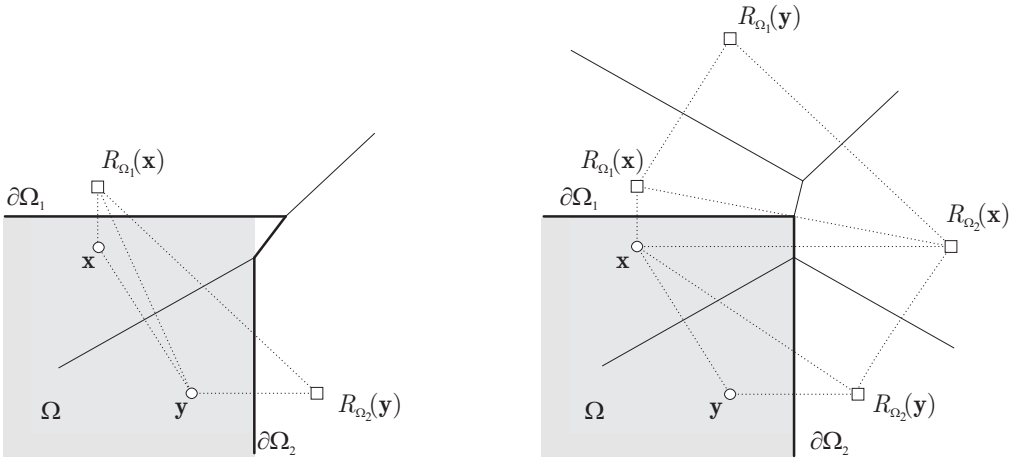


Figure 6.10: To accurately capture a corner, nearby seeds need to be reflected about both boundary segments incident on that corner

for a large number of generating points. This gives a systematic and consistent approach for discretizing Ω under the given assumptions.

Algorithm

The basic ideas laid above can be extended for discretization of more general domains, in particular those that are non-convex and have piecewise smooth boundaries (e.g. $\partial\Omega$ has corner points where there is a jump in the normal vector). These features lead to a number of complications that require modifications of the previous approach. For example, reflecting a point about the nearest boundary point may not be sufficient to capture a nearby corner (see Figure 6.10). We resolve this issue by reflecting the seeds about both boundary segments incident on the corner. Similarly, for non-convex domains, reflection of a seed far from the boundary may land inside the domain or interfere with the reflection of another seed (Figure 6.11). We check the sign and value of the distance function to exclude such a scenario. Finally, as seen in Figure 6.9, the reflection of most of the seeds in the interior of the domain has no effect on the approximation of the boundary. Thus, we add a condition to reflect only seeds that are in a band near the boundary. This significantly reduces the computational cost and improves the robustness of algorithm by alleviating the problem of interference, which is important in dealing with complex non-convex domains. Based on these considerations, the following algorithm is proposed.

We consider domains Ω that are formed by finite union, intersection and/or difference of smooth but perhaps unbounded regions Ω_i, $i = 1, \ldots, m$. Using formulas in (6.7), the distance function associated with Ω can be written as a function of d_{Ω_i}:

$$d_\Omega(\mathbf{x}) = F(d_{\Omega_1}(\mathbf{x}), \ldots, d_{\Omega_m}(\mathbf{x})) \tag{6.18}$$

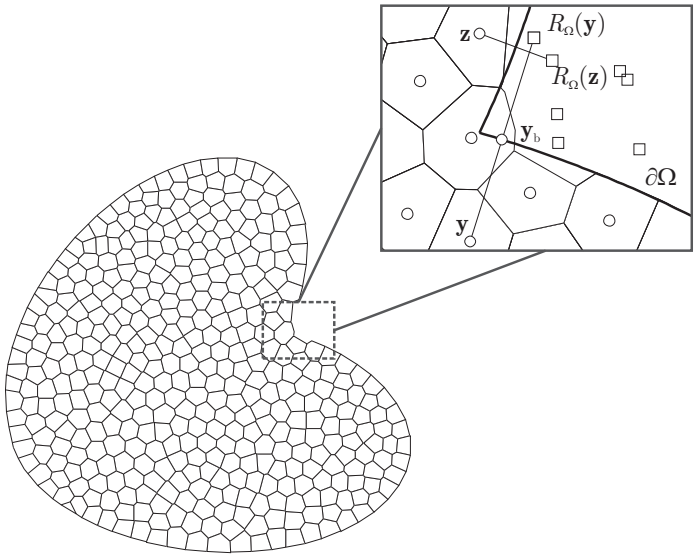


Figure 6.11: In this non-convex domain, the reflection $R_\Omega(\mathbf{y})$ is closer to the boundary of the domain than the seed $\mathbf{y}$ itself, i.e., $|d_\Omega(R_\Omega(\mathbf{y}))| < |d_\Omega(\mathbf{y})|$. Not only this reflection does not contribute to the approximation of the boundary, it causes interference with seed $\mathbf{z}$

Algorithm 1 Initial random seed placement

input: B, n *%%$B \supset \Omega$ is the bounding box and n is the desired number of seeds*
  set $\mathbf{P} = \emptyset$
  while $|\mathbf{P}| < n$ **do**
    generate random point $\mathbf{y} \in B$
    if $d_\Omega(\mathbf{y}) < 0$ **then**
      $\mathbf{P} \leftarrow \mathbf{P} \cup \{\mathbf{y}\}$
    end if
  end while
output: $\mathbf{P}$

It is expected that Ω_i's and operations represented by F are defined in such a way that distance to the *every boundary segment of* Ω *can be found among the values of* d_{Ω_i}.

The first step in the algorithm is to generate an initial set of points $\mathbf{P}$. Algorithm 1 shows the basic steps for obtaining a random point set of n size. The implementation of random seed generation is simplified by specifying a bounding box $B = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ that contains Ω. A random seed $\mathbf{y} \in B$ is accepted only if it lies inside the domain and this is determined by evaluating the sign of the $d_\Omega(\mathbf{y})$.

As discussed above, the set of reflections must be chosen carefully in order to deal with possible non-convex and non-smooth features of Ω. In Equation (6.17), $R_\Omega(\mathbf{P})$ may not be sufficient for producing a good discretization $\mathcal{M}_\Omega$. The procedure for computing the new set of reflections, denoted by $R(\mathbf{P})$, is outlined in Algorithm 2. A seed $\mathbf{y} \in \mathbf{P}$ is reflected about boundary segment $\partial\Omega_i$ provided that:

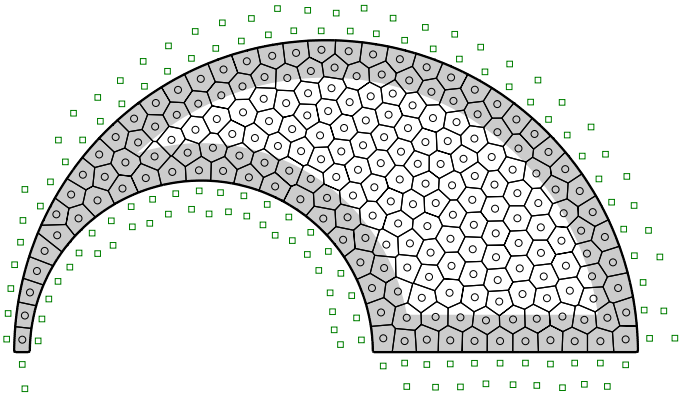


Figure 6.12: In the algorithm, only the seeds in band of length $\alpha(n, \Omega)$ near the boundary (shaded area in the figure) are reflected

$$|d_{\Omega_i}(\mathbf{y})| < \alpha(n, \Omega) \tag{6.19}$$

where $\alpha(n, \Omega)$ is a distance value proportional to the width of an element:

$$\alpha(n, \Omega) := c \left(\frac{|\Omega|}{n} \right)^{1/2} \tag{6.20}$$

We choose the constant of proportionality c to be greater than 1 so that α is larger than the average element width. Note that convex corners are captured as nearby seeds are reflected about both boundary segments incident on these corners.

The reflection $\overline{\mathbf{y}} = R_{\Omega_i}(\mathbf{y})$ is accepted if it lies outside the domain, i.e., $d_\Omega(\overline{\mathbf{y}}) > 0$. Moreover, the following criterion is added to avoid interference with the reflection of other seeds:

$$|d_\Omega(\overline{\mathbf{y}})| > \eta \, |d_{\Omega_i}(\mathbf{y})| \tag{6.21}$$

where $0 < \eta < 1$ is a specified parameter to adjust for numerical errors (round-off and numerical differentiation). Figure 6.11 illustrates the idea behind this criterion: In the case of a convex domain, $|d_\Omega(\overline{\mathbf{y}})| = |d_{\Omega_i}(\mathbf{y})|$ so no complication will occur. In general, however, the reflection may be closer to a boundary segment of Ω, other than the one that generated it (i.e., Ω_i), in which case the reflection will not help with the approximation of the boundary and may possibly interfere with the reflection of another seed. In particular, we expect an interference when $d_\Omega(\overline{\mathbf{y}}) < -d_{\Omega_i}(\mathbf{y})$. Hence we exclude this possibility by accepting only the reflections that satisfy the condition in Equation (6.21).

Once the set of reflections are determined, the Voronoi diagram of the $\mathbf{P} \cup R(\mathbf{P})$ is constructed. For each $\mathbf{y} \in \mathbf{P}$, we compute the centroid of Voronoi cell $V_\mathbf{y}$ to complete the first iteration of the meshing routine. Following the idea of Lloyd's algorithm, the set

Algorithm 2 Reflection

input: $\mathbf{P}, \alpha, \eta$
  $R(\mathbf{P}) = \emptyset$
  for each $\mathbf{y} \in \mathbf{P}$ **do**
    if $|d_\Omega(\mathbf{y})| < \alpha$ **then**
      for $i = 1$ to m **do**
        if $|d_{\Omega_i}(\mathbf{y})| < \alpha$ **then**
          let $\overline{\mathbf{y}} = R_{\Omega_i}(\mathbf{y})$
          if $d_\Omega(\overline{\mathbf{y}}) > 0$ **and** $|d_\Omega(\overline{\mathbf{y}})| > \eta \, |d_{\Omega_i}(\mathbf{y})|$ **then**
            $R(\mathbf{P}) \leftarrow R(\mathbf{P}) \cup \{\overline{\mathbf{y}}\}$
          end if
        end if
      end for
    end if
  end for
output: $R(\mathbf{P})$

of centroids $\mathbf{P}_c$ will replace $\mathbf{P}$ in the next iteration until convergence is achieved. The convergence criterion is based on the magnitude of the gradient of the energy functional. In particular, the algorithm should terminate when $\|\nabla \mathcal{E}\| \approx 0$. Recalling Equation (6.13), the norm of the gradient is given by:

$$\|\nabla \mathcal{E}\| := \left(\sum_{\mathbf{y} \in \mathbf{P}} \|\nabla_{\mathbf{y}} \mathcal{E}\|^2 \right)^{1/2} = \left(\sum_{\mathbf{y} \in \mathbf{P}} m_{\mathbf{y}}^2 \, \|\mathbf{y} - \mathbf{y}_c\|^2 \right)^{1/2} \tag{6.22}$$

We can see that this quantity is in fact a measure of the movement of the seeds in two consecutive iterations, weighted by the "size" of their Voronoi cells through terms $m_{\mathbf{y}}$. To identify the appropriate convergence tolerance, we note:

$$m_{\mathbf{y}} \propto \frac{1}{n} \int_\Omega \mu(\mathbf{x}) \mathrm{d}\mathbf{x}, \quad \|\mathbf{y} - \mathbf{y}_c\|^2 \propto |V_{\mathbf{y}}| \propto \frac{|\Omega|}{n} \tag{6.23}$$

and so the norm of the gradient scales as:

$$\|\nabla \mathcal{E}\| \propto \left[n \left(\frac{1}{n} \int_\Omega \mu(\mathbf{x}) \mathrm{d}\mathbf{x} \right)^2 \frac{|\Omega|}{n} \right]^{1/2} = \frac{|\Omega|^{1/2}}{n} \int_\Omega \mu(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{6.24}$$

We define the following "non-dimensional" error parameter:

$$E_r := \frac{n \, \|\nabla \mathcal{E}\|}{|\Omega|^{1/2} \int_\Omega \mu(\mathbf{x}) \mathrm{d}\mathbf{x}} \tag{6.25}$$

Algorithm 3 Main function

input: n, M, ϵ_{tol} *%%number of seeds n, max. number of iterations M, tolerance ϵ_{tol}*
  generate $\mathbf{P}$, a random point set of size n
  $i \leftarrow 0$, $E_r \leftarrow 1$, $\mathbf{P}_c \leftarrow \mathbf{P}$ *%Initialization of variables*
  while $i \leq M$ **and** $E_r \geq \epsilon_{\text{tol}}$ **do**
    $\mathbf{P} \leftarrow \mathbf{P}_c$
    compute reflections $R(\mathbf{P})$
    construct diagram $\mathcal{T}(\mathbf{P} \cup R(\mathbf{P}); \mathbb{R}^2)$
    calculate $\mathbf{P}_c \leftarrow \{\mathbf{y}_c : \mathbf{y} \in \mathbf{P}\}$
    compute E_r using Equation (6.25)
    $i \leftarrow i + 1$
  end while
output: $\mathcal{M}_\Omega(\mathbf{P}) = \left\{V_\mathbf{y} \in \mathcal{T}(\mathbf{P} \cup R(\mathbf{P}); \mathbb{R}^2) : \mathbf{y} \in \mathbf{P}\right\}$

Thus convergence is established when:

$$E_r < \epsilon_{\text{tol}} \tag{6.26}$$

Here $0 < \epsilon_{\text{tol}} \ll 1$ is the specified convergence tolerance. The pseudo-code for the main routine is shown in Algorithm 3. Note that the mesh is still defined by expression (6.17) with $R_\Omega(\mathbf{P})$ replaced by $R(\mathbf{P})$.

Remarks on min/max formulas

A remark is in order regarding min/max formulas in (6.7) and implicitly used in (6.18) and their influence on the behavior of the meshing algorithm. As mentioned before, these expressions may produce incorrect distance values in certain regions of the plane. A close inspection of the algorithm shows that the *magnitude* of d_Ω generated by equation (6.18) is used only in the evaluation of interference criterion (6.21). Elsewhere—in generating an initial point set, checking condition (6.19), or computing reflection of a seed—we either use the sign of d_Ω or the distance values of the constituent domains. The requirement on F and the structure of min/max formulas implicitly used in F guarantee that $|d_\Omega(\overline{\mathbf{y}})| = \left|d_{\Omega_j}(\overline{\mathbf{y}})\right|$ for some index j. Here d_Ω denotes the distance function generated by F, which may be different from the exact distance functions associated with Ω in some regions of the plane. In such a case, the violation of condition (6.21) implies

$$\left|d_{\Omega_j}(\overline{\mathbf{y}})\right| = |d_\Omega(\overline{\mathbf{y}})| \leq \eta\, |d_{\Omega_i}(\mathbf{y})| < |d_{\Omega_i}(\mathbf{y})| \tag{6.27}$$

and so the reflection $\overline{\mathbf{y}} = R_{\Omega_i}(\mathbf{y})$ is "correctly" rejected because it is expected to cause interference with approximation of Ω_j. In fact, in this algorithm, we do not need the exact distance function associated with Ω. Note our need for d_{Ω_i}'s is primarily due to the issue of corners and capturing non-convex features.

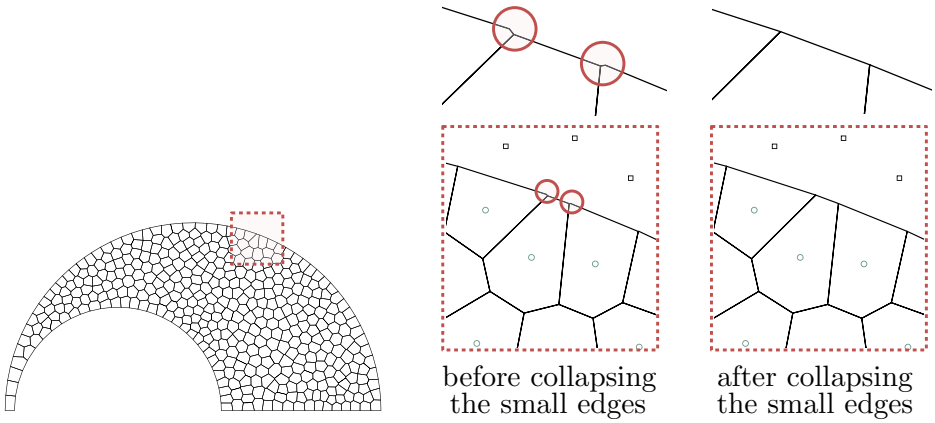


Figure 6.13: Small edges can form in elements near a curved boundary since the generating seeds are not the same distance from that boundary. The issue can be addressed by a post-processing step of collapsing these edges onto a single node

6.4 Two mesh modification procedures

We discuss two procedures that are applied to the Voronoi discretization to make it suitable for use it finite element applications. The first mesh modification procedure addresses the issue of small edges that can appear in the Voronoi meshes in the interior or around the boundary of the domain. In latter case, this phenomenon is due to the use of numerical differentiation to compute the gradient of the distance function and unequal distance of seeds to curved boundaries, as illustrated in Figure 6.13. A small interior edge can also form in the interior of the domain when four seeds are almost co-circular (see Figure 6.14). The presence of these edges can in turn lead to a high condition number of the finite element stiffness matrix associated with the mesh.

A simple remedy is to collapse the edges that are small compared to the element size into a single node. This operation does not distort the mesh topology (elements remain to be strictly convex polygons) and ensures a uniform quality of the mesh. An alternative approach is discussed in [139], which augments the energy functional to penalize the small edges. In our implementation, we search all the edges in the mesh by looping over all the elements. Given an edge of an element, we compute the angle β between the vectors connecting the center (the location with mean value of the coordinates of the nodes) of the element and the vertices forming the edge. The edge will be collapsed into a single node when

$$\beta < \epsilon_a \left(\frac{2\pi}{\ell} \right) \tag{6.28}$$

Here ℓ is the number of vertices of the element and ϵ_a is a user-defined tolerance.

The second issue regarding the Voronoi meshes is that since the seeds are placed ran-

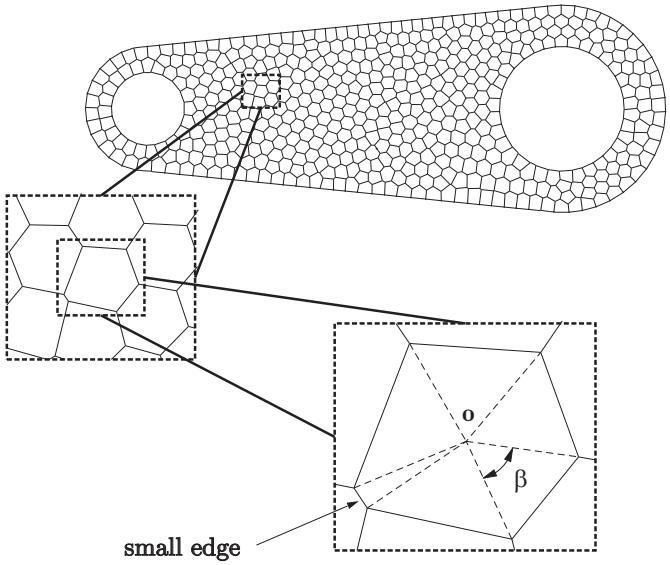


Figure 6.14: Illustration of small interior edges in a CVT and definition of angle β in Equation (6.28)

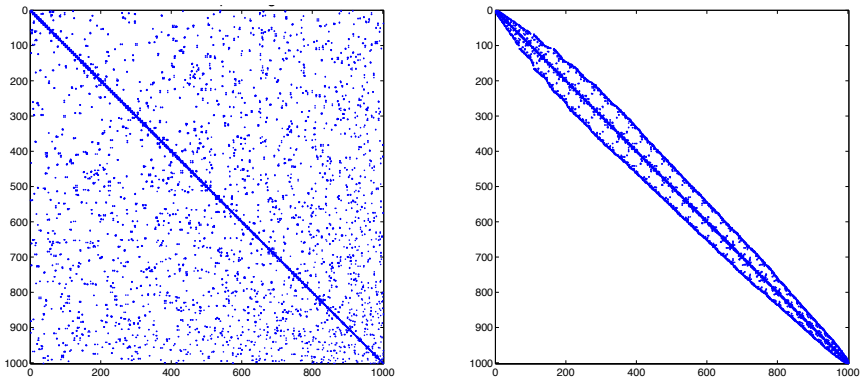


Figure 6.15: Sparsity pattern for the stiffness of polygonal mesh with 500 elements and 1002 nodes before RCM resequencing (left) and after resequencing (right). The bandwidth is reduced from 966 to 75.

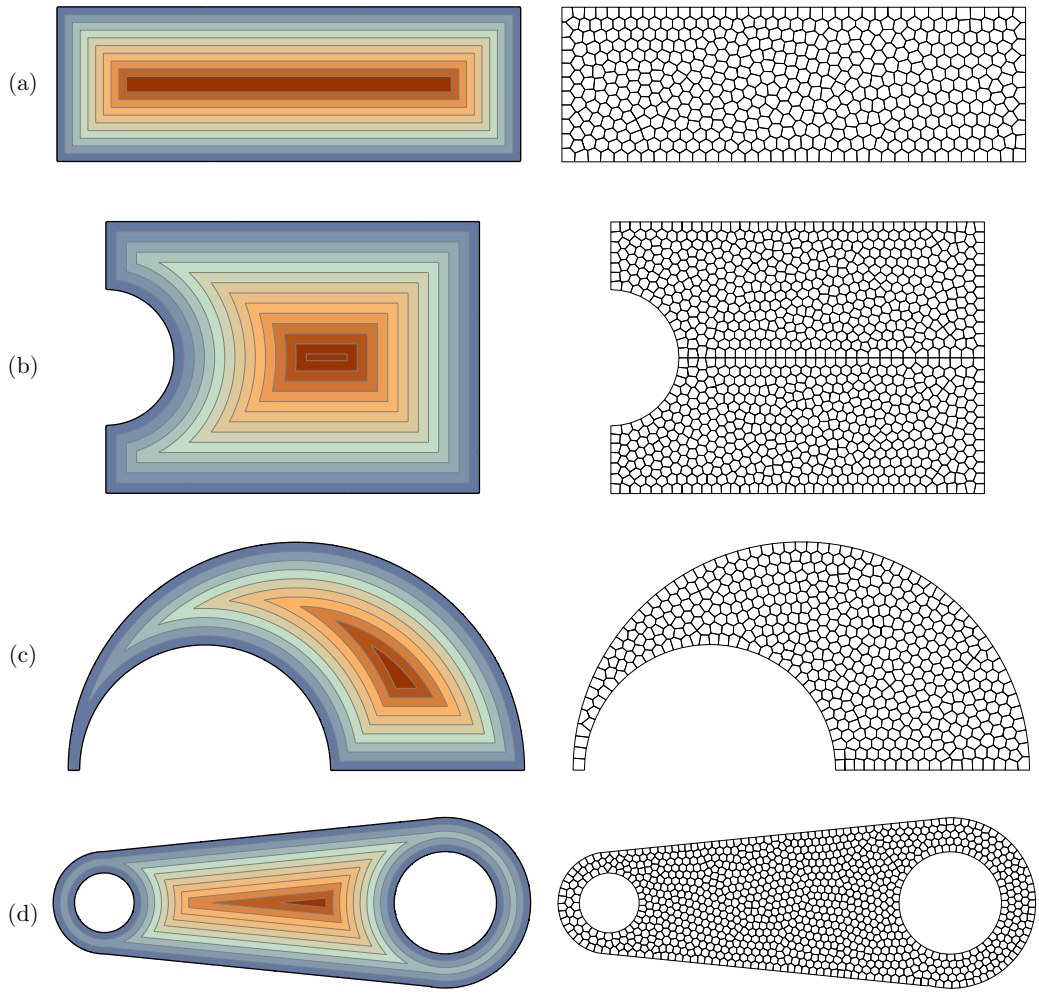


Figure 6.16: Distance functions (left) and sample meshes (right) for various domains (a) MBB beam (b) cantilever (c) Horn (d) Wrench

domly, the element and node numbering of the resulting mesh will be random. This can cause the associated stiffness matrix to exhibit an undesirable sparsity pattern with a large bandwidth. A remedy to this problem consists of using the Reverse Cuthill-McKee (RCM) algorithm, which is designed to reduce the bandwidth and profile (skyline) of sparse symmetric matrices by systematically reordering the node numbers [56]. For example, a typical mesh with 500 elements and $1,002$ nodes produced by the meshing algorithm has the sparsity pattern shown in Figure 6.15. After applying RCM, the bandwidth is reduced from 966 (almost full) to 75. This can lead to substantial savings in the computational time needed for solving the resulting linear systems with certain direct solvers.

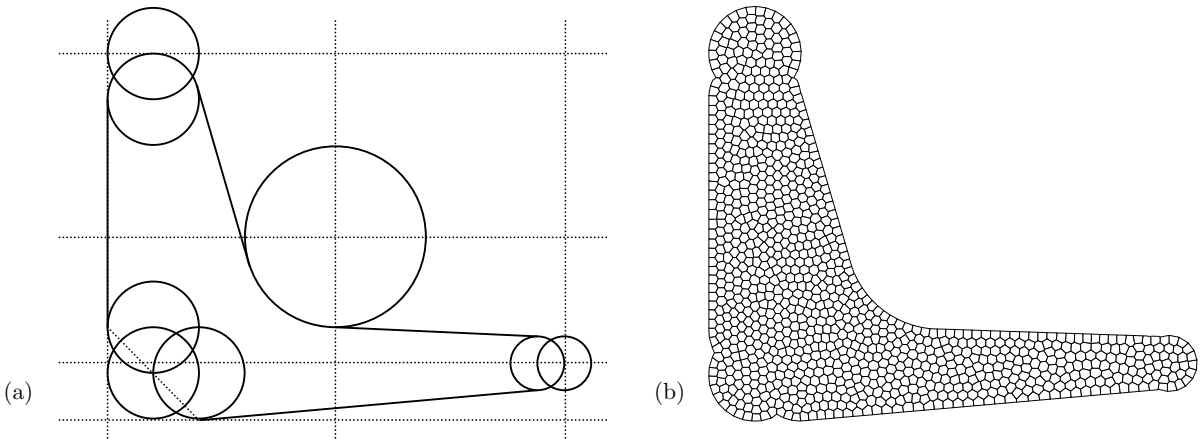


Figure 6.17: Suspension triangle (a) geometry and (b) discretization

6.5 Examples of domains and meshes

Figures 6.16 shows the contour plot of the distance function and a sample mesh for several domains that have been used for topology optimization examples throughout the thesis. The domains are relatively simple and can be constructed from basic constituent geometries such as half planes and circles. In the appendix, we show how these distance functions can be constructed using a basic library of distance functions. For these meshes, it is assumed $\mu(\mathbf{x}) \equiv 1$ and so $\mathbf{y}_c$ is the centroid of a polygon, $m_{\mathbf{y}} = |V_{\mathbf{y}}|$ and the expression for the error is simplified as:

$$E_r = \frac{n}{|\Omega|^{3/2}} \left(\sum_{\mathbf{y} \in \mathbf{P}} |V_{\mathbf{y}}|^2 \, \|\mathbf{y} - \mathbf{y}_c\|^2 \right)^{1/2} \tag{6.29}$$

In all cases, the parameters $c = 1.5$, $\epsilon_a = 0.1$ and $\epsilon_{\text{tol}} = 5 \times 10^{-3}$ were used. Notice that the mesh in Figure 6.16(b) is symmetric about the horizontal axis. This was done by simply constraining the seeds to one half of the domain and reflecting them about the axis of symmetry in each iteration to obtain the interior point set.

One can also describe more complicated geometries using the same basic ingredients. We show two such examples where the construction of the distance functions involves several constituent geometries and set operations. The first is an extended domain for the design of a socket wrench and the other is domain for a suspension triangle which is presented as an industrial application of topology optimization in [7]. The constituent domains of the suspension triangle are shown in Figure 6.17(a). The meshes in Figures 6.17(d) and 6.17(b) are both made up of 1,000 elements.

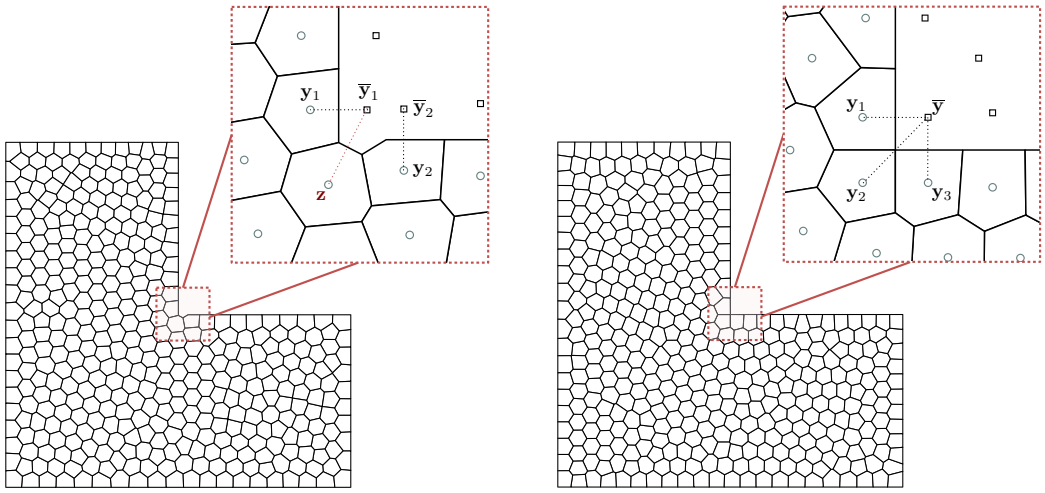


Figure 6.18: In the left figure, the reflection of point $\mathbf{y}_1$ has interfered with the approximation of the horizontal boundary by seeds $\mathbf{z}$ and $\mathbf{y}_2$. In the right figure, seeds $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$ are fixed in such way that they all have $\overline{\mathbf{y}}$ as their reflection

6.6 Extensions

We conclude the chapter by discussing a few extensions of our meshing algorithm, which were left out of the present algorithm (and the Matlab code in the appendix) for the sake of simplicity and clarity. The first is related to the approximation of certain non-convex corners, which may not be adequately captured with the reflection criteria described above. An example of a domain that exhibits such geometry is the L-shaped domain shown in Figure 6.18. Since the seeds near the non-convex corner are placed independently (during the Lloyd's iterations), their reflections may interfere with each other. One possible solution is to place some seeds at an equal distance from the corner and fix them during the Lloyd's iteration. This process can be made systematic by choosing, as the fixed seeds, the reflections of appropriately places seed *outside* such a corner (e.g. point $\overline{\mathbf{y}}$ in Figure 6.18).

The second extension is regarding the generation of non-uniform meshes. The Lloyd's algorithm with constant density function, $\mu(\mathbf{x})$, leads to a uniform distribution of seeds and subsequently a Voronoi discretization that is uniform in size over the entire domain. It is possible to generate non-uniform meshes with desirable gradation by selecting an appropriate density function, which biases the placement of points in certain regions (see the effects on varying density function in [65]). Note that, to generalize the code further in this direction, an integration scheme for convex polygons is needed to compute the centroids. Alternatively, we can choose an initial distribution of seeds such that regions that need refinement contain more seeds. Persson and Strang [125] and Persson [124] have employed a rejection method that relies on defining a mesh size function $h(\mathbf{x})$ over the domain. Figure

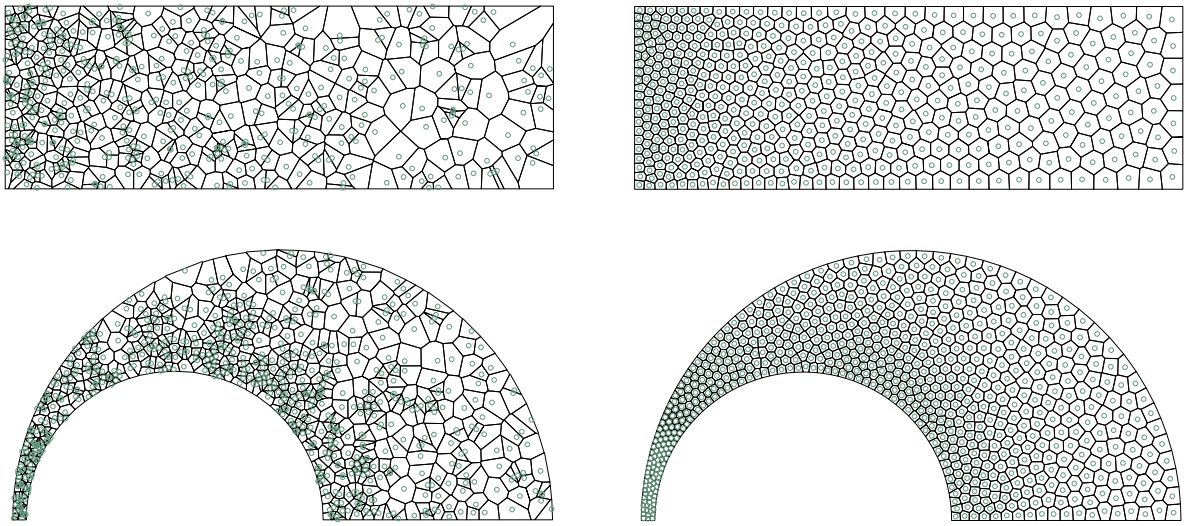


Figure 6.19: Sample graded meshes: the left figures show the initial distribution of seeds generated using a mesh size function and rejection method and the right figures show the final mesh after Lloyd's iterations

6.19 shows examples of graded meshes that can be generated by the algorithm presented in this work in conjunction with the rejection method. Note that Lloyd's iterations naturally lead to a smooth gradation in the mesh. Although not explored, it is also possible to combine the two approaches and specify suitable and related density and mesh size functions. Since the gradation in the mesh is often dictated by the geometry of the domain, it is natural that both μ and h be defined based on the distance function d_Ω.

Chapter 7

Conclusions and Extensions

In this thesis, we explored various theoretical and computational aspects of restriction methods for shape and topology optimization. In particular, we discussed three main set of issues, namely appropriate parametrization and restriction of shapes, finite element discretization of the associated sizing optimization problem, and the algorithm for solving the resulting large-scale discrete system. Regarding the first issue, we discussed and analyzed the Ersatz approximation in elasticity (and briefly a similar procedure for Stokes flow) as a first step in the "sizing" transformation. We presented a framework for analysis of the well-posedness of restriction methods and considered the implications for density and implicit function formulations. We provided an alternative rationale for density formulations that dispenses with the notion of microscopic material distribution in the presence of macroscopic restrictions on the density variation. In the case of implicit functions, since a clear delineation of regularity conditions is less frequently addressed in the literature, we discussed the significance of the transversality condition for these methods in practice, especially as it pertains to the issue of approximation of the Heaviside map. These observations can potentially serve as a starting point to devise more transparent implicit function and level-set methods and possibly eliminate the commonly used numerical heuristics such as modification of gradient information and frequent use of reinitialization.

We also explored various regularization schemes including the popular filtering methods, Tikhonov and total variation regularization and their characteristics from a numerical point of view. Noting that the choice of regularization is intimately linked with one's choice of parametrization of the unknown geometry, there are more possibilities that can be explored. One strategy worth pursuing is to consider an appropriately defined hierarchical basis (e.g., wavelets) and prescribe complexity of the optimal shapes through the degree of *sparsity* in such basis. This would in turn allow multi-resolution control of geometric features. Recall that total variation regularization can be viewed as a means to promote the sparsity of the spatial gradient of the optimal field. Such a perspective plays an important role in image processing. At the heart of the burgeoning field of compressed sensing [40, 41], for example, is the use of *a priori* knowledge that the unknown signals (e.g., images) have a sparse structure and can be reconstructed with fewer measurements than previously thought possible (on the order of internal sparsity) by means of an appropriately posed sparse

optimization problem. Analogously, in the optimal shape design, one would prescribe a desirable degree of sparsity of the unknown optimal shape which then would be recovered using a tailored sparse optimization framework.

With respect to the second issue, we discussed how the discretization of the design and response fields in optimal shape design can be non-trivial and that naive finite element approximations of the problem may suffer from numerical instabilities such as spurious checkerboard patterns. These grid-scale anomalies are in part due to the inaccurate analysis of the design response. For two-dimensional problems, we showed that isoparametric polygonal finite elements, unlike their low-order Lagrangian counterparts, are not susceptible to such instabilities. This behavior is attributed to the enhanced approximation characteristics of these elements, which also alleviate shear and volumetric locking phenomena. Regarding the latter property, we also showed that low-order polygonal discretizations satisfy the so-called Babuska-Brezzi condition required for stability of mixed variational formulation of incompressible elasticity and Stokes flow problems. Conceptually, the polygonal elements are the natural extension of commonly used linear triangles and bilinear quads to all convex n-gons. To facilitate their use, we developed a simple but robust meshing algorithm that utilizes Voronoi diagrams to generate convex polygonal discretizations of implicit geometries. A set of self-contained discretization and analysis Matlab codes using polygonal elements has been developed and made available for the community. While the meshing algorithm does readily extend to three dimensions, developing an element formulation for conforming polyhedra that retains the simplicity of its 2D counterpart remains an open question.

What lies at the heart of the above-mentioned numerical instabilities is the fact the parameterization of design and approximation of its response in topology optimization, are rigidly linked through the discretization process. Ideally, for each application, one should use the numerical solver that best suits the physics of the problem in order to compute the response of the candidate designs as efficiently and accurately as possible. To do so, the discretization of the governing equation must therefore be decoupled from that of the design field. This in turn requires an efficient data structure capable of supporting multiple dynamic discretizations and the transfer of information between them, systematic addition of sensitivity analysis routines to PDE-solvers, as well as guidelines for adaptivity of the design and response fields based on rigorous error estimates.

Finally, with regards to the optimization algorithms, we discussed the general sequential approximation optimization approach underlying popular algorithms such as the optimality criteria method and MMA. We also developed and explored at great length an operator splitting algorithm to solve topology optimizations problems with an additive regularization term and subject to convex constraints. The key idea is that the splitting algorithm matches the structure of the problem and treats the structural performance objective (that

depends on the state equation), the regularization term (that controls the geometric complexity) and the constraints, separately. On the one hand, this decomposition opens up the possibility to improve the quality of optimal designs by modifying its various ingredients. For example, we discovered that by augmenting the projection operation that enforces the box-constraints, the algorithm produces binary solutions with sharp boundaries even for large regularization effects. This is in contrast to the usual filtering methods, in which, as noted earlier, with more complexity control comes more severe smearing of the boundary. On the other hand, the convex subproblems generated by the splitting algorithm in each iteration are significantly simpler to solve since the dependence on the state equation is removed. As a result, the algorithm is able to accommodate nonsmooth regularization schemes. In the case of total variation regularization, for example, the subproblems are identical to the classical image de-noising problems for which efficient algorithms can be found in the image processing literature. We were also able to significantly speed up the convergence of the algorithm by introducing a diagonal approximation to the Hessian of the structural objective, which amounts to using a variable metric in computing the gradients in course of the optimization. In the extensions of this work, it would be interesting to consider nonsmooth regularizers such as the total variation within the present variable metric scheme. This would require the extension of available denoising algorithms (e.g., [43, 33]) for solving the resulting subproblems in each iteration. Also of interest is the use of accelerated first order methods such as those proposed in [118] and [16] that can improve the convergence speed of the algorithms. Developing a two-metric variation of such algorithms for the constrained minimization problems of topology optimization is especially promising.

Appendix A

Educational Codes

Sharing and publication of educational software has become a tradition in the topology optimization community. For instance, in addition to the popular "99 line" code [140] and its successor [12], Allaire and Pantz [9] presented a structural optimization code based on FreeFem++. Liu et al. [111] introduced a coupled level set method using the FEMLAB package and Challis [42] presented a discrete level-set Matlab code very much in the spirit of the "99 line" code. More recently, Suresh [150] developed a 199 line code for Pareto-optimal tracing with the aid of topological derivatives. We aim to extend and complement the previous works by presenting a Matlab implementation of topology optimization that, among other things, features a general framework for finite element discretization and analysis.

Many engineering applications of topology optimization cannot be defined on a rectangular domain or solved on a structured square mesh. The description and discretization of the design domain geometry, specification of the boundary conditions for the governing state equation, and accurate computation of the design response may require the use of *unstructured meshes*. One main goal here is to provide the users a self-contained analysis tool in Matlab and show how the topology optimization code should be structured so as to separate the analysis routine from the particular formulation used. We elected to focus on polygonal discretizations in this educational effort for the following reasons: (i) The concept of Voronoi diagrams offers a simple way to discretize two-dimensional geometries with convex polygons. In chapter 6, we discussed a simple and robust Voronoi mesh generation scheme that relies on an implicit description of the domain geometry; (ii) Polygonal finite elements outperform linear triangles and quads in topology optimization as they are not susceptible to numerical instabilities such as checkerboard patterns [101, 137, 161, 162]; (iii) The isoparametric formulation for polygonal finite elements can be viewed as extension of the common linear triangles and bilinear quads to all convex n-gons [149, 148, 162]. As a special case, these codes can generate and analyze structured triangular and quadrilateral meshes.

First, we present a simple and robust Matlab code, called `PolyMesher`, for Voronoi polygonal mesh generation algorithm discussed in chapter 6. The mesh generator can provide, among other things, the input needed for finite element and optimization codes

that use linear convex polygons. The use of polygonal elements makes possible meshing of complicated geometries with a self-contained Matlab code. The main ingredients of the present mesh generator are the implicit description of the domain and the centroidal Voronoi diagrams used for its discretization. The signed distance function provides all the essential information about the domain geometry and offers great flexibility to construct a large class of domains via algebraic expressions. Examples are provided to illustrate the capabilities of the code, which is compact and has fewer than 135 lines.

Next we present a simple and efficient Matlab code, called `PolyTop`, for solving topology optimization problems that features a general finite element analysis routine and encompasses a large class of formulations as discussed in chapter 1. We address the use of arbitrary meshes in topology optimization and discuss certain practical issues that have received little attention in the literature. For example, higher computational cost associated with unstructured meshes, cited as a motivation for adopting uniform grids, is sometimes attributed to the need for repeated calculations of the element stiffness matrices. However, as we demonstrate, the invariant quantities such as the element stiffness matrices and the global stiffness matrix connectivity can be computed once and stored for use in subsequent optimization iterations. This approach is feasible since the amount of memory required is relatively small considering the hardware capacities of current personal computers. Moreover, the overhead associated with computing multiple element stiffness matrices is small compared to the overall cost of the optimization algorithm which may require solving hundreds of linear systems. The Matlab code `PolyTop` retains the readability and efficiency of previous educational codes while offering a general FE framework. In terms of efficiency, our Matlab implementation has performance that is on par with the recently published 88 line code [12] and is faster for large meshes. As an example of arbitrary meshes, we have implemented the FE routine based on isoparametric polygonal finite elements. However, we note that that since the FE code is general, it is straightforward for the users to implement new elements (e.g. higher order, non-conforming, etc.) or use meshes generated by other software (an example of a Matlab mesh generator is `distMesh` by [125]).

Another related goal of this work is to present a modular code structure for topology optimization in which the analysis routine for solving the state equation and computing the sensitivities is made independent of the specific formulation chosen. This means that the analysis routine need not know about the specific formulation used which in turn allows the code to accommodate various formulations without compromising its clarity or pragmatism. By contrast, previous educational software, e.g., the 99- and 88-line and similar codes, often mix the analysis and formulation—perhaps in the interest of keeping the code short and compact—and so require multiple versions [12]. Within the present framework, the analysis routine can be developed, modified, or extended without any effect on the optimization code. Conversely, if a new problem demands a different type of analysis,

a suitable analysis package can replace the one presented. We note that other Matlab codes for topology optimization can make use of our general analysis code. The formalism of this decoupling approach is crucial when one seeks to develop the framework for solving more complicated and involved topology optimization problems including multiphysics response, multiple geometry or state constraints, though for the simple compliance benchmarks the difference may seem minor.

A.1 PolyMesher

We begin by describing the structure of the code, the input and output parameters and the user-defined `Domain` function that characterizes the meshing domain. Next we make some comments on the routines inside the meshing kernel. The kernel of the mesh generator is implemented in the `PolyMesher` function. The following variables are the user inputs to this function:

```
[Node,Element,Supp,Load,P] = PolyMesher(@Domain,NElem,MaxIter,P)
```

`Domain:` This is a Matlab function defining the domain. As shown above, a handle to this function (e.g., `@MichellDomain`) is passed to the kernel, providing access to information about domain geometry (i.e., the sign distance function), the bounding box B, and the boundary conditions. The details of this function are discussed below.

`NElem:` This is an integer that represents the desired number of elements in the mesh. When an initial point set `P` is specified, `NElem` is replaced by the number of elements in `P`.

`MaxIter:` This integer specifies the maximum number of Lloyd's iterations. By setting `MaxIter` to zero and providing an initial point set, the user can obtain/recover the Voronoi mesh produced by that point set.

`P:` The user has the option of inputting an initial set of seeds through the array `P`, in which the kth row represents the coordinates of the kth seed. If no such input is passed to the code, an initial random point set is generated (see description of the function `PolyMshr_RndPtSet` below).

The `PolyMesher` function computes and returns the Voronoi discretization along with the final set of seeds and the boundary conditions arrays (`Supp` and `Load` as described below). The mesh is represented by the commonly used data structure in finite element community, namely a node list and an element connectivity matrix. The node list, `Node`, is a two-column array whose ith row represents the coordinates of the node with index number

Use	Input	Output
bounding box	`Domain('BdBox')`	coordinates of bounding box `[xmin,xmax,ymin,ymax]`
distance values	`Domain('Dist',P)`	$(m+1) \times n$ matrix of distance values for point set $\mathbf{P}$ consisting of n points
boundary conditions	`Domain('BC',Node)`	cell consisting of `Load` and `Supp` arrays

Table A.1: Various uses and commands for the `Domain` function

i. The connectivity `Element` is stored inside a Matlab cell of size `NElem`. The kth entry of the cell contains the indices of the nodes incident on the kth element in counter-clockwise order[1].

All the domain-related information is included in `Domain` defined outside the kernel. This provides more flexibility for generating meshes for new domains and eliminates the need for the repeated modifications of the kernel. The domain function is used in the kernel for three distinct purposes: (1) retrieving the coordinates of the bounding box for the domain, (2) obtaining the distance of a given point set to the boundary segments, and (3) determining the boundary condition arrays for a given node list. In the sample domain functions, the input string `Demand` specifies the type of information requested. Table A.1 summarizes the different input and output choices.

Within the `Domain` function, the user must define the distance function of the meshing domain. This function, called `DistFnc` in the sample domain functions, accepts the coordinates of a point set $\mathbf{P}$ and return a matrix of $(m+1) \times n$ distance values. The entry located at ith column and jth row represents the signed distance value $d_{\Omega_i}(\mathbf{P}_j)$, while the last column is the signed distance for the entire domain Ω. If `d` is a matrix of distance values, this last column can be quickly accessed by command `d(:,end)`.

The user also defines the function that returns the lists of nodal loads and supports given the node list for the mesh. The nodal support array `Supp` has three columns, the first holds the node number, the second and third columns give support conditions for that node in the x- and y-direction, respectively. Value of 0 means that the node is free, and value of 1 specifies a fixed node. The nodal load vector `Load` is structured in a similar way, except for the values in the second and third columns represent the magnitude of the x- and y-components of the force.

Comments on the functions in the kernel

We now proceed to discuss some of the details of the implementation of the kernel and comment on its functions. We remark that in this implementation, it is assumed $\mu(\mathbf{x}) \equiv 1$.

[1]The cell structure allows for storing vectors of different size and is therefore suitable for connectivity of polygonal elements with different number of nodes.

`PolyMesher:` The main function follows closely the pseudo-code in Algorithm 3 in section 6.3. On line 7, we check to see if an initial point set is provided by the user. The variable `NElem` is updated on line 8 to make sure it is consistent with size of `P`. The iteration counter `It` and error value `Err` are initialized such that the while-loop is executed at least once. Upon the first execution, line 15 recovers the initial points in variable `P`. The area of the domain, needed for computing E_r and α, is initially set to the area of the bounding box on line 11. This value is updated when the centroids of the elements are computed[2].

Since the output of the Matlab `voronoin` command on line 17 is the set of vertices and connectivity of the entire Voronoi diagram (i.e., with all the cells in $\mathcal{T}(\mathbf{P} \cup R(\mathbf{P}); \mathbb{R}^2)$) and the discretization is composed only of the cells in $\mathcal{M}_\Omega$, we need to extract the nodes and connectivity of these cells. This is done in the `PolyMshr_ExtrNds` function (line 24). The mesh is updated once more in `PolyMshr_CllpsEdgs` to remove small edges that can appear in a CVT (line 25). Furthermore, since resulting node numbering and connectivity are random (as a consequence of random placement), the bandwidth of finite element stiffness matrix may be too large. This issue is addressed in the `PolyMshr_RsqsNds` function (line 26). The main function ends with obtaining the boundary condition arrays from the domain function (line 27) and plotting the final mesh (line 28).

`PolyMshr_RndPtSet:` This function generates an initial random point set of size `NElem`. A candidate point set `Y` of size `NElem` is generated using the coordinates of the bounding box and `rand` function in Matlab (lines 33-34). Only seeds in `Y` that lie inside the domain are accepted. The variable `Ctr` counts the number of seeds accepted so far. The while-loop terminates when the desired number of seeds is reached.

`PolyMshr_Rflct:` The implementation of this function follows Algorithm 2 in section 6.3. The gradient of the distance function is computed by means of numerical differentiation:

$$\mathbf{n} := \nabla d_{\Omega_i}(\mathbf{y}) \approx \epsilon_d^{-1} \left(d_{\Omega_i}(\mathbf{y} + (\epsilon_d, 0)) - d_\Omega(\mathbf{y}), d_{\Omega_i}(\mathbf{y} + (0, \epsilon_d)) - d_{\Omega_i}(\mathbf{y}) \right) \tag{A.1}$$

Here ϵ_d is a small positive number, set to 10^{-8} by default on line 43. We compute the normal vector for the entire point set at once on lines 46 and 47, where `n1` and `n2` denote the first and second components of the normal, respectively. The logical array index `I`, computed on line 48, is then used to identify and reflect the seeds only about the nearby boundary segments, i.e., those within distance of α. In order to compute the reflections `R_P` at once, the point set `P` is extended, by repeating its columns, to two matrices of the same size as `n1` and `n2` on lines 49 and 50. Once the candidate set of reflections `R_P` of the boundary-adjacent seeds are obtained (lines 51-52), the two conditions for accepting the

[2]This small overhead can be removed after a few iterations once a good estimate value is obtained

reflections are enforced on line 55 by means of another logical array index `J`.

`PolyMshr_CntrdPly:` This function returns the areas and centroids of the first `NElem` cells in the mesh. Since the density is assumed to be constant, we can compute the area and centroid of each cell using the available formulae for polygons. The signed area for an ℓ-sided polygon is given by:

$$A = \frac{1}{2}\sum_{k=1}^{\ell}\left(v_x^{[k]}v_y^{[k+1]} - v_x^{[k+1]}v_y^{[k]}\right) \tag{A.2}$$

where $(v_x^{[k]}, v_y^{[k]})$ is the coordinates of the kth vertex of the polygon, $k = 1, \ldots, \ell$. In the above equation, we are defining $(\ell+1)$th vertex to be the same as the first. Similarly, The formula for computing the centroid is:

$$c_x = \frac{1}{6A}\sum_{k=1}^{\ell}\left(v_x^{[k]} + v_x^{[k+1]}\right)\left(v_x^{[k]}v_y^{[k+1]} - v_x^{[k+1]}v_y^{[k]}\right) \tag{A.3}$$

$$c_y = \frac{1}{6A}\sum_{k=1}^{\ell}\left(v_y^{[k]} + v_y^{[k+1]}\right)\left(v_x^{[k]}v_y^{[k+1]} - v_x^{[k+1]}v_y^{[k]}\right) \tag{A.4}$$

`PolyMshr_ExtrNds:` The original discretization is passed to this function through variables `Element0` and `Node0`. The connectivity of the Voronoi cells that make up the mesh are stored in the first n arrays of the cell variable `Element0` as a result of passing the seeds `P` in the first block of the input to Matlab function `voronoin` on line 17. However, the vertices of these cells are not necessarily stored in the first block of rows of `Node0`. The extraction of the additional nodes requires modification of both the node list and the connectivity matrix. The function `PolyMshr_ExtrNds` first creates a one-dimensional array, `map`, containing the indices of the nodes that must remain in the mesh (the Matlab function `unique` is used to remove the appearance of duplicate nodes). The array `cNode`, needed for updating the node list and element connectivity matrix, is constructed on lines 69-70. By setting the entries of `cNode` that correspond to the nodes that must be removed to the maximum value in `map` (which is necessarily the index of a node that will remain in the node list) on line 70, we ensure that they are removed in `PolyMshr_RbldLists` function (see below for the description of this function).

`PolyMshr_CllpsEdgs:` This function addresses the issue of appearance of small edges in the meshes. The simple remedy, implemented in `PolyMshr_CllpsEdgs` function, is to collapse the edges that are small compared to the element size into a single node. We search all the edges in the mesh by looping over all the elements (for-loop in lines 76-84). Given an

edge of an element, we compute the angle between the vectors connecting the center (the location with mean value of the coordinates of the nodes) of the element and the vertices forming the edge (lines 79-80). The edge will be collapsed into a single node according to 6.28. To update the node list and element connectivity matrices, the set of edges that needs to be collapsed is stored in array `cEdge` (line 86). The input `cNode` is defined on line 89 such that the two nodes at the end of a tagged edge are replaced by a single node in `PolyMshr_RbldLists`.

`PolyMshr_RsqsNds:` In this function, a call is made to Reverse Cuthill-McKee function in Matlab to renumber the nodes once more to ensure that the finite element stiffness matrix associate with the mesh will a small bandwidth. The sparsity pattern of the corresponding stiffness matrix is first computed (lines 98-104). The assembly of this pseudo-stiffness matrix `K` is carried out by the use of Matlab's sparse assembly function (on line 105). The Matlab function `symrcm` returns the reverse Cuthill-McKee ordering, stored in variable `p`, which is then used to compute the `cNode` array passed to `PolyMshr_RbldLists` function. The reader is referred to the Matlab documentation on functions `sparse` and `symrcm` for further information of this implementation.

`PolyMshr_RbldLists:` This auxiliary function is used in all three mesh modification functions (`PolyMshr_ExtrNds`, `PolyMshr_CllpsEdgs`, and `PolyMshr_RsqsNds`) for updating the node list and element connectivity according to the information contained in the input array `cNode`. This is an array of integer with the same length as the input node list `Node0`. If the ith entry of `cNode` is set to have value k, then after the update, the ith node, `Node0(i)`, is replaced by the coordinates of kth node, `Node0(k)`. For example, in order to collapse nodes `i` and j, we can set `cNode(i)=j` (see line 89 in `PolyMshr_CllpsEdgs` function). Similarly, if we wish to remove the node with index `i` from the node list (which is the case in `PolyMshr_ExtrNds` function), we set `cNode(i)` to be the highest index of the nodes that will remain in the mesh after the modification. On line 112, we use the `unique` function in Matlab to identify the duplicate nodes and the resulting index array `ix` allows us to extract the desired nodal coordinates from `Node0`. When reducing the node list size we need to make sure that the last mapped node is the maximum node of the input map `cNode` (line 113). The connectivity `Element` is updated on line 116 while lines 117-119 guarantee that final arrangement of nodes in the connectivity cell `Element` is ordered counter-clockwise. Note that the `voronoin` does not necessarily store the nodes in the connectivity cell in a counter-clockwise order requiring this correction.

`PolyMshr_PlotMsh:` This function plots the mesh and boundary conditions. In order to use the Matlab `patch` function to plot the entire mesh at once, we create an element

connectivity matrix `ElemMat` that is padded with `NaNs`. This matrix has `NElem` rows and `MaxNVer` columns (as computed on line 125, `MaxNVer` is the maximum number of vertices of elements in the mesh). The location of support and loads are also marked provided that `Supp` and `Load` arrays are passed to the function.

Examples

We will now show to how construct the distance functions associated with the geometries shown in Figure 6.16 in order to illustrate the use of the code. The `Domain` functions for these example geometries and the library of distance functions are provided as supplementary material in [163].

MBB beam The domain is a rectangular box with width of 3 and height of 1, and bottom left corner located at the origin. The distance function is simply given by:

```
d = dRectangle(p,0,3,0,1)
```

and the bounding box is the entire domain, i.e., `BdBox = [0 3 0 1]`. The boundary conditions for the MBB problem are specified by searching the nodes located along the left edge and bottom corner. The following command was used to generate the result in Figure 6.16(a):

```
[Node,Element,Supp,Load,P] = PolyMesher(@MbbDomain,200,100);
```

Michell cantilever The second example is the extended domain for the Michell cantilever beam problem. The support is a circular hole and the a vertical load is applied at midspan of the free end. The distance function is constructed as follows:

```
d1 = dRectangle(p,0,5,-2,2)
d2 = dCircle(p,0,0,1)
d  = dDiff(d1,d2)
```

The mesh shown in Figure 6.16(b) consists of 1000 elements, and was constructed to be symmetric about horizontal axis (located at the midspan). The initial point set was restricted to the upper half of the domain by modifying line 36 as follows:

```
I = find(d(:,end)<0 & Y(:,2)>0);
```

Also during the iterations that point set was reflected about the x-axis to obtain the complete set of generating seeds. In particular, we replaced `P = Pc` on line 15 by the following line of code:

```
P = [Pc(1:NElem/2,:);[Pc(1:NElem/2,1),-Pc(1:NElem/2,2)]];
```

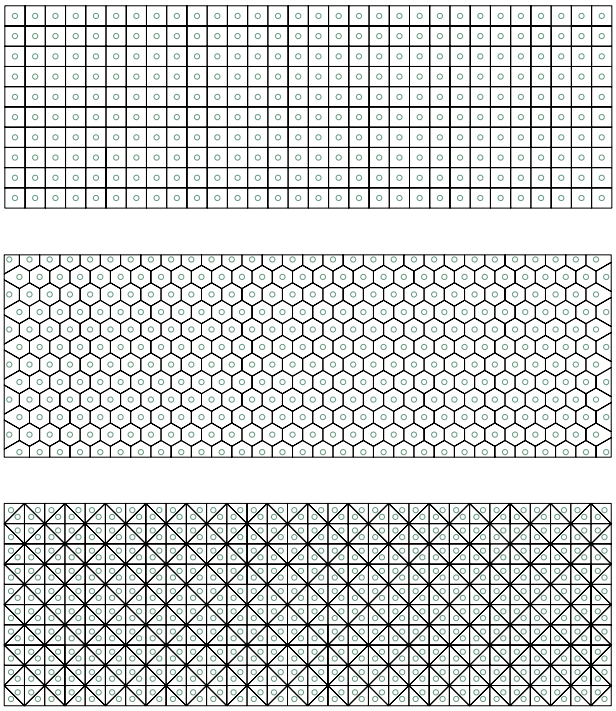


Figure A.1: Uniform discretizations of the MBB beam domain obtained from appropriate placements of the seeds

Horn The third example has the geometry of a horn, which comes from the difference of two half-circles. The distance function is computed by:

```
d1 = dCircle(p,0,0,1)
d2 = dCircle(p,-0.4,0,0.55)
d3 = dLine(p,0,0,1,0)
d  = dIntersect(d3,dDiff(d1,d2));
```

The mesh in Figures 6.16(c) consists of 500 elements.

Uniform meshes The proposed algorithm can also be used to generate certain uniform meshes (regular tessellations), as shown in Figure A.1 for the MBB domain. The user must specify the set of seed `P` and turn off the Lloyd's iterations by setting `MaxIter` to zero. The following set of seeds generates a uniform rectangular mesh with `nelx` elements in the x-direction and `nely` elements in the y-direction for the MBB beam problem:

```
dx = 3/nelx; dy = 1/nely;
[X,Y] = meshgrid(dx/2:dx:3,dy/2:dy:1);
P = [X(:)  Y(:)];
```

A.2 PolyTop

The framework of the educational code reflects the findings of chapter 1, which reviewed the various steps taken in casting the optimal shape problem as a sizing optimization problem and identified the structure of the resulting discrete optimization problem in sufficient generality. As mentioned before, the code also features a modular structure in which the analysis routine and the optimization algorithm are separated from the specific choice of topology optimization formulation. Within this framework, the finite element and sensitivity analysis routines contain no information related to the formulation and thus can be extended, developed and modified independently.

The model problem is compliance minimization problem (see the derivation in section 1.4)

$$\min\ f(\mathbf{z}) \qquad \text{subject to} \qquad g(\mathbf{z}) \leq 0 \tag{A.5}$$

where the objective and constraint functions are defined by

$$f(\mathbf{z}) = \mathbf{F}^T\mathbf{U}, \quad g(\mathbf{z}) = \frac{\mathbf{A}^T m_V(\mathbf{P}\mathbf{z})}{\mathbf{A}^T\mathbf{1}} - \overline{v} \tag{A.6}$$

Here $\mathbf{F}$ is given by (1.30) and is independent of the design variables $\mathbf{z}$, $\mathbf{U}$ solves (1.29) in which $\mathbf{K}$ depends "linearly" on $m_E(\mathbf{P}\mathbf{z})$ as stated in (1.40), $\mathbf{A} = (|\Omega_\ell|)$ is the vector of element volumes, and $\mathbf{P}$ is defined in (1.38).

Modular framework, formulation, and optimizer

The structure of the discrete optimization problem (A.5) allows for the separation of the analysis routine from the particular topology optimization formulation chosen. The analysis routine is defined to be the collection of functions in the code that compute the objective and constraint functions and thus require access to mesh information (e.g. the FE analysis and functions computing the volume or perimeter of the design). We can see that for the minimum compliance problem vectors $\mathbf{E} := m_E(\mathbf{y})$ and $\mathbf{V} := m_V(\mathbf{y})$, i.e., the element stiffnesses and volume fractions, are the only "design" related information that need to be provided to the analysis functions. The analysis functions need not know about the choice of interpolations functions which corresponds to the choice of sizing parametrization or the mapping $\mathcal{P}$ that places constraints on the design space. Therefore a general implementation of topology optimization in the spirit of this discussion must be structured in such a way that the finite element routines contain no information related to the specific topology optimization extended, developed and modified independently.

We can also see that certain quantities used in the analysis functions, such as element areas $\mathbf{A}$ and stiffness matrices $\mathbf{k}_\ell$ as well as the "connectivity" of the global stiffness matrix $\mathbf{K}$ need to be computed only once in the course of the optimization algorithm. Therefore, for

the sake of efficiency of the implementation, one should store the invariant quantities. We emphasize that this is independent of the uniformity of the FE partition $\mathcal{T}_h$, i.e., whether or not it is a structured mesh. Similarly the matrix $\mathbf{P}$ can be computed once in a pre-processing step and stored.

To use a gradient-based optimization algorithm for solving the discrete problem (A.5), which is the case in our Matlab code, we must also compute the gradient of the cost functions with respect to the design variables $\mathbf{z}$. The sensitivity analysis can be "separated" along the same lines. The analysis functions would compute the sensitivities of the cost functions *with respect to their own internal parameters*. Note that by the chain rule:

$$\frac{\partial f}{\partial z_k} = \sum_{\ell=1}^{N} \left(\frac{\partial E_\ell}{\partial z_k} \frac{\partial f}{\partial E_\ell} + \frac{\partial V_\ell}{\partial z_k} \frac{\partial f}{\partial V_\ell} \right) \tag{A.7}$$

or in equivalent vector form:

$$\nabla f = \frac{\partial \mathbf{E}}{\partial \mathbf{z}} \frac{\partial f}{\partial \mathbf{E}} + \frac{\partial \mathbf{V}}{\partial \mathbf{z}} \frac{\partial f}{\partial \mathbf{V}} \tag{A.8}$$

Therefore the analysis functions would only compute the sensitivities of f with respect to the internal parameters $\mathbf{E}$ and $\mathbf{V}$. In the compliance problem, $f = \mathbf{F}^T\mathbf{U}$ and we have:

$$\frac{\partial f}{\partial E_\ell} = -\mathbf{U}^T \frac{\partial \mathbf{K}}{\partial E_\ell} \mathbf{U} = -\mathbf{U}^T \mathbf{k}_\ell \mathbf{U}, \qquad \frac{\partial f}{\partial V_\ell} = 0 \tag{A.9}$$

This means that the FE function that computes the objective function also returns the negative of the element strain energies as the vector of sensitivities $\partial f/\partial \mathbf{E}$. Similarly, for compliant mechanism design where $f = -\mathbf{L}^T\mathbf{U}$ (see section 1.6), we have

$$\frac{\partial f}{\partial E_\ell} = \overline{\mathbf{U}}^T \frac{\partial \mathbf{K}}{\partial E_\ell} \mathbf{U} = \overline{\mathbf{U}}^T \mathbf{k}_\ell \mathbf{U} \tag{A.10}$$

where $\overline{\mathbf{U}}$ is the solution to the adjoint problem

$$(\mathbf{K} + \mathbf{K}_s)\, \overline{\mathbf{U}} = \mathbf{L} \tag{A.11}$$

The remaining terms in (A.8) depend on the formulation, i.e., on how the design variables $\mathbf{z}$ are related to the analysis parameters. For example, $\mathbf{E} = m_E(\mathbf{P}\mathbf{z})$ and $\mathbf{V} = m_V(\mathbf{P}\mathbf{z})$ implies:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_E}(\mathbf{P}\mathbf{z}), \qquad \frac{\partial \mathbf{V}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_V}(\mathbf{P}\mathbf{z}) \tag{A.12}$$

where $J_{m_E}(\mathbf{y}) := \mathrm{diag}(m'_E(y_1), \ldots, m'_E(y_N))$ is the Jacobian matrix of map m_E. The evaluation of expression (A.8) is carried out outside the analysis routine and the result, $\partial f/\partial \mathbf{z}$, is passed to the optimizer to update the values of the design variables.

In the same way that the analysis routine ought to be separated from the rest of topology optimization algorithm, the optimizer responsible for updating the values of the design variables should also be kept separate. This is perhaps better known in the topology optimization community as optimizers like MMA [153] are generally used as black-box routines. Of course, it is important to know and understand the inner workings of the optimizer. As noted in [82], certain choices of approximations of cost functionals in a sequential optimization algorithm lead to an Optimality-Criteria type update expression when there is only one constraint. An attempt to directly change the OC expression, for example, to accommodate a more general formulation such as the one presented may not be readily obvious. In order to account for the general box constraints (i.e., upper and lower bounds other than 0-1), one needs an additional intermediate variable. Also, the volume constraint in the compliance problem should be linearized if one wishes to adopt the sequential approximation interpretation. This is not an issue in SIMP since the volume functional is already linear in the design variables.

Matlab implementation

We now describe the Matlab implementation of the discrete topology optimization problem (A.5). The function `PolyTop` is the kernel of the code that contains the optimizer and analysis routines, including the FE routine and functions responsible for computing cost functional and their sensitivities. The prototype is written to solve the compliance minimization problem, but specific functions are designated to compute the objective and constraint functions. All the parameters related to topology optimization that link design variables with the analysis parameters (e.g. filter matrix, material interpolation functions), as well as the finite element model (e.g. the mesh, load and support boundary conditions for the state equation) are defined outside in `PolyScript`, a Matlab script that calls the kernel. This functional decoupling allows the user to run various formulations or discretizations without the need to change the kernel. Also interpolation and regularization functions can be changed easily for the purposes of continuation, for example, on the penalization parameter or filtering radius.

Input data & `PolyScript`

All the input and internal parameters of the code are collected in two Matlab `struct` arrays. One `struct`, called `fem`, contains all the FE-related parameters while the other, `opt`, has the variables pertaining to the topology optimization formulation and optimizer. Table A.2 shows the list of fields in these structure arrays. Note that some of the `fem` fields are populated inside the `PolyTop` kernel unless they are already specified. Also the user has access to all the model parameters since these structures reside in the Matlab workspace.

fem field	Definition
fem.NNode	Number of nodes
fem.NElem	Number of elements
fem.Node	[NNode x 2] array of nodes
fem.Element	[NElem x Var] cell array of elements
fem.Supp	[NSupp x 3] support array
fem.Load	[NLoad x 3] load array
fem.Nu0	Poisson's ratio of solid material
fem.E0	Young's modulus of solid material
fem.Reg	Tag for regular meshes
fem.ElemNDof†	Array showing number of DOFs of elements
fem.ShapeFnc†	Cell array with tabulated shape functions & weights
fem.k†	Array of local stiffness matrix entries
fem.i†	Index array for sparse assembly of fem.k
fem.j†	Index array for sparse assembly of fem.k
fem.e†	Array of element IDs corresponding to fem.k
fem.ElemArea†	Array of element areas
fem.F†	Global load vector
fem.FreeDofs†	Array of free degrees of freedom

opt field	Definition
opt.zMin	Lower bound for design variables
opt.zMax	Upper bound for design variables
opt.zIni	Initial array of design variables
opt.MatIntFnc	Handle to material interpolation function
opt.P	Matrix that maps design to element variables
opt.VolFrac	Specified volume fraction constraint
opt.Tol	Convergence tolerance on design variables
opt.MaxIter	Max. number of optimization iterations
opt.OCMove	Allowable move step in the OC update scheme
opt.OCEta	Exponent used in the OC update scheme

Table A.2: List of fields in the input structures. The fields marked with the superscript †, if empty, are populated inside PolyTop

In the representative implementation of `PolyScript`, the auxiliary functions `PolyMesher` and `PolyFilter` are called to initialize the finite element mesh and construct the linear filtering matrix $\mathbf{P}$ with input radius R. The implementation of `PolyFilter` is short but efficient and is provided as supplementary material. To construct the filtering matrix, this function simply computes the distances between the centroids of the elements in the mesh and defines the filtering weight based on the input filtering radius.

We briefly derive the usual discrete filtering formulas, corresponding to the linear hat filter (1.16), by sampling the integrand in (1.34) at the element centroids:

$$\begin{aligned} w_{\ell k} &= \int_{\Omega_k} F(\mathbf{x}_\ell^*, \overline{\mathbf{x}}) \mathrm{d}\overline{\mathbf{x}} \\ &= c(\mathbf{x}_\ell^*) \int_{\Omega_k} \max\left(1 - \frac{|\mathbf{x}_\ell^* - \overline{\mathbf{x}}|}{R}, 0\right) \mathrm{d}\overline{\mathbf{x}} \\ &\approx c(\mathbf{x}_\ell^*) \, |\Omega_k| \max\left(1 - \frac{|\mathbf{x}_\ell^* - \mathbf{x}_k^*|}{R}, 0\right) \end{aligned} \tag{A.13}$$

If we similarly approximate the integral defining $c(\mathbf{x}_\ell^*)$ and denote by $S(\ell)$ the set of indices of the elements Ω_k whose centroid falls within radius R of the centroid of element Ω_ℓ, i.e., $|\mathbf{x}_\ell^* - \mathbf{x}_k^*| \leq R$

$$y_\ell = \frac{\sum_{k \in S(\ell)} z_k \, |\Omega_k| \, (1 - |\mathbf{x}_\ell^* - \mathbf{x}_k^*| / R)}{\sum_{k \in S(\ell)} |\Omega_k| \, (1 - |\mathbf{x}_\ell^* - \mathbf{x}_k^*| / R)} \tag{A.14}$$

which is the commonly used expression. We note that often the $|\Omega_k|$ terms are omitted because the underlying mesh is uniform. We also reiterate that the vector representation of this expression is more appropriate for implementation in Matlab. Thus, the filter matrix $\mathbf{P}$ given by

$$[\mathbf{P}]_{\ell k} = \frac{\max\left(0, |\Omega_k| \, (1 - |\mathbf{x}_\ell^* - \mathbf{x}_k^*| / R)\right)}{\sum_{k \in S(\ell)} |\Omega_k| \, (1 - |\mathbf{x}_\ell^* - \mathbf{x}_k^*| / R)} \tag{A.15}$$

is stored as a sparse matrix.

If the filtering radius is too small, `PolyFilter` returns the identity matrix. One can also intentionally input a negative value to get the identity matrix. In such a case, each design variable corresponds to an element property—this is sometimes known as the "element-based" approach in the literature and, as discussed before, corresponds to an ill-posed continuum formulation and not surprisingly suffers from mesh-dependency. However, it may be used, for example, to recover Michell-type solutions, provided that numerical instabilities such as checkerboard patterns are suppressed.

Another auxiliary function given is the material interpolation function, whose function handle is passed to the kernel via the `opt.MatIntFnc` field. Given an input vector $\mathbf{y}$, this function must return the vector of the corresponding stiffnesses and volume fractions arrays $\mathbf{E} = m_E(\mathbf{y})$ and $\mathbf{V} = m_V(\mathbf{y})$ of the same length as $\mathbf{y}$ and the sensitivity vectors

$\partial \mathbf{E}/\partial \mathbf{y} := m'_E(\mathbf{y})$ and $\partial \mathbf{V}/\partial \mathbf{y} := m'_V(\mathbf{y})$[3]. For example in the case of SIMP:

```
function [E,dEdy,V,dVdy] = MatIntFnc(y,penal)
  eps = 1e-4;
  E = eps+(1-eps)*y.^penal;
  V = y;
  dEdy = (1-eps)*penal*y.^(penal-1);
  dVdy = ones(size(y,1),1);
```

Note also that the stiffness of the void region can be set here. *This again highlights the fact that the material interpolation model and the Ersatz approximation are independent of the choice of regularization scheme or bounds on the design variables.* The material interpolation function can accept input parameters (e.g. the penalization exponent `penal` in SIMP) so that, if desired, they can be changed by the user in the workspace (for example, for the purpose of continuation). To use another material interpolation function, we only need to change the `MatIntFnc` outside the `PolyTop` kernel.

The continuation on the penalty is implemented in `PolyScript`, outside the `PolyTop` kernel, as follows:

```
for penal = 1:0.5:4
  opt.MatIntFnc = @(y)MatIntFnc(y,'SIMP',penal);
  [opt.zIni,V,fem] = PolyTop(fem,opt);
end
```

This is the default example in `PolyScript` provided in the supplementary material. Upon running the script (for example, by simply entering `PolyScript` in the command prompt), a call is made to `PolyMesher` to generate the mesh, the struct arrays `fem` and `opt` are defined and the `PolyTop` kernel is executed inside this continuation loop. All the results presented in chapter 1 are generated using this setup.

Comments on the functions in `PolyTop`

The kernel `PolyTop` possesses fewer than 190 lines, of which 116 lines pertain to the finite element analysis including 81 lines for the element stiffness calculations for polygonal elements. In this section we explain the implementation of various functions in the kernel.

Main function: The function begins with initialization of the iteration parameters `Iter`, `Tol` and `Change` as well as the initial analysis parameters for the initial guess `z=opt.zIni` on line 8. The function `InitialPlot`, executed on line 10, plots a triangulation of the

[3]Again we understand $m'_E(\mathbf{y})$ and $m'_V(\mathbf{y})$ as vectors with elements $m'_E(y_\ell)$ and $m'_V(y_\ell)$. Essentially $m'_E(\mathbf{y})$ and $m'_V(\mathbf{y})$ are the diagonal entries of Jacobian matrices $J_{m_E}(\mathbf{y})$ and $J_{m_V}(\mathbf{y})$

mesh using the `patch` function and outputs a handle to the figure and also vector `FigData` that will be used to update the patch colors. The iterative optimization algorithm is nested inside the while-loop that terminates when either the maximum number of iterations, `opt.MaxIter`, is exceeded or the change in design variables, `Change`, is smaller than the prescribed tolerance. In each iteration, the current values of `E` and `V` are passed to analysis functions `ObjectiveFnc` and `ConstraintFnc` to, as their names suggest, compute the values and sensitivities of the objective and constraint functions (lines 14-15). The sensitivities with respect to the design variables are computed on lines 17-18 according to expressions (A.8) and (A.12). Note that the entry-by-entry multiplication `dEdy.*dfdE` produces the same vector as the matrix-vector multiplication $J_{m_E}(\mathbf{y})\,\frac{\partial f}{\partial \mathbf{E}}$ because the Jacobian matrix $J_{m_E}(\mathbf{y})$ is diagonal and has `dEdy` as its diagonal elements. The design sensitivities are then passed to `UpdateScheme` to obtain the next vector design variables. The analysis parameters for the new design are computed on line 21. Finally, the new value of the objective function and the maximum change in the current iteration are printed to the screen and element patch colors in the plot are updated to reflect the updated design.

`FEAnalysis:` Before describing the functions responsible for computing the objective and constraint functions, we discuss the implementation of the FE analysis in function `FEAnalysis` and its specific features pertaining to the topology optimization. As noted before, the structure of the global stiffness matrix is such that not only element stiffness matrices are invariant but also the connectivity of the mesh is fixed.

Efficient assembly of the global stiffness matrix in Matlab makes use of the built-in function `sparse` that generates a sparse matrix from an input array of values using two index arrays of the same length. To assemble the global stiffness matrix, we place all the entries of the local stiffness matrices (with the stiffness of reference solid material) in a single vector `fem.k`[4]. The index vectors `fem.i` and `fem.j` contain the global degrees of freedom of each entry in `fem.k`. In particular, this indicates to the `sparse` function that `fem.k(q)` should be placed in row `fem.i(q)` and column `fem.j(q)` of the global stiffness matrix. These vectors are computed and stored on lines 69-79. Our convention is that $2n-1$ and $2n$ are the horizontal and vertical degrees of freedom corresponding to the nth node. By design, the `sparse` function sums the entries in `fem.k` that have the same corresponding indices. To account for the different values of `E` that must be assigned to each element to represent the current design, we compute and store an additional index vector `fem.e`. This array keeps track of the elements to which the local stiffness matrix entries in `fem.k` belong. So the expression `E(fem.e)` returns the elongated list of element stiffnesses that has the

[4]The local stiffness matrices are obtained by calling the function `LocalK` on either line 68 or line 70. If the mesh is known to be uniform, by setting the `fem.Reg` tag equal to 1 in the initialization of the `fem` structure, only one such call is made (on line 68) and thus there is no overhead for repeated calculation of the same element stiffness matrices.

same size as the index vectors. The entry-by-entry multiplication `E(fem.e).*fem.k` then appropriately scales the local stiffness matrix values in `fem.k`. The *assembly* of the global stiffness matrix is therefore accomplished with the following line of code:

```
K = sparse(fem.i,fem.j,E(fem.e).*fem.k);
```

Lines 80-89, executed only during the first iteration, compute the list of free degrees of freedoms `fem.FreeDofs` and global load vector `fem.F` from the given `Supp` and `Load` matrices[5]. Note that only four lines of code are executed in the `FEAnalysis` function to obtain the nodal displacement after the initialization in the first iteration (lines 91-94). We have included line 92, following the recommendation of [12], to ensure that the "backslash" solver recognizes the stiffness matrix `K` as symmetric, which in turn reduces the time for solving the linear system.

`ObjectiveFnc:` This function computes the objective function of the optimization problem using the current values of `E` and `V`. Note again that this function is not given any information related to the optimization formulation. In the prototype implementation, we evaluate the compliance simply using the inner product of the global force and displacement vector, which are computed in call to the `FEAnalysis` function. The function also returns the sensitivity of the objective function with respect to `E` and `V`. In the case of compliance, `dfdV` is the zero vector while `dfdE` is the array consisting of negative of the element strain energies (cf. equation A.9). Since the index vectors `fem.i`, `fem.j`, and `fem.k` contain all the relevant FE information, they can be used to efficiently compute the strain energies. For element Ω_ℓ, we need to compute $-\sum \mathbf{U}_i \left(\mathbf{k}_\ell\right)_{ij} \mathbf{U}_j$ where the sum is taken over all DOFs i and j of element Ω_ℓ. This requires summing the block of `-U(fem.i).*fem.k.*U(fem.j)` that corresponds to element ℓ. Lines 30-33 carry out this computation using `cumsum`, the cumulative sum function in Matlab.

`ConstraintFnc:` This function computes the constraint function of the optimization problem, which for the compliance problem, is the volume fraction constraint (cf. (A.5)). Similar to the initialization step in `FEAnalysis`, the vector of areas is computed only once. Lines 43-45 compute the value and sensitivity of this function with respect to `E` and `V`.

`UpdateScheme:` The input to this function consists of the gradients of the objective and constraint functions `dfdz` and `dgdz`, the current set of design variables `z0`, and the current

[5]These matrices are expected to have the following format: `Supp` must have three columns, the first holding the node number, the second and third columns giving support conditions for that node in the x- and y-direction, respectively. Value of 0 indicates that the node is free, and value of 1 specifies a fixed node. The nodal load vector `Load` is structured in a similar way, except for the values in the second and third columns, which represent the magnitude of the x- and y-components of the force

value of the constraint function `g`. With this information, we can compute, for example, the approximate constraint function: the expression `g+dgdz'*(zNew-z0)` on line 56 gives $g_{\text{app}}(\mathbf{z}^{\text{new}})$, the value of the linearized constraint function at the candidate design variables. The implementation of the update scheme is consistent with the material in Appendix B. The bisection method is used (similar to the 88- and 99-line codes) to solve the dual problem. The move limit M is defined to be `opt.OCMove*(zMax-zMin)`on line 49.

`LocalK:` This function computes the local stiffness matrix of the isoparametric polygonal elements. The implementation and notation used follows the standard conventions[6] in most finite element textbooks (see, for example, [90]). Though closed-form expressions for the polygonal shape functions can be obtained (see the appendix of reference [156]), we use a more costly geometric construction in this code for the sake of brevity, as discussed in section 5.1. However, we eliminate the overhead associated with redundant shape function calculations by computing the needed quantities only once. For an isoparametric element, only the values of shape functions and their gradients at the integration points of the reference element are needed. This can be seen from the quadrature loop on lines 100-110 for computing the local stiffness matrix. The shape function values and the quadrature weights are computed in function `TabShapeFnc` and stored in `fem.ShapeFnc` (the description is given below). Note that this approach only affects the overhead of computing the element stiffness matrices in the initialization process.

`TabShapeFnc:` This function populates the field `fem.ShapeFnc` which contains the tabulated values of shape functions and their gradients at the integration points of the reference element along with the associated quadrature weights. `fem.ShapeFnc` is a cell of length $N_{\max}$, the maximum number of nodes for an element in the input mesh. The nth cell, `fem.ShapeFnc{n}`, is itself a structure array with three fields `N`, `dNdxi`, and `W` whose values are obtained from the reference n-gon (see lines 115-125). The only uses of these shape function values in `PolyTop` are in `LocalK` function on lines 99 and 101.

`PolyShapeFnc:` This function computes the set of linear shape functions for a reference n-gon at an interior point $\boldsymbol{\xi}$. As discussed in chapter 5, the Wachspress shape function corresponding to node i, $1 \leq i \leq n$, is defined as:

$$N_i(\boldsymbol{\xi}) = \frac{\alpha_i(\boldsymbol{\xi})}{\sum_{j=1}^{n} \alpha_j(\boldsymbol{\xi})} \tag{A.16}$$

[6]Regarding the connection between (1.40) and the well-known expression $\int_{\Omega_\ell} \mathbf{B}_I^T \mathbf{D} \mathbf{B}_J \mathrm{d}\mathbf{x}$ for the element stiffness matrix, we refer the reader to section 2.8 of [90].

where, adopting the notation $A_i(\boldsymbol{\xi}) := A(\mathbf{p}_{i-1}, \mathbf{p}_i, \boldsymbol{\xi})$, the interpolants α_i are given by[7]

$$\alpha_i(\boldsymbol{\xi}) = \frac{1}{A_i(\boldsymbol{\xi}) A_{i+1}(\boldsymbol{\xi})} \tag{A.17}$$

In the code, on lines 131-136, the area of the triangles formed by $\boldsymbol{\xi}$ and the vertices as well as their gradient with respect to it are computed using expressions:

$$A_i(\boldsymbol{\xi}) = \frac{1}{2} \begin{vmatrix} \xi_1 & \xi_2 & 1 \\ p_{1,i-1} & p_{2,i-1} & 1 \\ p_{1,i} & p_{2,i} & 1 \end{vmatrix}, \quad \frac{\partial A_i}{\partial \xi_1} = \frac{1}{2}\left(p_{2,i-1} - p_{2,i}\right), \quad \frac{\partial A_i}{\partial \xi_2} = \frac{1}{2}\left(p_{1,i} - p_{1,i-1}\right) \tag{A.18}$$

The derivatives of the interpolant are simply given by (computed on lines 140-141)

$$\frac{\partial \alpha_i}{\partial \xi_k} = -\alpha_i \left(\frac{1}{A_i} \frac{\partial A_i}{\partial \xi_k} + \frac{1}{A_{i+1}} \frac{\partial A_{i+1}}{\partial \xi_k} \right), \quad k = 1, 2 \tag{A.19}$$

and from (A.16) we have the following expression for the shape function gradients (lines 147):

$$\frac{\partial N_i}{\partial \xi_k} = \frac{1}{\sum_{j=1}^{n} \alpha_j} \left(\frac{\partial \alpha_i}{\partial \xi_k} - N_i \sum_{j=1}^{n} \frac{\partial \alpha_j}{\partial \xi_k} \right), \quad k = 1, 2 \tag{A.20}$$

`PolyTrnglt:` This function generates a directed triangulation of the reference n-gon by connecting its vertices to the input point $\boldsymbol{\xi}$ that lies in its interior. As shown in the Figure 5.3, the nodes of the reference n-gon are located at $\mathbf{p}_i = (\cos 2\pi i/n, \sin 2\pi i/n)$. This function is used both in the definition of the polygonal shape functions and the quadrature rule.

`PolyQuad:` One scheme to carry out the integration on the reference n-gon is to divide it into n triangles (by connecting the origin to the vertices) and use well-known quadrature rules on each triangle. For the verification problem in the next section, we have used three integration points per triangle. We note that numerical integration can alternatively be carried out using special quadrature rules recently developed for polygonal domains (see, for example, [115, 116]) that are more accurate.

`TriQuad, TriShape:` These functions are called by `PolyQuad` and provide the usual quadrature rule for the reference triangle and its linear shape functions.

[7] By convention, we set $\mathbf{p}_{n+1} = \mathbf{p}_1$ in this expression

mesh size	90×30	150×50	300×100	600×200
computing $\mathbf{P}$	0.25 (1.6%)	0.876 (2.1%)	9.12 (4.9%)	93.79 (9.2%)
populating `fem`	0.20 (1.3%)	0.50 (1.2%)	2.05 (1.1%)	8.08 (0.8%)
assembling $\mathbf{K}$	5.86 (37.8%)	16.73 (41.1%)	69.50 (37.2%)	289.09 (28.5%)
solving $\mathbf{KU} = \mathbf{F}$	3.56 (23.0%)	11.64 (28.6%)	60.06 (32.1%)	302.95 (29.8%)
mapping $\mathbf{z}$ and $\mathbf{E}, \mathbf{V}$	0.17 (1.1%)	0.86 (2.1%)	12.86 (6.9%)	203.82 (20.1%)
compliance sensitivities	0.94 (6.1%)	2.76 (6.8%)	12.94 (6.9%)	50.82 (5.0%)
plotting the solutions	2.53 (16.3%)	3.15 (7.7%)	6.20 (3.3%)	18.76 (1.8%)
OC update	0.96 (6.2%)	1.99 (4.9%)	5.17 (2.8%)	14.39 (1.4%)
total time of `PolyScript`	15.50	40.74	187.02	1015.89

Table A.3: Breakdown of the code runtime for 200 optimization iterations: times are in seconds with percentage of total runtime of `PolyScript` provided in the parentheses

Efficiency

We discuss the breakdown of the computational cost of the code and compare its efficiency to the 88 line code [12]. For the purposes of comparison, the MBB problem was solved using regular square meshes. A filtering radius of $R = 0.12$ (the height of the MBB beam is $H = 1$) was used[8]. In the case of the 88 line, the input tag `ft` was set to 2 to use the "consistent" density filter. In both cases, the SIMP penalty parameter was fixed at $p = 3$ and 200 optimizations iterations were performed on a machnine with an Intel(R) Core i7, 3.33GHz processor and 24.0GB of RAM running Matlab R2009b. Of course, the two codes produced identical topologies at each iteration.

The breakdown of the runtime of `PolyScript` is shown in Table A.3. We can see that the initialization time, which includes populating the `fem` structure and computing the filtering matrix, were modest for all mesh sizes. During subsequent iterations of `PolyTop`, assembling the stiffness matrix and solving the linear system of FE equations constituted the largest portion of the code runtime. The relative cost of mapping the design variables $\mathbf{z}$ to analysis quantities $\mathbf{E}$ and $\mathbf{V}$ and associated analysis sensitivities (lines 14-15 and 21 of the code) increased with the mesh size. Note that both operations involve multiplication with the filtering matrix which demands more memory for larger meshes. This increase was most significant for the 600×200 mesh perhaps due to the large size of the filtering matrix relative to available memory.

Table A.4 lists the runtime of `PolyScript` and the 88 line code. We note that the present Matlab implementation was faster than the 88 line for every mesh except the

[8]The larger the radius of filtering, the longer it takes to compute the filtering matrix and also the larger the amount of memory needed to store it.

mesh size	90×30	150×50	300×100	600×200
total time of `PolyScript`	15.5	**40.7**	**187**	**1016**
total time of 88 line	**14.8**	44.4	360	4463

Table A.4: Runtime comparison of `PolyScript` with the 88 line code [12] (times are reported in seconds for 200 optimization iterations)

smallest mesh. Also, the difference between the runtime of the two codes became larger as the mesh size grew. This suggests that as the size of the problem increases, `PolyTop` becomes more efficient than the 88 line—notice that there is more than a four-fold speedup for the 600×200 mesh. This observation prompted an investigation to identify the source of discrepancy.

We noticed that in every bisection iteration inside the OC update function of the 88 line code, the design volume is computed by summing the "physical" densities obtained from multiplying the candidate design variables by the filtering matrix. But this volume function can be rewritten as

$$V(\mathbf{z}) = \sum_{\ell=1}^{N} [\mathbf{P}\mathbf{z}]_\ell = \mathbf{1}^T (\mathbf{P}\mathbf{z}) = \left(\mathbf{1}^T\mathbf{P}\right)\mathbf{z} = \left(\mathbf{P}^T\mathbf{1}\right)^T \mathbf{z} \tag{A.21}$$

where $\mathbf{1}$ is the vector of length N with unit entries. Observe that the vector $\mathbf{P}^T\mathbf{1}$ can be computed once so that the calculation of V is subsequently reduced to taking the inner product of this vector with $\mathbf{z}$, thereby eliminating the need for costly multiplications by $\mathbf{P}$ in every bisection step.

Though the above expression is not explicitly used in `PolyTop`, the decoupling of the OC scheme from the analysis routine naturally leads to this more efficient calculation. Note that $\mathbf{P}^T\mathbf{1}$ is in fact the sensitivity vector `dgdz` provided to the `UpdateScheme` function in `PolyTop`[9]. The update equations are based on the linearization of the constraint function $g(\mathbf{z})$ in the form (see section 1.5):

$$g(\mathbf{z}) = g\left(\mathbf{z}^0\right) + \left(\mathbf{z} - \mathbf{z}^0\right)^T \nabla g\left(\mathbf{z}^0\right) \tag{A.22}$$

where $\mathbf{z}^0$ is the design variables from the current iteration. Since the volume constraint is linear, this is identical to the expression for the volume function (the reader can verify the equivalence of the two expressions). This observation is perhaps further illustration of the virtue of decoupling philosophy advocated here.

We conclude with a remark regarding the overhead associated with computing multi-

[9]Note, however, that in `PolyTop` the volume function is normalized by the volume of the entire domain and elements can have different areas (so $\mathbf{1}$ is replaced by $\mathbf{A}$ in `PolyTop`). Also, the constraint function is defined as the difference between the normalized volume and specified volume fraction $\overline{v}$.

ple element stiffness matrices. As mentioned before, the initialization of `fem.k` was done computing the element stiffness matrix only once (by setting `fem.Reg=1)` for the purposes of comparison with the 88 line. However, even without the use of `fem.Reg` tag, the cost associated with the repeated element stiffness matrix calculations as a percentage of total cost is small. For example, for the mesh of 300×100 quads, this took 6.12s which constituted 3.3% of the total cost of 200 optimization iterations. Likewise, for a polygonal mesh of 10,000 elements, the element matrix calculations took 8.39s which was 5.8% of the total cost of 200 optimization iterations.

Implementation of the two-metric projection algorithm

Finally we briefly discuss some implementation aspects of the two-metric projection algorithm for the Tikhonov regularized topology optimization problem (cf. section 4.8) as a representative of the operator splitting approaches developed in chapters 3 and 4. In particular, we will discuss a simple implementation in the framework of `PolyTop`. Note that unlike the forward-backward splitting method, which requires a sparse quadratic programming optimizer (an external library was used to generate the results in the thesis), the two-metric projection, owing to its use of a more convenient metric for projection, can be readily and efficiently implemented in Matlab.

The design variables in this formulation are the nodal density fields and so the matrix $\mathbf{P}$ in `opt.P` represents the node-to-element mapping with its (e,k)-entry given by $[\mathbf{P}]_{ek} = N_k(\mathbf{x}_e^*)$. As with the filtering matrix, this should be computed outside of `PolyTop` once in a pre-processing step. To remain consistent with the formulation presented in chapter 4, the volume function is defined as $g(\mathbf{z}) = \mathbf{A}^T m_V(\mathbf{P}\mathbf{z})$ and in place of a volume fraction parameter, the user will specify the value of the penalty parameter λ, which we assume is stored in `opt.lambda`.

To define the regularization term, a function must be added to the `PolyTop` code to compute the matrix $\mathbf{G}$ defined in section 4.6, which requires that the user defines the regularization parameter β. Storing $\mathbf{G}$ in field `opt.G`, the Tikhonov regularizer and its gradient can be computed in each iteration (in the while-loop on lines 11-25) as

```
R=0.5*z'*opt.G*z; dRdz=opt.G*z;
```

The other input to the optimizer is the Hessian approximation $\mathbf{H}_n$ as defined in equation (4.23). Its diagonal entries can be computed from `dfdz` in each iteration as

```
h_hat = max(max(-2*dfdz./z,1e-8);
```

From this, we compute and store $\mathbf{H}_n$ in the `opt` struct in order to pass it to the `UpdateScheme` function using the following line of code

```
opt.H = spdiags(h_hat(:),0,fem.NNode,fem.NNode);
```

The main change from the standard `PolyTop` code is the replacement of the current OC-based `UpdateScheme` routine by the two-metric projection algorithm. Here we consider the special case of a fixed step size parameter, i.e., $\tau_n \equiv \tau_0$ but implementation of the descent condition (4.74) is straightforward. We remark, however, that the from a conceptual point of view, this requires the optimizer to calculate the value of objective function for each new candidate design, which in turn would need access to the `fem` structure. One approach that preserves the modularity of the code is to check the descent condition and update the step size parameter in a loop in the main function of `PolyTop` and outside of `UpdateScheme`. In this case, the `UpdateScheme` performs an update for the input value of `tau` with the call

```
[z,Change] = UpdateScheme(dfdz,dgdz,dRdz,z,opt,tau);
```

Inside this function, we first compute the gradient of the composite objective $\nabla \tilde{J}$ which is given by

```
dJtdz = dfdz+opt.lambda*dgdz+dRdz;
```

Then the indices for the active and free constraints are obtained as follows

```
ActCnstr = find((z0-zMin<=eps&dJtdz>0)|( zMax-z0<=eps&dJtdz<0));
FreeCnstr = setdiff(1:length(z0),ActCnstr);
```

The scaling matrix $\mathbf{D}_n$, stored in variable `D`, is the diagonal matrix with respect to the active constraints obtained from $\mathbf{H}_n + \tau_n \mathbf{G}$ and can be computed as follows

```
D_full = opt.H+tau*opt.G;
D = spdiags(spdiags(D_full,0),0,length(z0),length(z0));
D(FreeCnstr,FreeCnstr) = D_full(FreeCnstr,FreeCnstr);
```

The interim update, defined according to equation (4.71), is simply

```
zCnd = z0-tau*(D\dJtdz);
```

and its projection onto the constraint set gives the next iterate

```
zNew = max(max(min(min(zCnd,z0+move),zMax),z0-move),zMin);
```

Note that this is identical to line 55 of the current `PolyTop` code.

Code: PolyMesher

```
%------------------------------ PolyMesher --------------------------------%
% Ref: C Talischi, GH Paulino, A Pereira, IFM Menezes, "PolyMesher: A     %
%      general-purpose mesh generator for polygonal elements written in   %
%      Matlab," Struct Multidisc Optim, Vo. 42, pp. 309-328, 2012         %
%--------------------------------------------------------------------------%
function [Node,Element,Supp,Load,P] = PolyMesher(Domain,NElem,MaxIter,P)
if ¬exist('P','var'), P=PolyMshr_RndPtSet(NElem,Domain); end
NElem = size(P,1);
Tol=5e-3; It=0; Err=1; c=1.5;
BdBox = Domain('BdBox');
Area = (BdBox(2)-BdBox(1))*(BdBox(4)-BdBox(3));
Pc = P; figure;
while(It≤MaxIter && Err>Tol)
  Alpha = c*sqrt(Area/NElem);
  P = Pc;                                          %Lloyd's update
  R_P = PolyMshr_Rflct(P,NElem,Domain,Alpha);      %Generate the reflections
  [Node,Element] = voronoin([P;R_P]);              %Construct Voronoi diagram
  [Pc,A] = PolyMshr_CntrdPly(Element,Node,NElem);
  Area = sum(abs(A));
  Err = sqrt(sum((A.^2).*sum((Pc-P).*(Pc-P),2)))*NElem/Area^1.5;
  fprintf('It: %3d   Error: %1.3e\n',It,Err); It=It+1;
  if NElem≤2000, PolyMshr_PlotMsh(Node,Element,NElem); end;
end
[Node,Element] = PolyMshr_ExtrNds(NElem,Node,Element);  %Extract node list
[Node,Element] = PolyMshr_CllpsEdgs(Node,Element,0.1);  %Remove small edges
[Node,Element] = PolyMshr_RsqsNds(Node,Element);        %Reoder Nodes
BC=Domain('BC',Node); Supp=BC{1}; Load=BC{2};           %Recover BC arrays
PolyMshr_PlotMsh(Node,Element,NElem,Supp,Load);         %Plot mesh and BCs
%-------------------------------------------------- GENERATE RANDOM POINTSET
function P = PolyMshr_RndPtSet(NElem,Domain)
P=zeros(NElem,2); BdBox=Domain('BdBox'); Ctr=0;
while Ctr<NElem
  Y(:,1) = (BdBox(2)-BdBox(1))*rand(NElem,1)+BdBox(1);
  Y(:,2) = (BdBox(4)-BdBox(3))*rand(NElem,1)+BdBox(3);
  d = Domain('Dist',Y);
  I = find(d(:,end)<0);                  %Index of seeds inside the domain
  NumAdded = min(NElem-Ctr,length(I));  %Number of seeds that can be added
  P(Ctr+1:Ctr+NumAdded,:) = Y(I(1:NumAdded),:);
  Ctr = Ctr+NumAdded;
end
%------------------------------------------------------------ REFLECT POINTSET
function R_P = PolyMshr_Rflct(P,NElem,Domain,Alpha)
eps=1e-8; eta=0.9;
d = Domain('Dist',P);
NBdrySegs = size(d,2)-1;          %Number of constituent bdry segments
n1 = (Domain('Dist',P+repmat([eps,0],NElem,1))-d)/eps;
n2 = (Domain('Dist',P+repmat([0,eps],NElem,1))-d)/eps;
I = abs(d(:,1:NBdrySegs))<Alpha;  %Logical index of seeds near the bdry
P1 = repmat(P(:,1),1,NBdrySegs);  %[NElem x NBdrySegs] extension of P(:,1)
P2 = repmat(P(:,2),1,NBdrySegs);  %[NElem x NBdrySegs] extension of P(:,2)
R_P(:,1) = P1(I)-2*n1(I).*d(I);
R_P(:,2) = P2(I)-2*n2(I).*d(I);
d_R_P = Domain('Dist',R_P);
J = abs(d_R_P(:,end))≥eta*abs(d(I)) & d_R_P(:,end)>0;
```

```
R_P = R_P(J,:);
%------------------------------------------- COMPUTE CENTROID OF POLYGON
function [Pc,A] = PolyMshr_CntrdPly(Element,Node,NElem)
Pc=zeros(NElem,2); A=zeros(NElem,1);
for el = 1:NElem
  vx=Node(Element{el},1); vy=Node(Element{el},2); nv=length(Element{el});
  vxS=vx([2:nv 1]); vyS=vy([2:nv 1]); %Shifted vertices
  temp = vx.*vyS - vy.*vxS;
  A(el) = 0.5*sum(temp);
  Pc(el,:) = 1/(6*A(el,1))*[sum((vx+vxS).*temp),sum((vy+vyS).*temp)];
end
%------------------------------------------------------ EXTRACT MESH NODES
function [Node,Element] = PolyMshr_ExtrNds(NElem,Node0,Element0)
map = unique([Element0{1:NElem}]);
cNode = 1:size(Node0,1);
cNode(setdiff(cNode,map)) = max(map);
[Node,Element] = PolyMshr_RbldLists(Node0,Element0(1:NElem),cNode);
%------------------------------------------------- COLLAPSE SMALL EDGES
function [Node0,Element0] = PolyMshr_CllpsEdgs(Node0,Element0,Tol)
while(true)
  cEdge = [];
  for el=1:size(Element0,1)
    if size(Element0{el},2)<4, continue; end;  %Cannot collapse triangles
    vx=Node0(Element0{el},1); vy=Node0(Element0{el},2); nv=length(vx);
    beta = atan2(vy-sum(vy)/nv, vx-sum(vx)/nv);
    beta = mod(beta([2:end 1]) -beta,2*pi);
    betaIdeal = 2*pi/size(Element0{el},2);
    Edge = [Element0{el}',Element0{el}([2:end 1])'];
    cEdge = [cEdge; Edge(beta<Tol*betaIdeal,:)];
  end
  if (size(cEdge,1)==0), break; end
  cEdge = unique(sort(cEdge,2),'rows');
  cNode = 1:size(Node0,1);
  for i=1:size(cEdge,1)
    cNode(cEdge(i,2)) = cNode(cEdge(i,1));
  end
  [Node0,Element0] = PolyMshr_RbldLists(Node0,Element0,cNode);
end
%------------------------------------------------------ RESEQUENSE NODES
function [Node,Element] = PolyMshr_RsqsNds(Node0,Element0)
NNode0=size(Node0,1); NElem0=size(Element0,1);
ElemLnght=cellfun(@length,Element0); nn=sum(ElemLnght.^2);
i=zeros(nn,1); j=zeros(nn,1); s=zeros(nn,1); index=0;
for el = 1:NElem0
  eNode=Element0{el}; ElemSet=index+1:index+ElemLnght(el)^2;
  i(ElemSet) = kron(eNode,ones(ElemLnght(el),1))';
  j(ElemSet) = kron(eNode,ones(1,ElemLnght(el)))';
  s(ElemSet) = 1;
  index = index+ElemLnght(el)^2;
end
K = sparse(i,j,s,NNode0, NNode0);
p = symrcm(K);
cNode(p(1:NNode0))=1:NNode0;
[Node,Element] = PolyMshr_RbldLists(Node0,Element0,cNode);
%------------------------------------------------------ REBUILD LISTS
function [Node,Element] = PolyMshr_RbldLists(Node0,Element0,cNode)
Element = cell(size(Element0,1),1);
[foo,ix,jx] = unique(cNode);
```

```
113 if size(Node0,1)>length(ix), ix(end)=max(cNode); end;
114 Node = Node0(ix,:);
115 for el=1:size(Element0,1)
116   Element{el} = unique(jx(Element0{el}));
117   vx=Node(Element{el},1); vy=Node(Element{el},2); nv=length(vx);
118   [foo,iix] = sort(atan2(vy-sum(vy)/nv,vx-sum(vx)/nv));
119   Element{el} = Element{el}(iix);
120 end
121 %-------------------------------------------------------------------- PLOT MESH
122 function PolyMshr_PlotMsh(Node,Element,NElem,Supp,Load)
123 clf; axis equal; axis off; hold on;
124 Element = Element(1:NElem)';                 %Only plot the first block
125 MaxNVer = max(cellfun(@numel,Element));      %Max. num. of vertices in mesh
126 PadWNaN = @(E) [E NaN(1,MaxNVer-numel(E))];  %Pad cells with NaN
127 ElemMat = cellfun(PadWNaN,Element,'UniformOutput',false);
128 ElemMat = vertcat(ElemMat{:});               %Create padded element matrix
129 patch('Faces',ElemMat,'Vertices',Node,'FaceColor','w'); pause(1e-6)
130 if exist('Supp','var')&&¬isempty(Supp)&&¬isempty(Load)%Plot BC if specified
131   plot(Node(Supp(:,1),1),Node(Supp(:,1),2),'b>','MarkerSize',8);
132   plot(Node(Load(:,1),1),Node(Load(:,1),2),'m^','MarkerSize',8); hold off;
133 end
134 %-------------------------------------------------------------------------%
```

Code: PolyScript

```
%------------------------------ PolyScript ------------------------------%
% Ref: C Talischi, GH Paulino, A Pereira, IFM Menezes, "PolyTop: A Matlab %
% implementation of a general topology optimization framework using       %
% unstructured polygonal finite element meshes", Struct Multidisc Optim,  %
% Vo. 45, pp. 329-357, 2012                                               %
%-------------------------------------------------------------------------%

%% ------------------------------------------------- CREATE 'fem' STRUCT
[Node,Element,Supp,Load] = PolyMesher(@MbbDomain,5000,30);
fem = struct(...
  'NNode',size(Node,1),...      % Number of nodes
  'NElem',size(Element,1),...   % Number of elements
  'Node',Node,...               % [NNode x 2] array of nodes
  'Element',{Element},...       % [NElement x Var] cell array of elements
  'Supp',Supp,...               % Array of supports
  'Load',Load,...               % Array of loads
  'Nu0',0.3,...                 % Poisson's ratio of solid material
  'E0',1.0,...                  % Young's modulus of solid material
  'Reg',0 ...                   % Tag for regular meshes
   );
%% ------------------------------------------------- CREATE 'opt' STRUCT
R = 0.04;
VolFrac = 0.5;
m = @(y)MatIntFnc(y,'SIMP',3);
P = PolyFilter(fem,R);
zIni = VolFrac*ones(size(P,2),1);
opt = struct(...
  'zMin',0.0,...                % Lower bound for design variables
  'zMax',1.0,...                % Upper bound for design variables
  'zIni',zIni,...               % Initial design variables
  'MatIntFnc',m,...             % Handle to material interpolation fnc.
  'P',P,...                     % Matrix that maps design to element vars.
  'VolFrac',VolFrac,...         % Specified volume fraction cosntraint
  'Tol',0.01,...                % Convergence tolerance on design vars.
  'MaxIter',150,...             % Max. number of optimization iterations
  'OCMove',0.2,...              % Allowable move step in OC update scheme
  'OCEta',0.5 ...               % Exponent used in OC update scheme
   );
%% ------------------------------------------------------- RUN 'PolyTop'
figure;
for penal = 1:0.5:4        %Continuation on the penalty parameter
   disp(['current p: ', num2str(penal)]);
   opt.MatIntFnc = @(y)MatIntFnc(y,'SIMP',penal);
   [opt.zIni,V,fem] = PolyTop(fem,opt);
end
%% -------------------------------------------------------------------------
```

Code: PolyTop

```
%------------------------------ PolyTop ----------------------------------%
% Ref: C Talischi, GH Paulino, A Pereira, IFM Menezes, "PolyTop: A Matlab %
% implementation of a general topology optimization framework using       %
% unstructured polygonal finite element meshes", Struct Multidisc Optim,  %
% Vo. 45, pp. 329-357, 2012                                               %
%-------------------------------------------------------------------------%
function [z,V,fem] = PolyTop(fem,opt)
Iter=0; Tol=opt.Tol*(opt.zMax-opt.zMin); Change=2*Tol; z=opt.zIni; P=opt.P;
[E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
[FigHandle,FigData] = InitialPlot(fem,z);
while (Iter<opt.MaxIter) && (Change>Tol)
  Iter = Iter + 1;
  %Compute cost functionals and analysis sensitivities
  [f,dfdE,dfdV,fem] = ObjectiveFnc(fem,E,V);
  [g,dgdE,dgdV,fem] = ConstraintFnc(fem,E,V,opt.VolFrac);
  %Compute design sensitivities
  dfdz = P'*(dEdy.*dfdE + dVdy.*dfdV);
  dgdz = P'*(dEdy.*dgdE + dVdy.*dgdV);
  %Update design variable and analysis parameters
  [z,Change] = UpdateScheme(dfdz,g,dgdz,z,opt);
  [E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
  %Output results
  fprintf('It: %i \t Objective: %1.3f\tChange: %1.3f\n',Iter,f,Change);
  set(FigHandle,'FaceColor','flat','CData',1-V(FigData)); drawnow
end
%-------------------------------------------------------- OBJECTIVE FUNCTION
function [f,dfdE,dfdV,fem] = ObjectiveFnc(fem,E,V)
[U,fem] = FEAnalysis(fem,E);
f = dot(fem.F,U);
temp = cumsum(-U(fem.i).*fem.k.*U(fem.j));
temp = temp(cumsum(fem.ElemNDof.^2));
dfdE = [temp(1);temp(2:end)-temp(1:end-1)];
dfdV = zeros(size(V));
%------------------------------------------------------ CONSTRAINT FUNCTION
function [g,dgdE,dgdV,fem] = ConstraintFnc(fem,E,V,VolFrac)
if ¬isfield(fem,'ElemArea')
  fem.ElemArea = zeros(fem.NElem,1);
  for el=1:fem.NElem
    vx=fem.Node(fem.Element{el},1); vy=fem.Node(fem.Element{el},2);
    fem.ElemArea(el) = 0.5*sum(vx.*vy([2:end 1])-vy.*vx([2:end 1]));
  end
end
g = sum(fem.ElemArea.*V)/sum(fem.ElemArea)-VolFrac;
dgdE = zeros(size(E));
dgdV = fem.ElemArea/sum(fem.ElemArea);
%------------------------------------------- OPTIMALITY CRITERIA UPDATE
function [zNew,Change] = UpdateScheme(dfdz,g,dgdz,z0,opt)
zMin=opt.zMin; zMax=opt.zMax;
move=opt.OCMove*(zMax-zMin); eta=opt.OCEta;
l1=0; l2=1e6;
while l2-l1 > 1e-4
  lmid = 0.5*(l1+l2);
  B = -(dfdz./dgdz)/lmid;
  zCnd = zMin+(z0-zMin).*B.^eta;
```

```
  zNew = max(max(min(min(zCnd,z0+move),zMax),z0-move),zMin);
  if (g+dgdz'*(zNew-z0)>0),  l1=lmid;
  else                       l2=lmid;  end
end
Change = max(abs(zNew-z0))/(zMax-zMin);
%------------------------------------------------------------- FE-ANALYSIS
function [U,fem] = FEAnalysis(fem,E)
if ¬isfield(fem,'k')
  fem.ElemNDof = 2*cellfun(@length,fem.Element); % # of DOFs per element
  fem.i = zeros(sum(fem.ElemNDof.^2),1);
  fem.j=fem.i; fem.k=fem.i; fem.e=fem.i;
  index = 0;
  if ¬isfield(fem,'ShapeFnc'), fem=TabShapeFnc(fem); end
  if fem.Reg, Ke=LocalK(fem,fem.Element{1}); end
  for el = 1:fem.NElem
    if ¬fem.Reg,  Ke=LocalK(fem,fem.Element{el}); end
    NDof = fem.ElemNDof(el);
    eDof = reshape([2*fem.Element{el}-1;2*fem.Element{el}],NDof,1);
    I=repmat(eDof ,1,NDof); J=I';
    fem.i(index+1:index+NDof^2) = I(:);
    fem.j(index+1:index+NDof^2) = J(:);
    fem.k(index+1:index+NDof^2) = Ke(:);
    fem.e(index+1:index+NDof^2) = el;
    index = index + NDof^2;
  end
  NLoad = size(fem.Load,1);
  fem.F = zeros(2*fem.NNode,1);  %external load vector
  fem.F(2*fem.Load(1:NLoad,1)-1) = fem.Load(1:NLoad,2);  %x-crdnt
  fem.F(2*fem.Load(1:NLoad,1))   = fem.Load(1:NLoad,3);  %y-crdnt
  NSupp = size(fem.Supp,1);
  FixedDofs = [fem.Supp(1:NSupp,2).*(2*fem.Supp(1:NSupp,1)-1);
               fem.Supp(1:NSupp,3).*(2*fem.Supp(1:NSupp,1))];
  FixedDofs = FixedDofs(FixedDofs>0);
  AllDofs   = [1:2*fem.NNode];
  fem.FreeDofs = setdiff(AllDofs,FixedDofs);
end
K = sparse(fem.i,fem.j,E(fem.e).*fem.k);
K = (K+K')/2;
U = zeros(2*fem.NNode,1);
U(fem.FreeDofs,:) = K(fem.FreeDofs,fem.FreeDofs)\fem.F(fem.FreeDofs,:);
%------------------------------------------------- ELEMENT STIFFNESS MATRIX
function [Ke] = LocalK(fem,eNode)
D=1/(1-fem.Nu0^2)*[1 fem.Nu0 0;fem.Nu0 1 0;0 0 (1-fem.Nu0)/2];%plane stress
nn=length(eNode); Ke=zeros(2*nn,2*nn);
W = fem.ShapeFnc{nn}.W;
for q = 1:length(W)  %quadrature loop
  dNdxi = fem.ShapeFnc{nn}.dNdxi(:,:,q);
  J0 = fem.Node(eNode,:)'*dNdxi;
  dNdx = dNdxi/J0;
  B = zeros(3,2*nn);
  B(1,1:2:2*nn) = dNdx(:,1)';
  B(2,2:2:2*nn) = dNdx(:,2)';
  B(3,1:2:2*nn) = dNdx(:,2)';
  B(3,2:2:2*nn) = dNdx(:,1)';
  Ke = Ke+B'*D*B*W(q)*det(J0);
end
%------------------------------------------------ TABULATE SHAPE FUNCTIONS
function fem = TabShapeFnc(fem)
```

```
ElemNNode = cellfun(@length,fem.Element); % number of nodes per element
fem.ShapeFnc = cell(max(ElemNNode),1);
for nn = min(ElemNNode):max(ElemNNode)
  [W,Q] = PolyQuad(nn);
  fem.ShapeFnc{nn}.W = W;
  fem.ShapeFnc{nn}.N = zeros(nn,1,size(W,1));
  fem.ShapeFnc{nn}.dNdxi = zeros(nn,2,size(W,1));
  for q = 1:size(W,1)
    [N,dNdxi] = PolyShapeFnc(nn,Q(q,:));
    fem.ShapeFnc{nn}.N(:,:,q) = N;
    fem.ShapeFnc{nn}.dNdxi(:,:,q) = dNdxi;
  end
end
%------------------------------------------------ POLYGONAL SHAPE FUNCTIONS
function [N,dNdxi] = PolyShapeFnc(nn,xi)
N=zeros(nn,1); alpha=zeros(nn,1); dNdxi=zeros(nn,2); dalpha=zeros(nn,2);
sum_alpha=0.0; sum_dalpha=zeros(1,2); A=zeros(nn,1); dA=zeros(nn,2);
[p,Tri] = PolyTrnglt(nn,xi);
for i=1:nn
  sctr = Tri(i,:); pT = p(sctr,:);
  A(i) = 1/2*det([pT,ones(3,1)]);
  dA(i,1) = 1/2*(pT(3,2)-pT(2,2));
  dA(i,2) = 1/2*(pT(2,1)-pT(3,1));
end
A=[A(nn,:);A]; dA=[dA(nn,:);dA];
for i=1:nn
  alpha(i) = 1/(A(i)*A(i+1));
  dalpha(i,1) = -alpha(i)*(dA(i,1)/A(i)+dA(i+1,1)/A(i+1));
  dalpha(i,2) = -alpha(i)*(dA(i,2)/A(i)+dA(i+1,2)/A(i+1));
  sum_alpha = sum_alpha + alpha(i);
  sum_dalpha(1:2) = sum_dalpha(1:2)+dalpha(i,1:2);
end
for i=1:nn
  N(i) = alpha(i)/sum_alpha;
  dNdxi(i,1:2) = (dalpha(i,1:2)-N(i)*sum_dalpha(1:2))/sum_alpha;
end
%---------------------------------------------------- POLYGON TRIANGULATION
function [p,Tri] = PolyTrnglt(nn,xi)
p = [cos(2*pi*((1:nn))/nn); sin(2*pi*((1:nn))/nn)]';
p = [p; xi];
Tri = zeros(nn,3); Tri(1:nn,1)=nn+1;
Tri(1:nn,2)=1:nn; Tri(1:nn,3)=2:nn+1; Tri(nn,3)=1;
%------------------------------------------------------ POLYGONAL QUADRATURE
function [weight,point] = PolyQuad(nn)
[W,Q]= TriQuad;                  %integration pnts & wgts for ref. triangle
[p,Tri] = PolyTrnglt(nn,[0 0]);  %triangulate from origin
point=zeros(nn*length(W),2); weight=zeros(nn*length(W),1);
for k=1:nn
  sctr = Tri(k,:);
  for q=1:length(W)
    [N,dNds] = TriShapeFnc(Q(q,:));  %compute shape functions
    J0 = p(sctr,:)'*dNds;
    l = (k-1)*length(W) + q;
    point(l,:) = N'*p(sctr,:);
    weight(l) = det(J0)*W(q);
  end
end
%---------------------------------------------------- TRIANGULAR QUADRATURE
```

```
function [weight,point] = TriQuad
point=[1/6,1/6;2/3,1/6;1/6,2/3]; weight=[1/6,1/6,1/6];
%------------------------------------------- TRIANGULAR SHAPE FUNCTIONS
function [N,dNds] = TriShapeFnc(s)
N=[1-s(1)-s(2);s(1);s(2)]; dNds=[-1,-1;1,0;0,1];
%---------------------------------------------------------- INITIAL PLOT
function [handle,map] = InitialPlot(fem,z0)
Tri = zeros(length([fem.Element{:}])-2*fem.NElem,3);
map = zeros(size(Tri,1),1); index=0;
for el = 1:fem.NElem
  for enode = 1:length(fem.Element{el})-2
    map(index+1) = el;
    Tri(index+1,:) = fem.Element{el}([1,enode+1,enode+2]);
    index = index + 1;
  end
end
handle = patch('Faces',Tri,'Vertices',fem.Node,'FaceVertexCData',...
               1-z0(map),'FaceColor','flat','EdgeColor','none');
axis equal; axis off; axis tight; colormap(gray);
%-------------------------------------------------------------------------%
```

Code: PolyFilter

```
%--------------------------------------------------------------------------%
function [P] = PolyFilter(fem,R)
if R<0, P = speye(fem.NElem); return; end %P is set to identity when R<0
ElemCtrd = zeros(fem.NElem,2);
for el = 1:fem.NElem        %Compute the centroids of all the elements
  vx=fem.Node(fem.Element{el},1); vy=fem.Node(fem.Element{el},2);
  temp = vx.*vy([2:end 1])-vy.*vx([2:end 1]);
  A = 0.5*sum(temp);
  ElemCtrd(el,1) = 1/(6*A)*sum((vx+vx([2:end 1])).*temp);
  ElemCtrd(el,2) = 1/(6*A)*sum((vy+vy([2:end 1])).*temp);
end
[d] = DistPntSets(ElemCtrd,ElemCtrd,R);  %Obtain distance values & indices
P = sparse(d(:,1),d(:,2),1-d(:,3)/R);    %Assemble the filtering matrix
P = spdiags(1./sum(P,2),0,fem.NElem,fem.NElem)*P;
%--------------------------------- COMPUTE DISTANCE BETWEEN TWO POINT SETS
function [d] = DistPntSets(PS1,PS2,R)
d = cell(size(PS1,1),1);
for el = 1:size(PS1,1)       %Compute the distance information
    dist = sqrt((PS1(el,1)-PS2(:,1)).^2 + (PS1(el,2)-PS2(:,2)).^2);
    [I,J] = find(dist≤R);   %Find the indices for distances less that R
    d{el} = [I,J+(el-1),dist(I)];
end
d = cell2mat(d);             %Matrix of indices and distance value
%--------------------------------------------------------------------------%
```

Bibliography

[1] R. Adams and J. Fournier, *Sobolev Spaces*, Academic Press, 2nd ed., 2003.

[2] M. Adlers, *Sparse Least Squares Problems with Box Constraints*, Department of Mathematics, Linkoping University, Thesis, 1998.

[3] G. Allaire, *Shape Optimization by the Homogenization Method*, Springer, New York, 2001.

[4] G. Allaire, E. Bonnetier, G. A. Francfort, and F. Jouve, *Shape optimization by the homogenization method*, Numer Math, 76 (1997), pp. 27–68.

[5] G. Allaire and G. A. Francfort, *Existence of minimizers for non-quasiconvex functionals arising in optimal design*, Ann I H Poincare-An, 15 (1998), pp. 301–339.

[6] G. Allaire and A. Henrot, *On some recent advances in shape optimization*, Jan 2001.

[7] G. Allaire and F. Jouve, *A level-set method for vibration and multiple loads structural optimization*, Comput Methods Appl Mech Engrg, 194 (2005), pp. 3269–3290.

[8] G. Allaire, F. Jouve, and A.-M. Toader, *Structural optimization using sensitivity analysis and a level-set method*, J Comput Phys, 194 (2004), pp. 363–393.

[9] G. Allaire and O. Pantz, *Structural optimization with FreeFem++*, Struct Multidisc Optim, 32 (2006), pp. 173–181.

[10] S. R. M. Almeida, G. H. Paulino, and E. C. N. Silva, *Layout and material gradation in topology optimization of functionally graded structures: a global-local approach*, Struct Multidisc Optim, 42 (2010), pp. 885–868.

[11] L. Ambrosio and G. Buttazzo, *An optimal design problem with perimeter penalization*, Calc Var Partial Dif, 1 (1993), pp. 55–69.

[12] E. Andreassen, A. Clausen, M. Schevenels, B. Lazarov, and O. Sigmund, *Efficient topology optimization in MATLAB using 88 lines of code*, Struct Multidisc Optim, 43 (2011), pp. 1–16.

[13] J. S. Arora, *Analysis of optimality criteria and gradient projection methods for optimal structural design*, Comput Methods Appl Mech Engrg, 23 (1980), pp. 185–213.

[14] U. M. Ascher, H. Huang, and K. van den Doel, *Artificial time integration*, BIT, 47 (2007), pp. 3–25.

[15] F. Aurenhammer, *Voronoi diagrams–a survey of a fundamental geometric data structure*, Comput Surv, 23 (1991), pp. 345–405.

[16] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J Image Sci, 2 (2008), pp. 183–202.

[17] ——, *Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems*, IEEE Trans Image Proc, 18 (2009), pp. 2419–2434.

[18] ——, *Gradient-based algorithms with applications to signal recovery problems*, in Convex Optimization in Signal Processing and Communications, Cambridge university press, 2010.

[19] T. Belytschko, S. P. Xiao, and C. Parimi, *Topology optimization with implicit functions and regularization*, Int J Numer Meth Engng, 57 (2003), pp. 1177–1196.

[20] M. P. Bendsøe, *Optimal design as material distribution probelm*, Struct Optimization, 1 (1989), pp. 193–202.

[21] M. P. Bendsøe and R. B. Haber, *The Michell layout problem as a low volume fraction limit of the perforated plate topology optimization problem: An asymptotic study*, Struct Optimization, 6 (1993), pp. 263–267.

[22] M. P. Bendsøe and N. Kikuchi, *Generating optimal topologies in structural design using a homogenization method*, Comput Methods Appl Mech Engrg, 71 (1988), pp. 197–224.

[23] M. P. Bendsøe and O. Sigmund, *Material interpolation schemes in topology optimization*, Arch Appl Mech, 69 (1999), pp. 635–654.

[24] ——, *Topology Optimization: Theory, Methods and Applications*, Springer, 2003.

[25] D. P. Bertsekas, *Projected newton methods for optimization problems with simple constraints*, SIAM J Control Opt, 20 (1982), pp. 221–246.

[26] ——, *Nonlinear Programming*, Athena Scientific, 2nd ed., 1999.

[27] J. E. Bolander and S. Saito, *Fracture analyses using spring networks with random geometry*, Eng Fract Mech, 61 (1998), pp. 569–591.

[28] T. Borrvall, *Topology optimization of elastic continua using restriction*, Arch Comput Method E, 8 (2001), pp. 251–285.

[29] T. Borrvall and J. Petersson, *Topology optimization using regularized intermediate density control*, Comput Methods Appl Mech Engrg, 190 (2001), pp. 4911–4928.

[30] ——, *Topology optimization of fluids in stokes flow*, Int J Numer Meth Eng, 41 (2003), pp. 77–107.

[31] B. Bourdin, *Filters in topology optimization*, Int J Numer Meth Eng, 50 (2001), pp. 2143–2158.

[32] B. Bourdin and A. Chambolle, *Design-dependent loads in topology optimization*, ESAIM Contr Optim Ca, 9 (2003), pp. 19–48.

[33] K. Bredies, *A forward–backward splitting algorithm for the minimization of non-smooth convex functionals in Banach space*, Inverse Probl, 25 (2009), p. 015005.

[34] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, 2nd ed., 2002.

[35] F. Brezzi and M. Fortin, *Mixed and hybrid finite element method*, Springer, New York, 1991.

[36] T. E. Bruns, *A reevaluation of the SIMP method with filtering and an alternative formulation for solid-void topology optimization*, Struct Multidisc Optim, 30 (2005), pp. 428–436.

[37] M. Bruyneel and P. Duysinx, *Note on topology optimization of continuum structures including self-weight*, Struct Multidisc Optim, 29 (2005), pp. 245–256.

[38] M. Burger and R. Stainko, *Phase-field relaxation of topology optimization with local stress constraints*, SIAM J Control Optim, 45 (2006), pp. 1447–1466.

[39] P. H. Calamai and J. J. Moré, *Projected gradient methods for linearly constrained problems*, Math Program, 39 (1987), pp. 93–116.

[40] E. J. Candes, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Comm Pure Appl Math, 59 (2006), pp. 1207–1223.

[41] E. J. Candes and M. B. Wakin, *An introduction to compressive sampling*, IEEE Signal Process Mag, 25 (2008), pp. 21–30.

[42] V. J. Challis, *A discrete level-set topology optimization code written in matlab*, Struct Multidisc Optim, 41 (2009), pp. 453–464.

[43] A. Chambolle, *An algorithm for total variation minimization and applications*, J Math Imaging Vis, 20 (2004), pp. 89–97.

[44] A. Chambolle and C. Larsen, *C-infinity regularity of the free boundary for a two-dimensional optimal compliance problem*, Calc Var Partial Dif, 18 (2003), pp. 77–94.

[45] D. Chapelle and K. J. Bathe, *The inf-sup test*, Comput Struct, 47 (1993), pp. 537–545.

[46] G. H. G. Chen and R. T. Rockafellar, *Convergence rates in forward-backward splitting*, SIAM J Optimiz, 7 (1997), pp. 421–444.

[47] D. Chenais, *On the existence of a solution in a domain identification problem*, J Math Anal Appl, 52 (1975), pp. 189–219.

[48] A. Cherkaev, *Variational Methods for Structural Optimization*, Springer Verlag, New York, 2000.

[49] P. G. Ciarlet, *Mathematical Elasticity: Volume 1: Three-Dimensional Elasticity*, Elsevier, 1988.

[50] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley, New York, 1983.

[51] A. Cohen and R. Masson, *Wavelet methods for second-order elliptic problems, preconditioning, and adaptivity*, SIAM J Sci Comput, 21 (2000), pp. 1006–1026.

[52] G. Cohen, *Optimization by decomposition and coordination: a unified approach*, IEEE Trans Autom Control, 23 (1978), pp. 222–232.

[53] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Model Sim, 4 (2006), pp. 1168–1200.

[54] R. D. Cook, D. S. Malkus, and M. E. Plesha, *Concepts and Applications of Finite Element Analysis*, Wiley, New York, 4th ed., 2002.

[55] A. L. Cunha, *A fully Eulerian method for shape optimization, with application to Navier-Stokes flows*, Master's thesis, Carnegie Mellon University, 2004.

[56] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of 24th National Conference ACM, 1969, pp. 157–172.

[57] B. Dacorogna and P. Marcellini, *Existence of minimizers for non-quasiconvex integrals*, Arch Ration Mech An, 131 (1995), pp. 359–399.

[58] M. Dambrine and D. Kateb, *On the Ersatz material approximation in level-set methods*, ESAIM Contr Optim Ca, 16 (2009), pp. 618–634.

[59] M. J. de Ruiter and F. V. Keulen, *Topology optimization using a topology description function*, Struct Multidisc Optim, 26 (2004), pp. 406–416.

[60] L. Dede, M. J. Borden, and T. J. R. Hughes, *Isogeometric analysis for topology optimization with a phase field model*, ICES-Report 11-29, The Institute for Computational Engineering and Sciences, 2011.

[61] M. C. Delfour and J. P. Zolésio, *Shapes and geometries: analysis, differential calculus, and optimization*, Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[62] A. Diaz and O. Sigmund, *Checkerboard patterns in layout optimization*, Struct Optimization, 10 (1995), pp. 40–45.

[63] J. Donea and A. Huerta, *Finite Element Methods for Flow Problems*, John Wiley and Sons, Ltd., West Sussex, England, 2003.

[64] Q. Du, M. Emelianenko, and L. Ju, *Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations*, SIAM J Numer Anal, 44 (2006), pp. 102–119.

[65] Q. Du, V. Faber, and M. D. Gunzburger, *Centroidal Voronoi tessellations: Applications and algorithms*, SIAM Rev, 41 (1999), pp. 637–676.

[66] Q. Du and M. D. Gunzburger, *Grid generation and optimization based on centroidal Voronoi tessellations*, Appl Math Comput, 133 (2002), pp. 591–607.

[67] Q. Du, M. D. Gunzburger, and L. Ju, *Constrained centroidal Voronoi tessellations for surfaces*, SIAM J Sci Comput, 24 (2003), pp. 1488–1506.

[68] Q. Du and D. Wang, *The optimal centroidal Voronoi tessellations and the Gersho's conjecture in the three-dimensional space*, Comput Math Appl, 49 (2005), pp. 1355–1373.

[69] J. Duchi and Y. Singer, *Efficient online and batch learning using forward backward splitting*, J Mach Learn Res, 10 (2009), pp. 2899–2934.

[70] R. G. Duran and M. A. Muschietti, *The Korn inequality for Jones domains*, Electron J Differential Equations, 127 (2004), pp. 1–10.

[71] H. W. Engl, M. Hangke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer Academic Publishers, 1996.

[72] J. Erhel and F. Guyomarc'h, *An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems*, SIAM J Matrix Anal Appl, 21 (2000), pp. 1279–1299.

[73] A. Ern and J. L. Guermond, *Theory and Practice of Finite Elements*, Springer Verlag, New York, 2004.

[74] L. C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics, American Mathematical Society, Rhode Island, 1998.

[75] L. C. Evans and R. F. Gariepy, *Measure Theory and Fine Properties of Functions*, CRC Press, 1991.

[76] A. Evgrafov, *The limits of porous materials in the topology optimization of Stokes flows*, Appl Math Optim, 52 (2005), pp. 263–277.

[77] C. Fleury and V. Braibant, *Structural optimization: a new dual method using mixed variables*, Int J Numer Meth Eng, 23 (1986), pp. 409–428.

[78] M. S. Floater and J. Kosinka, *On the injectivity of Wachspress and mean value mappings between convex polygons*, Adv Comput Math, 32 (2010), pp. 163–174.

[79] E. M. Gafni and D. Bertsekas, *Two-metric projection methods for constrained optimization*, SIAM J Control Opt, 20 (1984), pp. 936–964.

[80] A. Gillette, A. Rand, and C. Bajaj, *Error estimates for generalized barycentric interpolation*, Adv Comput Math, DOI 10.1007/s10444-011-9218-z (2012).

[81] V. Girault and P. A. Raviart, *Fintie Element Method for Navier-Stokes equations*, Springer Verlag, 1986.

[82] A. A. Groenwold and L. F. P. Etman, *On the equivalence of optimality criterion and sequential approximate optimization methods in the classical topology layout problem*, Int J Numer Meth Eng, 73 (2008), pp. 297–316.

[83] ——, *A quadratic approximation for structural topology optimization*, Int J Numer Meth Eng, 82 (2010), pp. 505–524.

[84] A. A. Groenwold, L. F. P. Etman, and D. W. Wood, *Approximated approximations for SAO*, Struct Multidisc Optim, 41 (2010), pp. 39–56.

[85] J. K. Guest, J. H. Prevost, and T. Belytschko, *Achieving minimum length scale in topology optimization using nodal design variables and projection functions*, Int J Numer Meth Eng, 61 (2004), pp. 238–254.

[86] X. Guo and Y. X. Gu, *A new density-stiffness interpolation scheme for topology optimization of continuum structures*, Eng Computation, 21 (2004), pp. 9–22.

[87] X. Guo, K. Zhao, and M. Y. Wang, *A new approach for simultaneous shape and topology optimization based on dynamic implicit surface function*, Control Cybern, 34 (2005), pp. 255–282.

[88] R. B. Haber, C. S. Jog, and M. P. Bendsøe, *A new approach to variable-topology shape design using a constraint on perimeter*, Struct Optimization, 11 (1996), pp. 1–12.

[89] Y. Huang, H. Qin, and D. Wang, *Centroidal Voronoi tessellation-based finite element superconvergence*, Int J Numer Meth Eng, 76 (2008), pp. 1819–1839.

[90] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, 2000.

[91] C. S. Jog and R. B. Haber, *Stability of finite element models for distributed-parameter optimization and topology design*, Comput Methods Appl Mech Engrg, 130 (1996), pp. 203–226.

[92] C. S. Jog, R. B. Haber, and M. P. Bendsøe, *Topology design with optimized, self-adaptive materials*, Int J Numer Meth Eng, 37 (1994), pp. 1323–1350.

[93] L. Ju, M. D. Gunzburger, and W. Zhao, *Adaptive finite element methods for elliptic PDEs based on conforming centroidal Voronoi-Delaunay triangulations*, SIAM J Sci Comput, 28 (2006), pp. 2023–2053.

[94] A. Kawamoto, T. Matsumori, S. Yamasaki, T. Nomura, T. Kondoh, and S. Nishiwaki, *Heaviside projection based topology optimization by a PDE-filtered scalar function*, Struct Multidisc Optim, 44 (2010), pp. 19–24.

[95] J. H. Kim and G. H. Paulino, *Isoparametric graded finite elements for nonhomogeneous isotropic and orthotropic materials*, J Appl Mech-T ASME, 69 (2002), pp. 502–514.

[96] R. V. Kohn and G. Strang, *Optimal design and relaxation of variational problems I*, Comm Pure Appl Math, 39 (1986), pp. 113–137.

[97] ——, *Optimal design and relaxation of variational problems II*, Comm Pure Appl Math, 38 (1986), pp. 139–182.

[98] ——, *Optimal design and relaxation of variational problems III*, Comm Pure Appl Math, 39 (1986), pp. 353–377.

[99] I. Kosaka and C. C. Swan, *A symmetry reduction method for continuum structural topology optimization*, Comput Struct, 70 (1999), pp. 47–61.

[100] S. Kreissl, G. Pingen, and K. Maute, *An explicit level set approach for generalized shape optimization of fluids with the Lattice Boltzmann method*, Int J Numer Meth Fl, 65 (2011), pp. 496–519.

[101] M. Langelaar, *The use of convex uniform honeycomb tessellations in structural topology optimization*, Proceedings of 7th World Congress on Structural and Multidisciplinary Optimization, (2007).

[102] B. S. Lazarov and O. Sigmund, *Filters in topology optimization based on Helmholtz-type differential equations*, Int J Numer Meth Eng, 86 (2011), pp. 765–781.

[103] T. Lewinski, M. Zhou, and G. I. N. Rozvany, *Extended exact least-weight truss layouts–Part II: Unsymmetric cantilevers*, Int J Mech Sci, 36 (1994), pp. 399–419.

[104] ——, *Extended exact solutions for least-weight truss layouts–Part I: Cantilever with a horizontal axis of symmetry*, Int J Mech Sci, 36 (1994), pp. 375–398.

[105] O. Ley, *Lower-bound gradient estimates for first-order Hamilton-Jacobi equations and applications to the regularity of propagating fronts*, Adv Differential Equations, 6 (2001), pp. 547–576.

[106] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM J Numer Anal, 16 (1979), pp. 964–979.

[107] R. Lipton, *A saddle-point theorem with application to structural optimization*, J Optimiz Theory App, 81 (1994), pp. 549–568.

[108] ——, *Stress constrained G-closure and relaxation of structural design problems*, Quart Appl Math, 62 (2004), pp. 295–322.

[109] W. B. Liu, P. Neittaanmaki, and D. Tiba, *Existence for shape optimization problems in arbitrary dimension*, SIAM J Control Opt, 41 (2002), p. 1440.

[110] Y. Liu, W. Wang, B. Levy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, *On centroidal Voronoi tesselleations–Energy smoothness and fast computation*, ACM T Graphic, 28 (2009), pp. 1–17.

[111] Z. Liu, J. Korvink, and R. Huang, *Structure topology optimization: fully coupled level set method via FEMLAB*, Struct Multidisc Optim, 29 (2005), pp. 407–417.

[112] J. M. Martinez, *A note on the theoretical convergence properties of the SIMP method*, Struct Multidisc Optim, 29 (2005), pp. 319–323.

[113] M. Meyer, A. Barr, H. Lee, and M. Desbrun, *Generalized barycentric coordinates on irregular polygons*, J Graphics Tools, 7 (2002), pp. 13–22.

[114] M. L. Minion, *Higher-order semi-implicit projection methods*, tech. rep., Lawernce Livermore National Laboratory, 2001.

[115] S. E. Mousavi, H. Xiao, and N. Sukumar, *Generalized Gaussian quadrature rules on arbitrary polygons*, Int J Numer Meth Eng, 82 (2010), pp. 99–113.

[116] S. Natarajan, S. Bordas, and D. R. Mahapatra, *Numerical integration over arbitrary polygonal domains based on Schwarz-Christoffel conformal mapping*, Int J Numer Meth Eng, 80 (2009), pp. 103–134.

[117] P. Neittaanmaki and D. Tiba, *Existence and approximation in optimal shape design problems*, University of Jyvaskyla, Report, (1998).

[118] Y. Nesterov, *Gradient methods for minimizing composite objective function.* available at http://www.ecore.be/DPs/dp1191313936.pdf., 2007.

[119] N. Olhoff, M. P. Bendsøe, and J. Rasmussen, *On CAD-integrated structural topology and design optimization*, Comput Methods Appl Mech Engrg, 89 (1991), pp. 259–279.

[120] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153 of Applied Mathematical Sciences, Springer-Verlag, New York, 2003.

[121] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov supaces for sequences of linear systems*, SIAM J Sci Comput, 28 (2006), pp. 1651–1674.

[122] M. Patriksson, *Cost approximation: A unified framework of descent algorithms for nonlinear programs*, SIAM J Optimiz, 8 (1998), pp. 561–582.

[123] P.-O. Persson. http://persson.berkeley.edu/thesis/structoptim/.

[124] ——, *Mesh size functions for implicit geometries and PDE-based gradient limiting*, Eng Comput, 22 (2006), pp. 95–109.

[125] P.-O. Persson and G. Strang, *A simple mesh generator in MATLAB*, SIAM Rev, 46 (2004), pp. 329–345.

[126] J. Petersson, *A finite element analysis of optimal variable thickness sheets*, SIAM J Numer Anal, 36 (1999), pp. 1759–1778.

[127] ——, *Some convergence results in perimeter-controlled topology optimization*, Comput Methods Appl Mech Engrg, 171 (1999), pp. 123–140.

[128] J. Petersson and O. Sigmund, *Slope constrained topology optimization*, Int. J. Numer. Meth. Engng, 41 (1998), pp. 1417–1434.

[129] G. Pingen, M. Waidmann, A. Evgrafov, and K. Maute, *A parametric level-set approach for topology optimization of flow domains*, Struct Multidisc Optim, 41 (2010), pp. 117–131.

[130] O. Pironneau, *Optimal Shape Design of Elliptic Systems*, Springer-Verlag, 1983.

[131] T. A. Poulsen, *A new scheme for imposing a minimum length scale in topology optimization*, Int J Numer Meth Eng, 57 (2003), pp. 741–760.

[132] A. Ricci, *A constructive geometry for computer graphics*, Comp J, 16 (1973), pp. 157–160.

[133] A. Rietz, *Sufficiency of a finite exponent in SIMP (power law) methods*, Struct Multidisc Optim, 21 (2001), pp. 159–163.

[134] H. Royden, *Real Analysis*, Prentice Hall, 3rd ed., 1988.

[135] G. I. N. Rozvany, *A critical review of established methods of structural topology optimization*, Struct Multidisc Optim, 37 (2009), pp. 217–237.

[136] G. I. N. Rozvany, M. Zhou, and T. Birker, *Generalized shape optimization without homogenization*, Struct Optimization, 4 (1992), pp. 250–252.

[137] A. Saxena, *A material-mask overlay strategy for continuum topology optimization of compliant mechanisms using honeycomb discretization*, J Mech Design, 130 (2008), p. 082304.

[138] J. A. Sethian, *Fast marching methods*, SIAM Rev, 41 (1999), pp. 199–235.

[139] D. Sieger, P. Alliez, and M. Botsch, *Optimizing Voronoi diagrams for polygonal finite element computations*, Proceedings of the 19th International Meshing Roundtable, (2010), pp. 435–350.

[140] O. Sigmund, *A 99 line topology optimization code written in matlab*, Struct Multidisc Optim, 21 (2001), pp. 120–127.

[141] ——, *Morphology-based black and white filters for topology optimization*, Struct Multidisc Optim, 33 (2007), pp. 401–424.

[142] O. Sigmund and J. Petersson, *Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima*, Struct Optimization, 16 (1998), pp. 68–75.

[143] T. Sokół, *A 99 line code for discretized Michell truss optimization written in Mathematica*, Struct Multidisc Optim, 43 (2011), pp. 181–190.

[144] M. Stolpe and K. Svanberg, *An alternative interpolation scheme for minimum compliance topology optimization*, Struct Multidisc Optim, 22 (2001), pp. 116–124.

[145] ——, *On the trajectories of penalization methods for topology optimization*, Struct Multidisc Optim, 21 (2001), pp. 128–139.

[146] L. L. Stromberg, A. Beghini, W. F. Baker, and G. H. Paulino, *Application of layout and topology optimization using pattern gradation for the conceptual design of buildings*, Struct Multidisc Optim, 43 (2011), pp. 165–180.

[147] N. Sukumar, *Construction of polygonal interpolants: a maximum entropy approach*, Int J Numer Meth Eng, 61 (2004), pp. 2159–2181.

[148] N. Sukumar and E. A. Malsch, *Recent advances in the construction of polygonal finite element interpolants*, Arch Comput Method E, 13 (2006), pp. 129–163.

[149] N. Sukumar and A. Tabarraei, *Conforming polygonal finite elements*, Int J Numer Meth Eng, 61 (2004), pp. 2045–2066.

[150] K. Suresh, *A 199-line matlab code for Pareto-optimal tracing in topology optimization*, Struct Multidisc Optim, 42 (2010), pp. 665–679.

[151] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, *An improved level set method for incompressible two-phase flows*, Comput Fluids, 27 (1998), pp. 663–680.

[152] K. Suzuki and N. Kikuchi, *A homogenization method for shape and topology optimization*, Comput Methods Appl Mech Engrg, 93 (1991), pp. 291–318.

[153] K. Svanberg, *The Method of Moving Asymptotes–A new method for structural optimization*, Int J Numer Meth Eng, 24 (1987), pp. 359–373.

[154] ——, *A class of globally convergent optimization methods based on conservative convex separable approximations*, SIAM J Optimiz, 12 (2001), pp. 555–573.

[155] V. Sverak, *On optimal shape design*, J Math Pures Appl, 72 (1993), pp. 537–551.

[156] A. Tabarraei and N. Sukumar, *Application of polygonal finite elements in linear elasticity*, Int J Comput Methods, 3 (2006), pp. 503–520.

[157] A. Takezawa, S. Nishiwaki, and M. Kitamura, *Shape and topology optimization based on the phase field method and sensitivity analysis*, J Comput Phys, 229 (2010), pp. 2697–2718.

[158] C. Talischi, *Existence of solutions for a restriction-type topology optimization formulation*, Report, (2009).

[159] C. Talischi and G. H. Paulino, *A consistent operator splitting algorithm and a two-metric variant: Application to topology optimization*, Under review, (2012).

[160] ——, *An operator splitting algorithm for Tikhonov-regularized topology optimization*, Comput Methods Appl Mech Engrg, in press (2012).

[161] C. Talischi, G. H. Paulino, and C. H. Le, *Honeycomb Wachspress finite elements for structural topology optimization*, Struct Multidisc Optim, 37 (2009), pp. 569–583.

[162] C. Talischi, G. H. Paulino, A. Pereira, and I. F. M. Menezes, *Polygonal finite elements for topology optimization: A unifying paradigm*, Int J Numer Meth Eng, 82 (2010), pp. 671–698.

[163] ——, *PolyMesher: A general-purpose mesh generator for polygonal elements written in Matlab*, Struct Multidisc Optim, 45 (2012), pp. 309–328.

[164] ——, *PolyTop: A Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes*, Struct Multidisc Optim, 45 (2012), pp. 329—357.

[165] L. Tartar, *An introduction to the homogenization method in optimal design*, Optimal Shape Design: Lecture Notes in Mathematics, Springer, 2000, pp. 47–156.

[166] A. R. Terrel and K. R. Long, *Evaluation of level set topology optimization formulations for design of minimum-dispersion microfluidic devices*, in NECIS Summer Proceedings, 2006.

[167] K. van den Doel and U. M. Ascher, *On level set regularization for highly ill-posed distributed parameter estimation problems*, J Comput Phys, 216 (2006), pp. 707–723.

[168] N. P. van Dijk, M. Langelaar, and F. van Keulen, *A discrete formulation of a discrete level-set method treating multiple constraints*, Proceedings of 8th World Congress on Structural and Multidisciplinary Optimization, (2009).

[169] ——, *Explicit level-set-based topology optimization using an exact Heaviside function and consistent sensitivity analysis*, Int J Numer Meth Eng, 91 (2012), pp. 67–97.

[170] E. L. Wachspress, *A Rational Finite Element Basis*, Academic Press, 1975.

[171] F. Wang, B. Lazarov, and O. Sigmund, *On projection methods, convergence and robust formulations in topology optimization*, Struct Multidisc Optim, 43 (2011), pp. 767–784.

[172] M. Y. Wang and S. Y. Wang, *Bilateral filtering for structural topology optimization*, Int J Numer Meth Eng, 63 (2005), pp. 1911–1938.

[173] M. Y. Wang, X. M. Wang, and D. M. Guo, *A level set method for structural topology optimization*, Comput Methods Appl Mech Engrg, 192 (2003), pp. 227–246.

[174] M. Y. Wang and S. Zhou, *Synthesis of shape and topology of multi-material structures with a phase-field method*, J Comput-Aided Mater, 11 (2004), pp. 117–138.

[175] S. Y. Wang, K. M. Lim, B. C. Khoo, and M. Y. Wang, *An extended level set method for shape and topology optimization*, J Comput Phys, 221 (2007), pp. 395–421.

[176] S. J. Wright, *Optimization in machine learning*, in Neural Information Processing Systems (NIPS) Workshop, 2008.

[177] T. Yamada, K. Izui, S. Nishiwaki, and A. Takezawa, *A topology optimization method based on the level set method incorporating a fictitious interface energy*, Comput Methods Appl Mech Engrg, 199 (2010), pp. 2876–2891.

[178] S. Yamasaki, S. Nishiwaki, T. Yamada, K. Izui, and M. Yoshimura, *A structural optimization method based on the level set method using a new geometry-based re-initialization scheme*, Int J Numer Meth Eng, 83 (2010), pp. 1580–1624.

[179] M. Yip, J. Mohle, and J. E. Bolander, *Automated modeling of three-dimensional structural components using irregular lattices*, Comput-Aided Civ Inf, 20 (2005), pp. 393–407.

[180] L. Yuqui and X. Yin, *Generalized conforming triangular membrane element with vertex rigid rotational freedoms*, Finite Elem Anal Des, 17 (1994), pp. 259–271.

[181] W. H. Zhang and P. Duysinx, *Dual approach using a variant perimeter constraint and efficient sub-iteration scheme for topology optimization*, Comput Struct, 81 (2003), pp. 2173–2181.

[182] H. Zhao, *A fast sweeping method for Eikonal equations*, Math Comput, 74 (2004), pp. 603–627.

[183] M. Zhou and G. I. N. Rozvany, *The COC algorithm, part II: Topological, geometrical and generalized shape optimization*, Comput Methods Appl Mech Engrg, 89 (1991), pp. 309–336.

[184] S. Zhou and M. Y. Wang, *Multimaterial structural topology optimization with a generalized Cahn-Hilliard model of multiphase transition*, Struct Multidisc Optim, 33 (2007), pp. 89–111.