Efficient Structural Update for Three-Dimensional Topology Optimization Problems Using Level Set Functions

Emanuel Teichmann

Vom Fachbereich Mathematik
der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte Dissertation.

1. Gutachter: PD. Dr. Oleg Iliev
2. Gutachter: Prof. Dr. Joachim Schöberl

Vollzug der Promotion: 07. November 2008

D 386

Abstract

We present a new efficient and robust algorithm for topology optimization of 3D cast parts. Special constraints are fulfilled to make possible the incorporation of a simulation of the casting process into the optimization: In order to keep track of the exact position of the boundary and to provide a full finite element model of the structure in each iteration, we use a twofold approach for the structural update. A level set function technique for boundary representation is combined with a new tetrahedral mesh generator for geometries specified by implicit boundary descriptions. Boundary conditions are mapped automatically onto the updated mesh. For sensitivity analysis, we employ the concept of the topological gradient.

Modification of the level set function is reduced to efficient summation of several level set functions, and the finite element mesh is adapted to the modified structure in each iteration of the optimization process. We show that the resulting meshes are of high quality. A domain decomposition technique is used to keep the computational costs of remeshing low. The capabilities of our algorithm are demonstrated by industrial-scale optimization examples.

Acknowledgements

In the first place I would like to thank my advisors Heiko Andrä and Oleg Iliev. They gave me plenty of opportunity for independent work as well as unfailing guidance and valuable stimulation through many fruitful discussions. Their steady support enabled me to finish this thesis.

Sincere thanks are given to Iuliana Matei and Inga Shklyar. Their contributions to the solution of numerous questions of detail and suggestions for further development were particularly helpful.

My special gratitude goes to my parents and my wife Inger Lise, for their unconditional encouragement and believe in me. The backup they gave me throughout these years contributed vitally to the completion of this work.

This research was done at the Fraunhofer Institut für Techno- und Wirtschaftsmathematik in the framework of the project "MIDPAG - Innovative Methods for Integrated Dimensioning and Process Design for Cast Parts", which has been supported in part by the European Regional Development Fund, ERDF, and the Ministry of Economic Affairs, Transport, Agriculture, and Viniculture of Rhineland-Palatinate.

Contents

List of notations 1

Introduction 3
  0.1. Prior Work 5

Chapter 1. The Topology Optimization Problem 9
  1.1. The Continuous Topology Optimization Problem 9
  1.2. The Discrete Topology Optimization Problem 13

Chapter 2. Boundary Representation Using a Level Set Function 15
  2.1. Boundary Representation by a Continuous Level Set Function 16
  2.2. Boundary Representation by a Discrete Level Set Function 18
  2.3. Regularization 20

Chapter 3. Mesh Generation 23
  3.1. Uniform Mesh Generation 24
  3.1.1. Uniform 2D Triangulation 25
  3.1.2. Uniform 3D Tetrahedralization 28
  3.2. Adaptive Mesh Generation 32
  3.2.1. Adaptive 2D Triangulation 32
  3.2.2. Adaptive 3D Tetrahedralization 34
  3.3. Mesh Postprocessing and Improvement 37
  3.4. Boundary Condition Mapping 43
  3.5. Mesh Quality 45
  3.5.1. Mesh Quality Measures 47
  3.5.2. Mesh Quality Evaluation 48

Chapter 4. Numerical Algorithm 54
  4.1. The Underlying Optimization Algorithm 55
  4.2. Topological Sensitivity 57
  4.3. Structural Update 61
  4.3.1. Update of the Level Set Function 61
  4.3.2. Modification of the FE Mesh 65
  4.4. The Complete Numerical Algorithm 67

Chapter 5. Numerical Examples 70
5.1. Bridge 70
5.2. Dial Plate 75
5.3. Bearing Cap 80

Summary and Future Work 85

Bibliography 87

List of notations

$\emptyset$	the empty set
C	cell of a Cartesian grid
$\mathbb{C}$	Hooke's 4th order material tensor
D	design space
∂D	boundary of the design space
E	edge of a cell of a Cartesian grid
ε	linearized strain tensor
F	face of a cell of a Cartesian grid
$\mathcal{G}_\theta$	regular Cartesian grid of width θ
Γ_0	$\partial\Omega \cap D$
Γ_D	part of ∂D where Dirichlet boundary conditions are applied
Γ_N	part of ∂D where Neumann boundary conditions are applied
Γ_{N_0}	part of Γ_N where homogeneous Neumann boundary conditions are applied
Γ_{N_t}	part of Γ_N where inhomogeneous Neumann boundary conditions are applied
ϑ	level set function discretization parameter
h	finite element discretization parameter
h_k	diameter of the k-th finite element
$H^{-1/2}$	Sobolev space with norm $\|\|f\|\|^2_{H^{-1/2}}$
$\mathcal{I}$	2D or 3D image for adaptive mesh generation
j	objective function incorporating cost functional and constraints
J	cost functional
K	stiffness matrix
$\mathcal{K}$	simplicial complex
λ, μ	Lamé coefficients
$\mathbf{n}$	normal vector on $\partial\Omega$
$\mathbb{N}$	natural numbers
$\mathbb{N}^3$	three-dimensional space with integer coordinates
$\mathcal{N}_1(P)$	set of one-ring neighbors of P
P	corner point of a cell in a Cartesian grid
$\mathcal{P}$	orthogonal projection operator
φ	continuous level set function for domain representation
φ_ϑ	discretized level set function for domain representation

ψ	given load on Γ_N, $\psi \in \left(H^{-1/2}\left(\Gamma_N\right)\right)^3$,
$\mathbb{R}$	real numbers
$\mathbb{R}^3$	three-dimensional Euclidian space with real coordinates
σ	Cauchy stress tensor
$\mathcal{T}$	triangulation of a domain boundary
$\mathcal{U}_{ad}$	set of admissible domains
V_{target}	target volume
Ω	domain
$\overline{\Omega}$	closure of Ω
$\partial\Omega$	boundary of Ω
Ω_{bor}	border layer of Ω
Ω_h	domain represented by a finite element mesh
Ω_{int}	regular interior part of Ω
$\Omega_{\varphi_\vartheta}$	domain represented by a discrete level set function

Introduction

Modern design in mechanical engineering has not only the construction of functional and reliable parts as a goal, but includes more and more aspects as weight reduction and optimal utilization of a minimum amount of material. This reduces production costs as well as operation expenses, since the amount of energy needed to set parts into motion decreases with the weight of the part. This is important for instance in vehicle and aircraft construction.

Since human intuition quickly reaches its limits as the complexity of components increases, mathematical tools have been developed to assist construction engineers with finding better designs. This field is called *structural optimization* or *topology optimization*.

Topology optimization is particularly interesting for cast parts, since this manufacturing method to a unique extent allows for formation of fine and complex three-dimensional structures. Due to this high flexibility, cast components often feature a large optimization potential. One drawback, though, is the risk of eigenstresses building up during non-uniform cooling down. These stresses can reduce the strength of cast parts when superimposed with external working loads. However, inclusion of a simulation of the casting and solidification process in structural optimization makes possible to account for the effects of eigenstresses and thereby produce more efficient components.

Integration of a casting simulation in the optimization process demands two things: The exact position of the structure's boundary and thus the domain occupied by the cast alloy has to be known at any time. In addition, a complete finite element mesh approximating the structure has to be provided in each step of the optimization process.

Existing topology optimization algorithms do not satisfy both criteria (cf. section 0.1), and to the best of our knowledge the algorithm presented in this thesis is the first fulfilling the requirements and thus allowing for inclusion of a casting simulation in the optimization process.

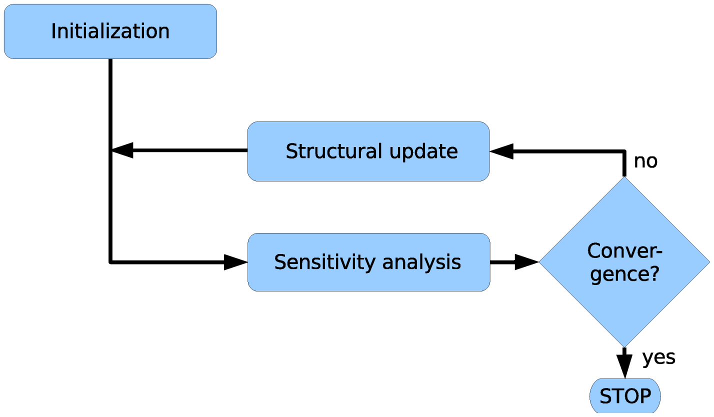


FIGURE 0.1. General structure of topology optimization algorithms. After the initial structure is provided (initialization), sensitivity analysis and structural update are iterated until the algorithm converges.

As all popular topology optimization algorithms, our method comprises three main steps: Initialization, sensitivity analysis, and structural update. A graphical illustration of the process is given in Figure 0.1. *Initialization* covers the definition of the starting structure, boundary conditions, and material under consideration. Loosely speaking, *sensitivity analysis* can be understood as computation of a derivative of the structure's form with respect to the objective function. It results in a criterion prescribing how the structure can be modified in order to decrease the objective function. In this work, we employ the concept of the topological gradient [**26**]. Finally, *structural update* denotes the realization of the modifications indicated by the sensitivity analysis, i.e., the actual deformation of the structure.

In order to implement the structural update under the constraints described above we employ and combine two different approaches: On the one hand, we make use of the flexibility provided by implicit boundary descriptions using level set functions [**38**]. Compared to other topology optimization methods which employ level set functions, we simplify the structural update significantly. Instead of solving a Hamilton-Jacobi equation and evolving the domain by a velocity function, we represent structural modifications by simple level set functions and define an addition operator. Thus, the level set function merely serves as a means of boundary representation.

On the other hand, we developed an octree-based tetrahedral mesh generator which takes the level set function as input and therewith operates directly on the implicit boundary description. A domain decomposition approach makes possible to avoid complete remeshing in each iteration of the optimization algorithm: A large part of the mesh close to the structure's boundary is kept, and only areas in which the level set function has been modified are remeshed. All boundary conditions are automatically transferred to the updated mesh, i.e., in each iteration we construct a complete finite element model which is not created by simple removal of elements from an existing mesh but has been adapted to the modified structure. This is crucial not only for the casting simulation, but allows in addition for exact sensitivity analysis and evaluation of the component's performance. We are not aware of any other structural optimization algorithm providing this feature, which is of great importance for execution of the algorithm.

Both our level set function update and the mesh modification algorithm are very efficient, which keeps computing times low and allows for optimization of industrial-scale cast parts.

This work is organized as follows: In section 0.1, we give a short overview of prior work done in the fields of structural optimization, mesh generation, and implicit boundary representation, without which we could not have accomplished this thesis. Thereafter, we present the topology optimization problem, which is a moving boundary value problem (chapter 1). Chapter 2 contains a review of implicit boundary representation by means of level set functions. Our mesh generation algorithm along with an assessment of the mesh quality is given in chapter 3. Chapters 2 and 3 form the basis for the description of the structural update. Chapter 4 comprises the optimization algorithm we use, the topological sensitivity, and the structural update for both level set function and finite element mesh, i.e., it provides a detailed presentation of the complete numerical topology optimization algorithm which is sketched in Figure 0.1. The capabilities of our algorithm are demonstrated by numerical examples in chapter 5. A summary and some ideas for future work complete this thesis.

0.1. Prior Work

In the field of structural optimization, three essential approaches have been developed. *Sizing*, which finds an optimal choice of a finite number of parameters describing, e.g., the cross section area or diameter of the so-called members of a structure, *shape optimization*, which alters the structure's boundary, and *topology optimization*, which in contrast to shape optimization allows for creation, merging, and deletion of holes. A schematic illustration of the three methods is shown in Figure 4.2.

Clearly, sizing is the least flexible of the mentioned structural optimization methods. However, it is widely applied in areas like steel construction where structures are built of normed parts which are not available in arbitrary sizes or shapes.

Shape optimization is a powerful tool for improvement of structures the topology and approximate form of which are known or prescribed. This restriction is the reason why the method often shows a strong dependency on the initial guess. The sensitivity used in shape optimization is called *shape derivative*. For a comprehensive introduction to shape derivatives and shape optimization see, e.g., [**56, 17**].

In this work, we focus on topology optimization. The first algorithms of this class used sensitivities based on local stress criteria as for instance the von Mises stress [**10**]. In 1995, Schumacher [**49**] introduced the concept of the *topological sensitivity*, which was further developed by Masmoudi et al. [**26, 33**], who introduced the name *topological gradient*, and thereafter widely adopted in structural optimization ([**55, 14, 36, 7**] and many more).

The strongest point of this method is its flexibility. It admits basically arbitrary starting structures, indicates all necessary topology changes, and incorporates shape optimization.

For the actual structural update, i.e., the implementation of the topology changes indicated by the topological sensitivity, various approaches are used.

Homogenization methods, which were first introduced by Bendsoe and Kikuchi [**10**] and later widely applied [**58, 18, 6**], actualize topology changes by using a material model with varying densities. Material is not directly removed, but the density is changed according to the optimality criteria. Several variations of the homogenization method have been developed to deal with the problem that the resulting structures cannot be manufactured. Grayscale designs with varying density have to be replaced by black-and-white designs which do not allow for intermediate densities.

Probably the most well-known approach is the *solid isotropic material with penalization* method (SIMP) [**9, 11**]. It assumes constant material properties in each discretization element (e.g., cube or tetrahedron) of the structure and uses the density in the cells as design variables. Material properties are assumed to be proportional to the relative density change raised to some power. The power law approach penalizes intermediate densities and leads eventually to the desired black-and-white designs.

Since all modifications are performed on one and the same mesh and only entire cells can be deleted from or added to the structure, the resulting designs show as a general rule very jagged edges and are not as smooth as the underlying density field might prescribe.

It is immediately clear that this approach is not suitable in our case. Intermediate densities cannot be modeled in the simulation of the casting process, and a sharp boundary of the structure is required at any time.

A more flexible approach consists in the construction of an isosurface of equal density, which then is considered to be the structure's boundary. This is used by the topology optimization software package TOSCA [**66**]. The resulting structures are generally smoother than the ones obtained with the SIMP method. However, it is not clear which isosurface has to be chosen and the results of the optimization process can differ considerably even for small changes in the value determining the isosurface.

Besides the various homogenization methods, the use of level set functions in topology optimization has become very popular. This type of boundary representation has been instigated by Osher and Sethian [**38**] for modeling of physical phenomena involving propagating interfaces as for example fluid dynamics or combustion processes.

The governing equation for interface propagation is the time dependent initial value problem

$$\begin{aligned} \phi_t + F|\nabla\phi| &= 0, \\ \phi(x, t = 0) &\quad \text{given}, \end{aligned}$$

where ϕ denotes the level set function, ϕ_t is the time derivative of the level set function, F is a so-called speed function, and $\nabla\phi$ denotes the spatial derivative of the level set function.

In the context of topology optimization, F usually depends on the stresses σ in the structure and on the structure's boundary, respectively (see, e.g., [**14, 52, 4, 2**]).

The benefits of the application of level set functions are flexibility, robustness and efficiency. All kinds of topology modifications such as nucleation or merging of holes can be performed without problems. However, none of the previously developed algorithms adapts the mesh used for computation of the stresses to the modified structure. Thus, these approaches cannot be used if the casting simulation is to be included.

According to the concise overview of mesh generation techniques given in [**39**], there are three main classes of mesh generation algorithms: Quadtree (2D)/octree (3D) techniques, techniques based on the Delaunay criterion, and advancing front algorithms. Each of these groups comprises numerous particular algorithms. The Delaunay refinement method [**44**], which is one of the most popular meshing algorithms, starts from an initial triangulation and refines the mesh until a prescribed element quality is reached. Advancing front algorithms [**41**] start at the structure's boundary, too, and generate meshes by creating elements along a front moving from the boundary into the domain. Quadtree and octree methods [**63**] subdivide squares/cubes enclosing the structure and thereby refine the mesh until a desired resolution is achieved. In order to smooth the mesh and to achieve a better fit to the structure, irregular cells are created by computation of intersection points of cubes and the structure's boundary.

Most of the meshing algorithms require explicit descriptions of the structure's boundaries. Persson [**42**] presented a technique for generation of unstructured meshes from implicit geometries, which improves an initial mesh of a random point cloud by interpreting the mesh as a truss structure and solving for force equilibrium. Zhang et al. [**64**] published an algorithm for adaptive octree-based mesh generation from volumetric imaging data. They minimize a quadratic error function and are able to preserve sharp features of the input structures. Our mesh generation algorithm follows central ideas presented in these two works.

REMARK 0.1. For further information on the inclusion of eigenstresses in topology optimization and for examples demonstrating the resulting advantages, we would like to refer to [**34**]. This work is closely related to our work in that it deals with other aspects of the same problem: Topology optimization for cast parts.

CHAPTER 1

The Topology Optimization Problem

In this chapter, we describe the continuous and discretized topology optimization problem under consideration. In section 1.1, we give a general representation of the design space and the domain, followed by the formulation of the objective function and the constraints. More details on the continuous case can be found in [**34**]. Section 1.2 describes the discretization.

1.1. The Continuous Topology Optimization Problem

Let D be a Lipschitz domain in $\mathbb{R}^3$ and Ω be a smooth subdomain of D with boundary $\partial\Omega \in C^{0,1}$ occupied by a linear elastic material. Ω denotes the structure, i.e. the domain we want to optimize, whereas D stands for the design space, which is the maximal domain the structure may occupy.

The boundary ∂D of D is made of two disjoint parts

$$\partial D = \overline{\Gamma_D \cup \Gamma_N}, \qquad \Gamma_D \cap \Gamma_N = \emptyset,$$

where Γ_D is of nonzero Lebesgue measure. On Γ_D, Dirichlet boundary conditions are applied, whereas Γ_N is the area of Neumann boundary conditions. Γ_N be divided into the boundary portions of homogeneous and inhomogeneous Neumann boundary conditions, respectively:

$$\Gamma_N = \overline{\Gamma_{N_0} \cup \Gamma_{N_t}},$$

where $\Gamma_{N_0} \cap \Gamma_{N_t} = \emptyset$.

We assume that the boundary $\partial\Omega$ of Ω satisfies

$$\partial\Omega = \Gamma_D \cup \Gamma_{N_t} \cup (\Gamma_{N_0} \cap \partial\Omega) \cup \Gamma_0, \tag{1}$$

with $\Gamma_0 = \partial\Omega \cap D$. A graphical representation of the model problem's boundaries is given in Figure 1.1.

Let us denote the prescribed Dirichlet and Neumann boundary conditions on Γ_D and Γ_{N_t} by u^* and t^*, respectively. Then, a general elliptic boundary value problem (BVP) of mixed type in the domain Ω can be stated as

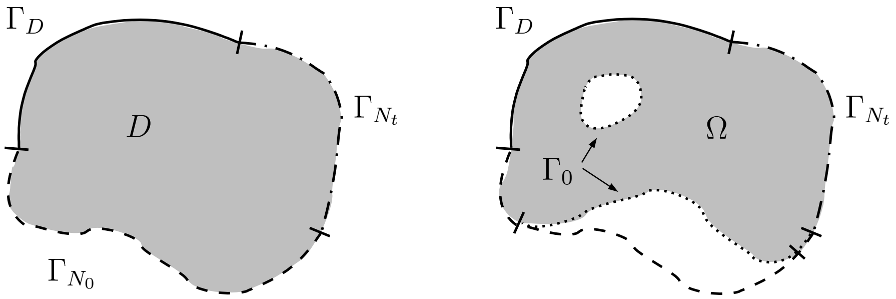


FIGURE 1.1. Design space D (left) and structure Ω (right) with the different boundary parts.

PROBLEM 1.1. **Mixed BVP in linear elasticity**

$$\left\{ \begin{array}{rcll} \mathcal{A}_\Omega(x)u(x) & = & F(x), & x \in \Omega, \\ \gamma_0(u)(x) & = & u^*(x), & x \in \Gamma_D, \\ \gamma_1(u)(x) & = & t^*(x), & x \in \Gamma_N \cap \partial\Omega. \end{array} \right. \tag{2}$$

Here, γ_0 and γ_1 are the Dirichlet and Neumann trace operator, respectively.

A particular case of the general boundary value problem (2) is the problem of linear elasticity. Here, the elliptic partial differential operator $\mathcal{A}_\Omega$ describes the balance of linear momentum and is given by

$$\mathcal{A}_\Omega u(x) = -\operatorname{div}(\sigma(u)(x)), \quad x \in \Omega, \tag{3}$$

where $u(x)$ is the displacement at point x and σ is the stress tensor. The relation between σ and the linearized strain tensor

$$\varepsilon = \frac{1}{2}\left(\nabla u + \nabla u^T\right)$$

is given by Hooke's law:

$$\sigma = \mathbb{C} : \varepsilon.$$

$\mathbb{C}$ denotes the 4-th order elasticity tensor, which in the case of an isotropic material described by the two Lamé coefficients λ and μ is given by

$$\mathbb{C}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}).$$

δ_{ij} is the Kronecker symbol:

$$\delta_{ij} = \left\{ \begin{array}{ll} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{array} \right. \qquad i, j = 1, 2, 3.$$

Using Hooke's law, equation (3), and denoting the outward normal on $\partial\Omega$ by $\mathbf{n}$, system (2) can be rewritten as follows:

$$\left\{\begin{array}{rcll} -\operatorname{div}(\mathbb{C}:\varepsilon(u(x))) &=& 0, & x\in\Omega,\\ u(x) &=& u^*(x), & x\in\Gamma_D,\\ (\mathbb{C}:\varepsilon(u(x)))\cdot\mathbf{n}(x) &=& t^*(x), & x\in\Gamma_{N_t},\\ (\mathbb{C}:\varepsilon(u(x)))\cdot\mathbf{n}(x) &=& 0, & x\in\Gamma_{N_0}. \end{array}\right. \tag{4}$$

In order to formulate the variational formulation of (4), we recall the definition of Sobolev spaces.

Definition 1.2. *Let $0\le r\in\mathbb{R}$, $r=s+t$, where $s\in\mathbb{N}_0$ and $t\in[0,1)$. Let n denote the space dimension and α be a multi-index of length $|\alpha|$. Then, the* **Sobolev space** *$H^r(\Gamma)$ is defined as the space of functionals $f:\Gamma\to\mathbb{R}$ with norm $||f||_{H^r(\Gamma)}<\infty$, where*

$$||f||_{H^r(\Gamma)}=\left(||f||^2_{H^s(\Gamma)}+|f|^2_{H^r(\Gamma)}\right)^{\frac{1}{2}}$$

and

$$|f|_{H^r(\Gamma)}=\left(\sum_{|\alpha|=s}\int_\Gamma\int_\Gamma\frac{|D^\alpha f(x)-D^\alpha f(y)|^2}{|x-y|^{n-1+2t}}ds_x ds_y\right)^{\frac{1}{2}}.$$

For $0>r\in\mathbb{R}$, the norm is defined as

$$||f||_{H^r(\Gamma)}=\sup_{0\neq g\in H^{-r}(\Gamma)}\frac{|(f,g)_{1/2}|}{||g||_{H^{-r}(\Gamma)}}.$$

Now, we can state the variational formulation associated with (4):

Problem 1.3. **Variational form of equations of linear elasticity**

$$\begin{array}{ll} \textbf{For a given load} & t^*\in\left(H^{-1/2}\left(\Gamma_{N_t}\right)\right)^3,\\ \textbf{find} & u\in\left(H^1(\Omega)\right)^3\\ \textbf{such that} & u|_{\Gamma_D}=u^*\\ \textbf{and} & a_\Omega(u,v)=l_\Omega(v)\quad\forall v\in H^1_{0,\Gamma_D}(\Omega). \end{array} \tag{5}$$

The bilinear form $a_\Omega(.,.)$ is defined as

$$a_\Omega(u,v)=\int_\Omega\nabla u:\mathbb{C}:\nabla v dx$$

and $l_\Omega(.)$ is the linear form

$$l_\Omega(v)=\int_{\Gamma_{N_t}}t^*\cdot v ds.$$

The Lax-Milgram lemma (cf., e.g., [**16**]) provides existence and uniqueness of the solution to the variational formulation (5) of the elasticity problem.

The aim of topology optimization in elasticity is to find the domain which is optimal in the sense that it minimizes a certain cost functional J while fulfilling a set of given constraints. Usually, J depends on both the domain, which in our case is the design variable, and the state variables. Thus, if s is a yet to be determined parametrization of the domain $\Omega = \Omega(s)$, we have

$$J = J(s, u).$$

In principal, the cost function can consist of any structural response or a combination of various responses. However, to fix ideas we choose the compliance

$$J(u) = \int_{\Gamma_{N_t}} t^* \cdot u ds,$$

which is a widely used criterion. Using Green's formula, the compliance can also be computed by

$$J(s, u) = \int_{\Omega(s)} [\mathbb{C} : \varepsilon(u)] : \varepsilon(u) dx.$$

The constraints under consideration are a volume constraint which reflects a desired volume reduction and the variational form of the equations of linear elasticity (5). The set of admissible domains is given by

$$\mathcal{U}_{ad} = \left\{ s \mid \Omega(s) \subset D, (\Gamma_{N_t} \cup \Gamma_D) \subset \partial\Omega(s) \right\}.$$

The continuous topology optimization problem can thus be formulated as follows:

Problem 1.4. **Continuous topology optimization problem**

$$\begin{aligned} &\min\nolimits_{s \in \mathcal{U}_{ad}} && J(s, u) = \int\limits_{\Omega(s)} [\mathbb{C} : \varepsilon(u)] : \varepsilon(u) dx \\ &\textbf{subject to} && \int\limits_{\Omega(s)} dx \leq V_{target} \qquad \textit{and} \\ & && a_{\Omega(s)}(u, v) = l_{\Omega(s)}(v). \end{aligned} \tag{6}$$

The volume constraint is considered in the objective function as follows:

$$j(s) = J(s, u) + \ell(|\Omega(s)| - V_{target}). \tag{7}$$

Here, ℓ can be interpreted as a Lagrange multiplier and $|\Omega(s)|$ is the Lebesgue measure of Ω.

It is well known that the optimization problem (6) is ill-posed in the sense that generally no solution exists in the original set of admissible functions $\mathcal{U}_{ad}$ (see, e.g., [**15, 5, 30**]). Since the problem is a non-convex optimization problem, there are often many local minima which usually have different topology. Without further constraints, the solution would be a microstructure with infinitesimal features [**35**]. To overcome this problem, either relaxation or restriction of $\mathcal{U}_{ad}$ is required to make the problem well-posed. We introduce a constrained set of admissible functions $\mathcal{U}_{ad}^R$ the choice of which will be discussed in chapter 2 where we fix the parametrization s of Ω.

1.2. The Discrete Topology Optimization Problem

In order to solve the topology optimization problem (6) numerically, we have to discretize it. On one hand, this concerns the solution of the elasticity problem (5) and the evaluation of the objective function. On the other hand, we have to use a discrete representation of the structure's boundary $\partial\Omega$.

It is possible to use one and the same discretization for both parts. This approach is used, e.g., in [**26**], where the elasticity problem is solved on a hexahedral mesh and the boundary of the structure is defined by corners, edges, and faces of the same mesh, and to some extent in homogenization methods [**10, 58, 18, 6**], which employ varying densities and partly construct black-and-white designs in a post-processing step.

However, the purposes and required features of the two discretizations differ significantly. The solution of the elasticity problem is concerned with the entire domain, whereas the domain representation can be reduced to a representation of the structure's boundary.

To meet these requirements, we distinguish between the discretization of the equations of elasticity and the discrete representation of Ω. Accordingly, we introduce two discretization parameters h and ϑ. The latter discretizes the boundary representation s, the discrete version of which will be denoted by s_ϑ. Naturally, the choice of ϑ depends on the choice of s and will therefore be discussed in section 2.2.

For the solution of the equations of elasticity (5) we employ the finite element method (FEM). For an introduction to this method we refer the reader to [**20, 61**]. Thorough presentations can be found in, e.g., [**12, 16, 65**].

The domain Ω is replaced by a polyhedral domain Ω_h, and the parts Γ_D and Γ_N where Dirichlet and Neumann boundary conditions are applied, are approximated by Γ_{Dh} and Γ_{Nh}, respectively. Γ_{Nh} is divided into the disjoint parts Γ_{N_0h} and Γ_{N_th}. We assume that the discrete equivalent of relation (1) is valid, too:

$$\partial\Omega_h = \Gamma_{Dh} \cup \Gamma_{N_th} \cup (\Gamma_{N_0h} \cap \partial\Omega_h) \cup \Gamma_{0h}.$$

Let us define the space of finite-dimensional shape functions by

$$\mathcal{V}_{\Omega_h} = \left\{ v \in (H^1(\Omega_h))^3 : v|_{\Gamma_{Dh}} = 0 \right\}$$

and let us denote the boundary conditions applied on Γ_{Dh} and $\Gamma_{N_t h}$ by u_h^* and t_h^*, respectively. With the bilinear form

$$a_{\Omega_h}(u_h, v_h) = \int_{\Omega_h} \nabla u_h : \mathbb{C} : \nabla v_h dx$$

and the linear form

$$l_{\Omega_h}(v_h) = \int_{\Gamma_{N_t h}} t_h^* \cdot v_h ds,$$

we can formulate the discrete version of the equations of linear elasticity:

Problem 1.5. **Discrete equations of elasticity**

$$\begin{array}{ll} \textbf{For a given load} & t_h^* \in \left(H^{-1/2}\left(\Gamma_{N_t h}\right)\right)^3, \\ \textbf{find} & u_h \in \left(H^1\left(\Omega_h\right)\right)^3 \\ \textbf{such that} & u_h|_{\Gamma_{Dh}} = u_h^* \\ \textbf{and} & a_{\Omega_h}(u_h, v_h) = l_{\Omega_h}(v_h) \quad \forall v_h \in \mathcal{V}_{\Omega_h}. \end{array} \tag{8}$$

Denoting the space of discrete admissible domains by $\mathcal{U}^R_{ad,\vartheta}$, the discrete topology optimization problem reads as follows:

Problem 1.6. **Discrete topology optimization problem**

$$\begin{array}{ll} \min_{s_\vartheta \in \mathcal{U}^R_{ad,\vartheta}} & J(s_\vartheta, u_h) = \int\limits_{\Omega(s_\vartheta)} \left[\mathbb{C} : \varepsilon_h(u_h)\right] : \varepsilon_h(u_h) dx \\ \textbf{subject to} & \int\limits_{\Omega(s_\vartheta)} dx \le V_{target} \qquad \textbf{and} \\ & a_{\Omega(s_\vartheta)}(u_h, v_h) = l_{\Omega(s_\vartheta)}(v_h). \end{array}$$

Chapter 2 contains the description of the boundary representation and the restriction of the set of admissible domains we use to make the problem well-posed. The discretization of Ω is described in chapter 3, where we present a fast and robust tetrahedral mesh generator. Chapter 4 consists of the numerical algorithm used for the actual solution of the optimization problem. There we also describe the structural update.

CHAPTER 2

Boundary Representation Using a Level Set Function

Having formulated the optimization problem we now want to focus on the parametrization s of the domain Ω. In doing so, we have to consider two properties: During the optimization process, all types of topology changes as for example creation or merging of holes can occur. Hence, flexibility of s is paramount. On the other hand we want to solve complex and large industrial optimization problems. Thus, the boundary representation method must be robust and computationally efficient. Naturally, s must provide a sufficiently precise approximation of Ω.

There are two classes of domain representations. On one hand, explicit representations as for example analytic expressions of simple geometric bodies, voxelizations, triangulated meshes for polygonal structures, or non-uniform rational Bézier splines (NURBS), are widely used in, e.g, computer graphics, computer aided design (CAD), and geometric modeling. It is easy to manually modify explicit structures by moving mesh nodes or control points, and they can be rendered very fast.

On the other hand there are implicit representations like volume of fluid (VOF) or the representation by a level set function. This method was instigated by Osher and Sethian [**38**] for propagating interfaces and free boundaries and is used in modeling of a large variety of physical phenomena.

During the last years, the level set approach has been introduced in the field of shape and topology optimization [**3, 4, 13, 37, 52, 60**]. Indeed, the method has all the characteristics we require: Level set functions can represent any domain and handle all kinds of topology changes such as nucleation, merging, and deletion of holes. In the discrete case, they enable an accurate description of the boundaries on a fixed Cartesian grid which leads to fast numerical algorithms. The use of a narrow band technique (cf. Figure 2.4) reduces memory consumption and computation times even further. In [**51**], the effect of the narrow band technique is estimated as follows: Level set computations that are performed over the entire domain require $\mathcal{O}(N^3)$ operations, where N is the number of nodes in the grid. If the number of points of the boundary in 3D is estimated as $\mathcal{O}(N^2)$, the operation count is reduced to $\mathcal{O}(kN^2)$, where

k is the number of grid cells in the narrow band. Our method of updating the structure makes extensive use of level set functions and is simple, stable, and very efficient. This will be discussed in section 4.3.

The present chapter introduces the concept of level set functions for boundary representation. Section 2.1 describes the continuous case, whereas section 2.2 contains the discrete case and the narrow band technique. Both sections finish with a level set-based reformulation of the continuous and discrete topology optimization problem, respectively. The chapter is concluded by a discussion of regularization techniques and possible restrictions of the set of admissible functions $\mathcal{U}_{ad}$ in section 2.3.

2.1. Boundary Representation by a Continuous Level Set Function

The main idea of the level set function approach is to represent the boundary $\partial\Omega(s)$ of $\Omega(s)$ as the zero level set of a function $\varphi \in C^0(D)$, which is negative in the structure's interior and positive outside:

$$\begin{cases} \varphi(x) & < & 0 & \Longleftrightarrow & x \in \Omega, \\ \varphi(x) & > & 0 & \Longleftrightarrow & x \in D \setminus \overline{\Omega}, \\ \varphi(x) & = & 0 & \Longleftrightarrow & x \in \partial\Omega. \end{cases}$$

Thus, we have

$$\partial\Omega(\varphi) = \{x \in D \mid \varphi(x) = 0\}.$$

The actual domain Ω is then given as the support of the negative part $\varphi^- := \frac{1}{2}(|\varphi| - \varphi)$ of φ:

$$\Omega(\varphi) = \operatorname{supp}(\varphi^-).$$

To simplify notation, we write in the following Ω_φ and $\partial\Omega_\varphi$ instead of $\Omega(\varphi)$ and $\partial\Omega(\varphi)$, respectively. Note that s, which represented an arbitrary parametrization of Ω, is now substantiated by φ.

One possible choice of φ is the constrained signed distance function. Let us denote the distance of a point x to the structure's boundary by $d(x) := \min_{y \in \partial\Omega_\varphi} |x - y|$. Then the constrained distance function is defined as

$$\varphi(x) = \begin{cases} -\alpha & \text{for } x \in \Omega_\varphi,\ d(x) \geq \alpha, \\ -d(x) & \text{for } x \in \Omega_\varphi,\ d(x) < \alpha, \\ +d(x) & \text{for } x \notin \Omega_\varphi,\ d(x) < \alpha, \\ +\alpha & \text{for } x \notin \Omega_\varphi,\ d(x) \geq \alpha \end{cases}.$$

Since φ is used only for boundary representation, it is sufficient to use the correct signed distance function in regions close to the structures boundary. In more remote regions, only the sign of the function is of interest. The width

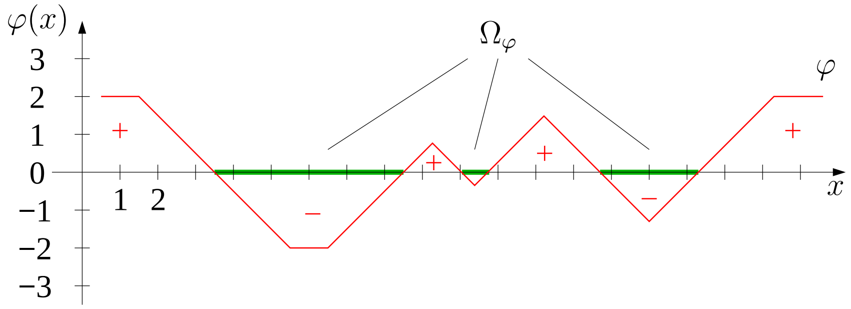


FIGURE 2.1. One-dimensional continuous level set function. Ω_φ is the support of the negative part of φ.

of the neighborhood of $\partial\Omega$ in which φ equals the signed distance function is denoted by c.

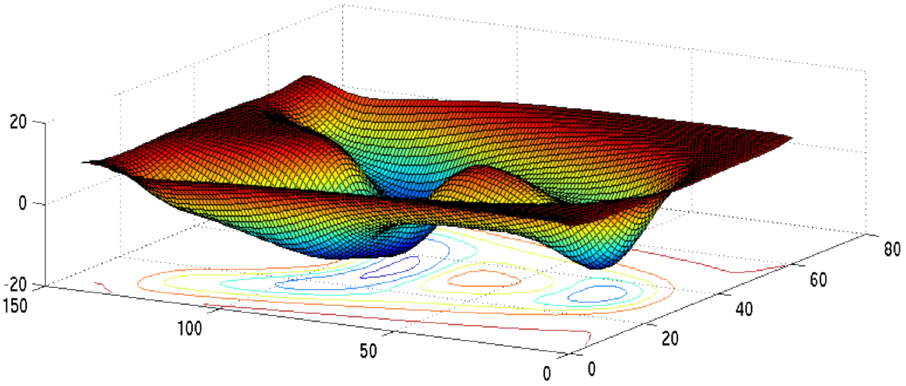


FIGURE 2.2. Two-dimensional continuous level set function representing a simple truss structure. The yellow area marks the zero level set of φ. Contour lines are plotted to illustrate the shape of the structure.

Figure 2.1 shows an example of a one-dimensional domain which is represented as described above. Ω_φ has two holes. The constant α is set to 2. A two-dimensional example is given in Figure 2.2. Here, we represent a simple truss structure as the zero level set (yellow area) of a constrained level set function. Using φ as design variable in the optimization problem, the set of admissible domains is given by

$$\mathcal{U}_{ad} = \left\{\varphi \in C^0(D,\mathbb{R}) \mid \Omega_\varphi \subset D, (\Gamma_{N_t} \cup \Gamma_D) \subset \partial\Omega_\varphi\right\}$$

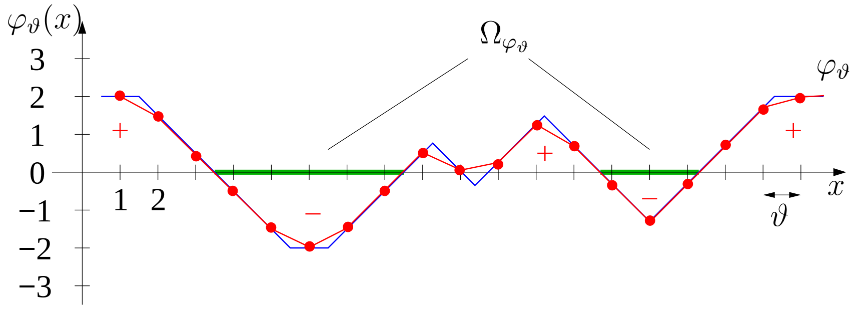


FIGURE 2.3. One-dimensional discrete level set function. The red curve represents a linear interpolation of the values of the continuous level set function (blue) in the grid points.

and (6) can be written as

$$
\begin{aligned}
&\min_{\varphi\in\mathcal{U}_{ad}} && J(\varphi,u) = \int\limits_{\Omega_\varphi} [\mathbb{C}:\varepsilon(u)]:\varepsilon(u)dx \\
&\text{subject to} && \int\limits_{\Omega_\varphi} dx \le V_{target} \quad \text{and} \\
& && a_{\Omega_\varphi}(u,v) = l_{\Omega_\varphi}(v).
\end{aligned}
\tag{9}
$$

2.2. Boundary Representation by a Discrete Level Set Function

Usually, level set functions are discretized on a fixed Cartesian grid. If we denote the width of the grid by ϑ, the regular Cartesian grid in $\mathbb{R}^3$ is defined as

$$\mathcal{G}_\vartheta = \left\{x\in\mathbb{R}^3 | \exists\, \{i,j,k\} \in \mathbb{N}^3 \ s.t. \ x = (i\vartheta, j\vartheta, k\vartheta)\right\}.$$

Restricting the continuous level set function φ introduced in section 2.1 to $\mathcal{G}_\vartheta$, we obtain a discrete level set function $\varphi_\vartheta : \mathcal{G}_\vartheta \to \mathbb{R}$ which represents a discretization $\Omega_{\varphi_\vartheta}$ of the domain Ω_φ:

$$
\begin{cases}
\varphi_\vartheta(x) < 0 \iff x\in\Omega_{\varphi_\vartheta}, \\
\varphi_\vartheta(x) > 0 \iff x\in D\setminus\overline{\Omega_{\varphi_\vartheta}}, \\
\varphi_\vartheta(x) = 0 \iff x\in\partial\Omega_{\varphi_\vartheta}.
\end{cases}
$$

Naturally, φ_ϑ is defined analogously to the continuous case:

$$
\varphi_\vartheta(x) = \begin{cases}
-\alpha & \text{for } x\in\Omega_{\varphi_\vartheta},\ d(x)\ge\alpha, \\
-d(x) & \text{for } x\in\Omega_{\varphi_\vartheta},\ d(x)<\alpha, \\
+d(x) & \text{for } x\notin\Omega_{\varphi_\vartheta},\ d(x)<\alpha, \\
+\alpha & \text{for } x\notin\Omega_{\varphi_\vartheta},\ d(x)\ge\alpha
\end{cases}.
$$

Figure 2.3 displays the discrete version of the one-dimensional level set function introduced in the previous section for $\vartheta = 1$. In order to determine the

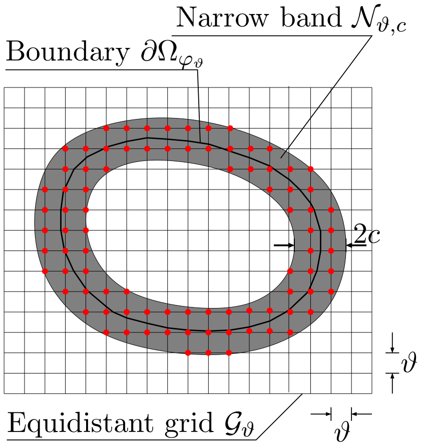

Narrow band $\mathcal{N}_{\vartheta,c}$
Boundary $\partial\Omega_{\varphi_\vartheta}$
$2c$
ϑ
Equidistant grid $\mathcal{G}_\vartheta$
ϑ

FIGURE 2.4. Narrow band representation of a domain. The thick black line represents the structure's boundary $\partial\Omega_{\varphi_\vartheta}$, the shaded area is the narrow band for the continuous case, and the red dots are the points belonging to $\mathcal{N}_{\varphi_\vartheta,c}$.

zero level set, we interpolate the discrete values of φ_ϑ linearly and compute the roots of the resulting piecewise linear function. Obviously, $\Omega_{\varphi_\vartheta}$ does not contain all the features of Ω_φ. The two holes which are correctly represented in the continuous case (blue curve) are merged in the discrete case (red curve). Correspondingly, small holes which do not contain a grid point will be erased. This is not shown in the picture, but it follows immediately if φ_ϑ is multiplied by -1 and the inverse domain is considered.

As mentioned in the introduction of this chapter, memory consumption and CPU time can be significantly reduced if a narrow band technique is used. The main idea behind this concept is that knowledge of the values of the level set function in a neighborhood of the boundary $\partial\Omega_h$ is sufficient for construction of a piecewise linear boundary representation. For most of the grid points we only need to know if they lie inside or outside the structure.

Considering the Cartesian grid $\mathcal{G}_\vartheta$, we define the *narrow band of width* c of $\partial\Omega_{\varphi_\vartheta}$ as

$$\mathcal{N}_{\varphi_\vartheta,c} := \left\{ p \in \mathcal{G}_\vartheta | \min_{z \in \partial\Omega_{\varphi_\vartheta}} |z - p| \le c \right\}.$$

For a two-dimensional example, $\mathcal{N}_{\varphi_\vartheta,c}$ is illustrated in Figure 2.4. The thick black line represents the structure's discretized boundary $\partial\Omega_{\varphi_\vartheta}$, the shaded area is the narrow band for the continuous case, and the red dots are the points belonging to $\mathcal{N}_{\varphi_\vartheta,c}$.

Choosing φ_ϑ as design variable, the discrete counterpart of (9) is the following:

PROBLEM 2.1. **Discrete topology optimization problem using a level set function**

$$\begin{aligned} \min_{\varphi_\vartheta \in \mathcal{U}_{ad,\vartheta}} \quad & J(\varphi_\vartheta, u_h) = \int\limits_{\Omega_{\varphi_\vartheta}} [\mathbb{C} : \varepsilon_h(u_h)] : \varepsilon_h(u_h) dx \\ \textbf{subject to} \quad & \int\limits_{\Omega_{\varphi_\vartheta}} dx \leq V_{target} \qquad \textbf{and} \\ & a_{\Omega_{\varphi_\vartheta}}(u_h, v_h) = l_{\Omega_{\varphi_\vartheta}}(v_h). \end{aligned}$$

2.3. Regularization

As mentioned in chapter 1, the topology optimization problem (6) is ill-posed and restriction of the set of admissible domains is necessary. Loosely speaking, we want to prevent creation of too small holes and development of too small features.

In terms of set operations this can be achieved by application of the morphological set transformations *erosion* and *dilation*:

Definition 2.2. *Let $\widetilde{\Omega}$ be the modified domain and let S be an arbitrary but fixed subset of $\mathbb{R}^3$ containing the origin. S is called* **structural element**. *Let S_x be the set obtained by translation of S by a vector x.*
Then, **erosion of $\widetilde{\Omega}$ by** *S is defined as*

$$\widetilde{\Omega} \ominus S = \left\{ z \in \mathbb{R}^3 | S_z \subseteq \widetilde{\Omega} \right\}. \tag{10}$$

Similarly, **dilation of $\widetilde{\Omega}$ by** *S is defined as*

$$\widetilde{\Omega} \oplus S = \left\{ z \in \mathbb{R}^3 | S_z \cap \widetilde{\Omega} \neq \emptyset \right\}. \tag{11}$$

Consecutive application of these operations leaves the main features of $\widetilde{\Omega}$ untouched and affects only unwanted fine details.

In the framework of level set functions, erosion and dilation can be formulated as convolutions. We denote the filtering kernel, which is a functional on $\mathbb{R}^3$, by $\widetilde{k}$, and the level set function representing the modified domain $\widetilde{\Omega}$ by $\widetilde{\varphi}$. The convolution of $\widetilde{k}$ and $\widetilde{\varphi}$ is defined as

$$(\widetilde{k} * \widetilde{\varphi})(x) = \int_D \widetilde{k}(y) \widetilde{\varphi}(x - y) dy. \tag{12}$$

Convolution corresponds to dilation, whereas deconvolution, which we denote by the operator $\star$, is the equivalent to erosion.

The restricted set of admissible domains is given by

$$\mathcal{U}_{ad}^R = \left\{ \varphi \mid \Omega_\varphi \subset D, (\Gamma_{N_t} \cup \Gamma_D) \subset \partial\Omega_\varphi, \exists \phi \in C^0(D, \mathbb{R}) : \varphi = \widetilde{k} \star (\phi * \widetilde{k}) \right\},$$

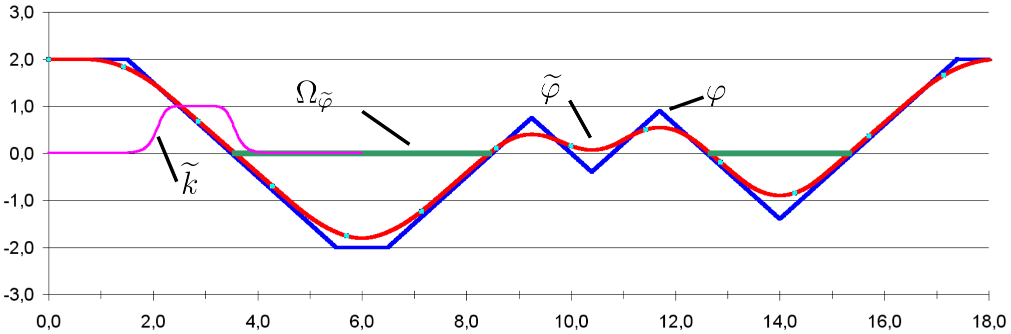


FIGURE 2.5. Erosion of a level set function. The level set function φ is convoluted with the filtering kernel $\widetilde{k}$. The domain $\Omega_{\widetilde{\varphi}}$ associated with $\widetilde{\varphi}$ has one big hole instead of the two smaller holes which were present in the original domain ($\widetilde{\varphi}$ is not negative between 10 and 11). The fine feature separating the holes has been removed by erosion.

where $\widetilde{k}$ is a fixed filtering kernel, e.g.,

$$\widetilde{k}(x) = \frac{1}{0.1 + x^4}$$

as originally proposed in [**54**].

To illustrate erosion, we give a 1D-example in Figure 2.5. The level set function φ represents a domain with a two holes separated by a small feature centered at $x = 10.4$. The filtering kernel $\widetilde{k}$ is chosen as above but displayed with an offset for better visualization. The red curve is the convolution $\widetilde{\varphi} = \varphi * \widetilde{k}$. Obviously, the two holes present in the original domain have been merged and the separating feature has been erased.

REMARK 2.3. The deconvolution operator $\star$ is not the inverse of the convolution operator $*$. It is only a short notation for the algorithm-based process of reconstructing the second argument of a convolution where the filtering kernel is known. Naturally, $\star$ is not commutative. The order of the arguments determines which one is the filtering kernel.

In the discrete case, restriction of $\mathcal{U}_{ad,\vartheta}$ is actualized as in the continuous case. Again, we apply erosion and dilation, but since we are working on the Cartesian grid $\mathcal{G}_\vartheta$, the definitions look as follows:

Definition 2.4. *Let Ω_ϑ be the discretized modified domain and let S_ϑ be an arbitrary but fixed subset of the Cartesian grid $\mathcal{G}_\vartheta$ containing the origin. Denoting by $S_{\vartheta,z}$ the set obtained by translation of S_ϑ by the vector z, we can define* **erosion** *and* **dilation** *as*

$$\widetilde{\Omega_\vartheta} \ominus S_\vartheta = \left\{ z \in \mathcal{G}_\vartheta | S_{\vartheta,z} \subseteq \widetilde{\Omega_\vartheta} \right\}$$

and

$$\widetilde{\Omega_\vartheta} \oplus S_\vartheta = \left\{ z \in \mathcal{G}_\vartheta | S_{\vartheta,z} \cap \widetilde{\Omega_\vartheta} \neq \emptyset \right\},$$

respectively.

In the framework of discrete level set functions, the integration in the convolution (12) becomes a summation over all points belonging to the support of the kernel:

$$(\widetilde{k}_\vartheta * \widetilde{\varphi_\vartheta})(x) = \frac{1}{|\mathrm{supp}(\widetilde{k}_\vartheta)|} \sum_{y \in \mathrm{supp}(\widetilde{k}_\vartheta)} \widetilde{k}_\vartheta(y) \widetilde{\varphi_\vartheta}(x-y).$$

The simplest choices of $\widetilde{k}_\vartheta$ are the seven point stencil and the cube of 27 points having x as center point, where $\widetilde{k}_\vartheta \equiv 1$. Other choices can use larger neighborhoods of x and/or assign different values of $\widetilde{k}_\vartheta$ to the points $x \in \mathrm{supp}(\widetilde{k}_\vartheta)$, thereby strengthening or weakening the effect of erosion and dilation. The higher the value of $\widetilde{k}_\vartheta$ in the center point compared to the surrounding points, the stronger is the discrete convolution.

Finally, the restricted set of admissible domains in the discretized topology optimization problem is given by

$$\begin{aligned} \mathcal{U}^R_{ad,\vartheta} = \{ \ \varphi_\vartheta \ | \quad & \Omega_{\varphi_\vartheta} \subset D, (\Gamma_{Nth} \cup \Gamma_{Dh}) \subset \partial\Omega_{\varphi_\vartheta}, \\ & \exists \phi_\vartheta \in C^0(D, \mathbb{R}) : \varphi_\vartheta = \widetilde{k}_\vartheta \star (\phi_\vartheta * \widetilde{k}_\vartheta) \Big\}. \end{aligned}$$

CHAPTER 3

Mesh Generation

As mentioned in chapter 1, we employ the finite element method (FEM) for solution of the elasticity problem (8). There are two major tasks connected to this method: On the one hand, the domain has to be discretized in a proper way. On the other hand, a large sparse system of linear equations has to be set up and solved.

Intuitively, the latter will be considered the more difficult and time-consuming problem. However, this is generally not true. Despite the enormous progress in the young field of automated mesh generation, which started to evolve in the middle of the 1980s, the meshing process is still often the by far harder challenge and may consume the by far larger part of the total computing time.

In the current chapter, we present a modification of a mesh generation algorithm belonging to the group of octree techniques. In order to motivate this and to justify it in view of the large number of existing mesh generation algorithms, we have to anticipate a detail of the implementation of the optimization algorithm, which is presented in section 4.4. There we explain that we remesh the structure in every iteration.

Frequent remeshing made it necessary to modify an existing algorithm and to adapt it to our demands. Clearly, the computing time needed for generation of a finite element mesh becomes a critical factor if the process is repeated several times in a single optimization run. Therefore, speed and efficiency of the algorithm are of great importance. The remeshing itself is in turn motivated by the casting process and a gain in accuracy of the solution to the elasticity equations in modified domains.

Another point that has to be considered is robustness of the mesh generation algorithm. In the course of the optimization process, structures can become very complex. To ensure that the optimization is not aborted due to an error in mesh generation before a minimum is reached, we have to create a mesh under any circumstances. Naturally, as one cannot have everything, this may lead to a reduction in accuracy of the domain representation. However, we think that the continuation of the optimization process is of greater importance.

As we will show in section 4.3.2, our algorithm is especially suitable for topology optimization problems which may involve significant changes of the structure. A large fraction of the computationally expensive part of the mesh can be kept, which reduces computing time significantly.

In the following, we give an abstract description of the concept of a finite element mesh. Thereafter, we describe the mesh generation algorithm, the main concepts and ideas of which are taken from Zhang et al. [**64**]. To illustrate the process, we first describe uniform triangulation of two-dimensional domains (section 3.1.1). This approach will be extended to three dimensions (section 3.1.2), and finally we present adaptive triangulation and tetrahedralization in two and three space dimensions, respectively (sections 3.2.1 and 3.2.2).

Intuitively, tetrahedral mesh generation is the process of approximating the domain Ω under consideration by a set of tetrahedra which are joined along their faces. For a more formal description, we use the notion of a simplex and a simplicial complex, respectively (cf. [**57**]):

Definition 3.1. *A* **simplex** *or* n-**simplex** *is the convex hull of a set of points* $\{P_i \in \mathbb{R}^n, i = 1, ..., n+1\}$ *which are affinely independent, i.e., for each* i *the set* $\{P_i - P_j | j = 1, ..., n+1\}$ *is linearly independent.* 0*-simplices are called* **vertices***,* 1*-simplices* **edges***,* 2*-simplices* **faces***, and* 3*-simplices* **tetrahedra***.*

Definition 3.2. *A* **simplicial complex** $\mathcal{K}$ *consists of a set of vertices* $V = \{v_i \in \mathbb{R}^3, i = 1, ..., m\}$ *and the simplices of* $\mathcal{K}$*, i.e., a set of non-empty subsets of* V *fulfilling the following conditions: Each non-empty subset containing a single vertex* v_i *is a simplex in* $\mathcal{K}$*, and every non-empty subset of a simplex in* $\mathcal{K}$ *is again a simplex in* $\mathcal{K}$*.*

With these concepts, we can formally describe mesh generation as construction of a simplicial complex.

3.1. Uniform Mesh Generation

The basic idea of the mesh generation process is to subdivide the domain Ω into two disjunct parts, a regular interior and a boundary layer:

$$\Omega = \Omega_{int} \cup \Omega_{bor}$$

such that $\Omega_{int} \cap \Omega_{bor} = \emptyset$ and $\overline{\Omega_{int}} \cap \overline{\Omega_{bor}} = \partial\Omega_{int}$. For this purpose, we introduce an underlying regular Cartesian grid $\mathcal{G}_h$ of width h and define Ω_{int} as the union of all grid cells which are completely contained in Ω, whereas Ω_{bor} is the remaining thin layer of width $\leq h : \Omega_{bor} = \Omega \setminus \Omega_{int}$.

Thus, the parts are chosen in such a way that a large portion of the structure can be meshed extremely fast by using pre-computed configurations and a thin layer, the meshing of which requires more computing time, provides sufficient accuracy of domain representation.

For practical purposes and the sample computations we performed, we obtained good results choosing

$$h = \frac{1}{c} \min \left\{ x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min} \right\}, \tag{13}$$

where $\{x, y, z\}_{\{min,max\}}$ denote the minimal and maximal extension of the structure in all coordinate directions, respectively, and the constant c is chosen between 20 and 50.

For a more formal description of Ω_{int} and Ω_{bor} and the actual mesh generation, we adopt some concepts which are used in [**64**]:

Definition 3.3. *Let E be an edge of a cell C in a Cartesian grid $\mathcal{G}$ and P_1, P_2 be the endpoints of E.*
E is called a **sign change edge** *if*

$$(\varphi(P_1) \leq 0 \ \wedge \ \varphi(P_2) > 0) \ \vee \ (\varphi(P_2) \leq 0 \ \wedge \ \varphi(P_1) > 0),$$

i.e., one of its endpoints lies inside and the other one lies outside the domain Ω.

Definition 3.4. *A cell C of a Cartesian grid $\mathcal{G}$ is called* **boundary cell** *if at least one of its edges is a sign change edge.*

Definition 3.5. *A cell C of a Cartesian grid $\mathcal{G}$ is called* **interior cell** *if all of its corner points $P_i, i = 1, \ldots, 4$ ($P_i, i = 1, \ldots, 8$ in 3D) fulfill $\varphi(P_i) \leq 0$, i.e., all corner points belong to the domain Ω.*

Using these notions we define the interior part Ω_{int} of Ω as the union of all interior cells and the boundary part Ω_{bor} as the intersection of the union of all boundary cells with Ω. Figure 3.1 illustrates the domain decomposition in the 2D case.

From the above definitions it is clear that Ω_{int} and Ω_{bor} depend on the width h of the chosen Cartesian grid $\mathcal{G}_h$. Nevertheless, for the sake of simplicity, we drop the mesh width in notation and keep in mind that $\Omega_{int} = \Omega_{int}(h)$ and $\Omega_{bor} = \Omega_{bor}(h)$.

3.1.1. Uniform 2D Triangulation.

In the two-dimensional uniform case, Ω_{int} is meshed by simple subdivision of each interior cell into two triangles. To avoid anisotropy, we turn the diagonal in every other cell.

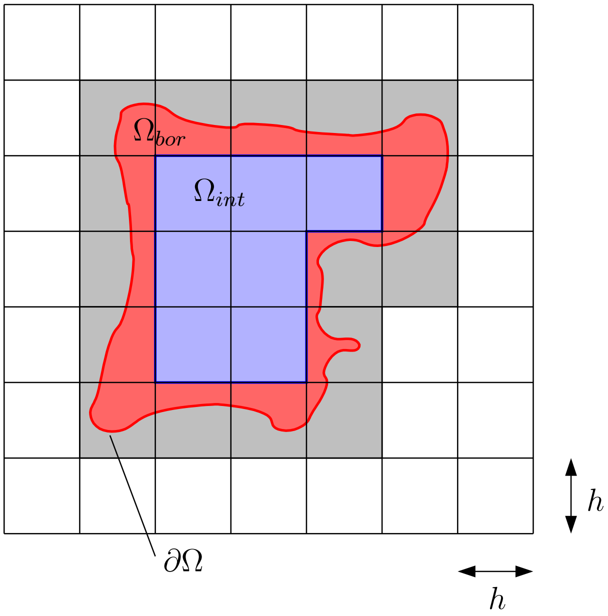


FIGURE 3.1. Decomposition of the domain in interior Ω_{int} (blue) and boundary part Ω_{bor} (red). Shaded cells are boundary cells.

For mesh generation in Ω_{bor} we need a piecewise linear approximation $\mathcal{T}$ of $\partial\Omega$. $\mathcal{T}$ consists of a set of nodes $\{P_1, \ldots, P_N\} \subset \mathbb{R}^2$, which, for example, might be chosen such that each boundary cell contains exactly one node, and a set of edges $\left\{E_{ij} = \overline{P_i P_j}\right\}$, where P_i, P_j belong to neighbored boundary cells. Since our approximation depends on the width h of the underlying grid, we denote it by $\mathcal{T}_h$.

A high-quality and feature-preserving approximation can be obtained by the dual contouring method [**29**], which uses volumetric Hermite data, i.e., position and normal information of intersections of sign change edges and the surface to be approximated to compute one minimizer for each boundary cell.

The minimizers are obtained by minimization of a predefined quadratic error function e_q. Garland and Heckbert [**25**] proposed the following:

$$e_{q1}(x) = \sum_i (\mathbf{n}_i \cdot (x - p_i))^2.$$

Here, p_i are the intersection points of the boundary cell's sign change edges and the surface $\partial\Omega$ and $\mathbf{n}_i$ are the unit normal vectors to the surface in p_i. $e_{q1}(x)$ is the sum of squared distances of the vertex x to the straight lines defined by the intersection points and normals.

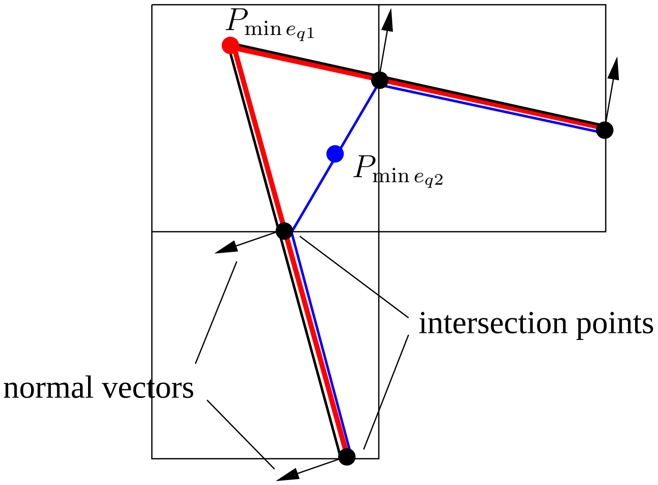


FIGURE 3.2. Preservation and smoothing of discontinuities of $\partial\Omega$ by using e_{q1} and e_{q2} in minimizer computation, respectively. e_{q1} (red curve) preserves the sharp corner, e_{q2} (blue curve) smoothes the structure.

If only the positions of intersection points are considered and normal information is neglected, one can define another quadratic error function

$$e_{q2}(x) = \sum_i (x - p_i)^2.$$

It is immediately clear that the minimizer p_{min} of $e_{q2}(x)$ is the arithmetic mean of the intersection points, $p_{min} = \frac{1}{n}\sum_{i=1}^{n} p_i$.

By connecting the minimizers of respectively two cells sharing a sign change edge a line segment is associated to each sign change edge. The union of all these line segments forms the desired approximation $\mathcal{T}$ of $\partial\Omega$.

The advantage of $e_{q1}(x)$ is that the resulting triangulation of the domain boundary can preserve discontinuities and sharp features of $\partial\Omega$, whereas the solution of $e_{q2}(x)$ is equivalent to Taubin et al's discretization of the Laplace-Beltrami operator over triangular surfaces [**59, 62**] and corresponds to a smoothing of $\partial\Omega$. The difference between the two quadratic error functions is illustrated in Figure 3.2.

When the minimizers of all boundary cells have been computed, the actual mesh generation for Ω_{bor} can be performed. To describe this process we introduce two more concepts following [**64**]:

Definition 3.6. *A corner point P of a boundary cell of a Cartesian grid $\mathcal{G}$ is called* **interior point** *if* $\varphi(P) \leq 0$.

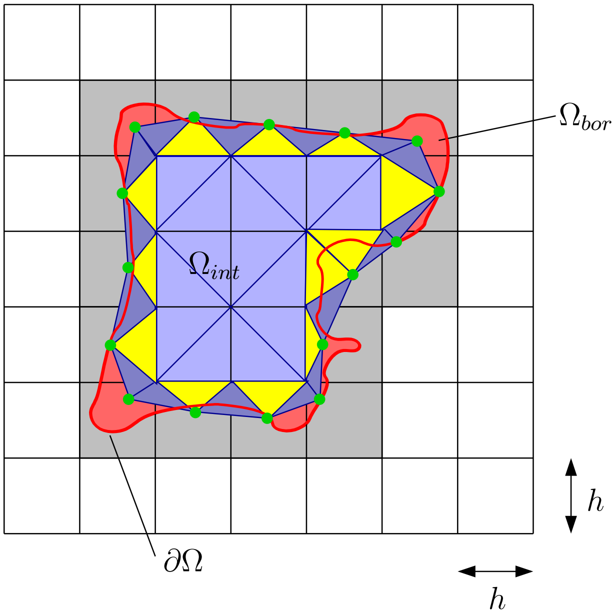


FIGURE 3.3. Uniform triangulation. First, Ω_{int} is meshed. After computation of the minimizers (green dots), Ω_{bor} is meshed in two steps: Triangles associated to interior edges and one minimizer are constructed (yellow). Then, two minimizers and an interior point form the dark blue triangles. The red line shows the structure's boundary $\partial\Omega$.

Definition 3.7. *An edge E with endpoints P_1, P_2 of a boundary cell of a Cartesian grid $\mathcal{G}$ is called* **interior edge** *if both P_1 and P_2 are interior points.*

In the 2D case, mesh generation in Ω_{bor} is done in two steps: First, a triangle is created for each interior edge. The edge forms the triangle's basis, and the minimizer of the boundary cell to which the edge belongs is the third node. The triangles created by this step are the yellow ones in Figure 3.3. In a second step, a triangle is assigned to each sign change edge. One of the edge's endpoints is an interior point. This point and the minimizers of the two boundary cells sharing the sign change edge form a triangle. In Figure 3.3, these triangles are indicated by blue color.

3.1.2. Uniform 3D Tetrahedralization.

In three space dimensions, the uniform mesh generation procedure is very similar to the 2D case. Again, the domain Ω is divided into Ω_{int} and Ω_{bor}, which are defined as the union of all interior cells and the intersection of the union of all boundary cells and the domain Ω, respectively.

As in 2D, Ω_{int} is meshed by subdivision of cells. Each cell is divided into five tetrahedra. To avoid the diagonal choosing problem and to generate a consistent tetrahedral mesh, one can use two different cell decompositions for adjacent cells. The two configurations are shown in Figures 3.4(a) and 3.4(b), respectively.

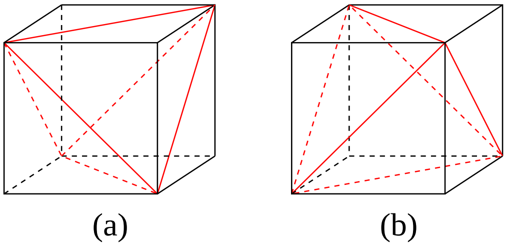


FIGURE 3.4. Decomposition of interior cells into tetrahedra. Two different configurations are used to avoid the diagonal choosing problem.

Mesh generation for Ω_{bor} is slightly more complicated than in 2D, but the ideas are the same. First, a minimizer is computed for each boundary cell. The quadratic error functions we use here are e_{q1} and e_{q2} introduced in section 3.1.1.

Respectively four boundary cells share a sign change edge. Their minimizers are connected to a quad and subdivided along the diagonal, which associates two triangles to each sign change edge. The union of all these triangles forms a triangulation $\mathcal{T}_h$ of $\partial\Omega$.

For the detailed description of the actual mesh generation process for Ω_{bor}, we introduce the notion of an interior face:

Definition 3.8. *A face F of a boundary cell of a Cartesian grid $\mathcal{G}$ is called* **interior face** *if all of its corner points $P_i, i = 1, \ldots, 4$ fulfill $\varphi(P_i) \leq 0$, i.e., all corner points belong to the domain Ω.*

Once the minimizers are computed, Ω_{bor} is meshed in three steps:

Each interior face of a boundary cell forms together with the cell's minimizer a pyramid, which is split into two tetrahedra by dividing the base along the diagonal prescribed by the decomposition configuration of the interior cell sharing this face with the boundary cell.

After that we create the tetrahedra connected to interior edges. Each interior edge is shared by two or three boundary cells. The first case is illustrated

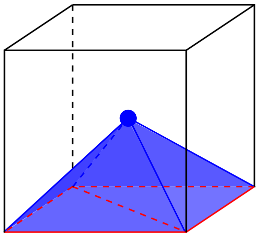


Figure 3.5. Uniform tetrahedralization: Decomposition of a pyramid formed by a boundary cell's interior face (red square) and minimizer (blue point) into two tetrahedra.

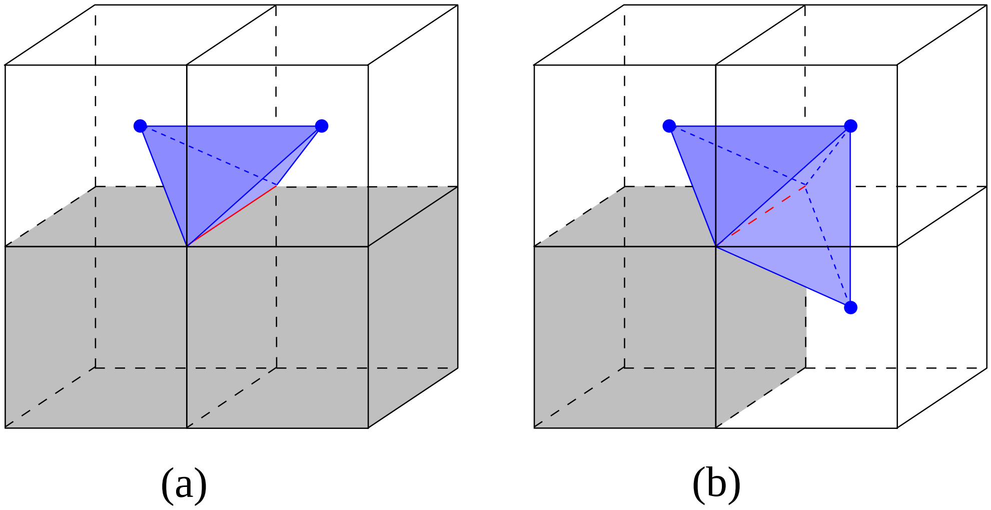


Figure 3.6. Uniform tetrahedralization: Possible cases of interior edges. Shaded cells are interior cells, the red edge is the interior edge.

by Figure 3.6(a). The endpoints of the edge and the minimizers of the two boundary cells form one tetrahedron. If the interior edge is shared by three boundary cells (cf. Figure 3.6(b)), two tetrahedra are created by the end points of the edge and respectively two minimizers of adjacent boundary cells.

Finally, the tetrahedra associated to interior points are added. Here, we have to distinguish three cases: An interior point can be part of one, two, or three sign change edges. Examples for these configurations are shown in Figure 3.7(a)(b)(c). In all three cases the interior point and the four minimizers of the boundary cells sharing a sign change edge form two tetrahedra. Therefore, we obtain two, four, or six tetrahedra per interior point. To avoid anisotropy, the quads formed by four minimizers are divided into two triangles in such

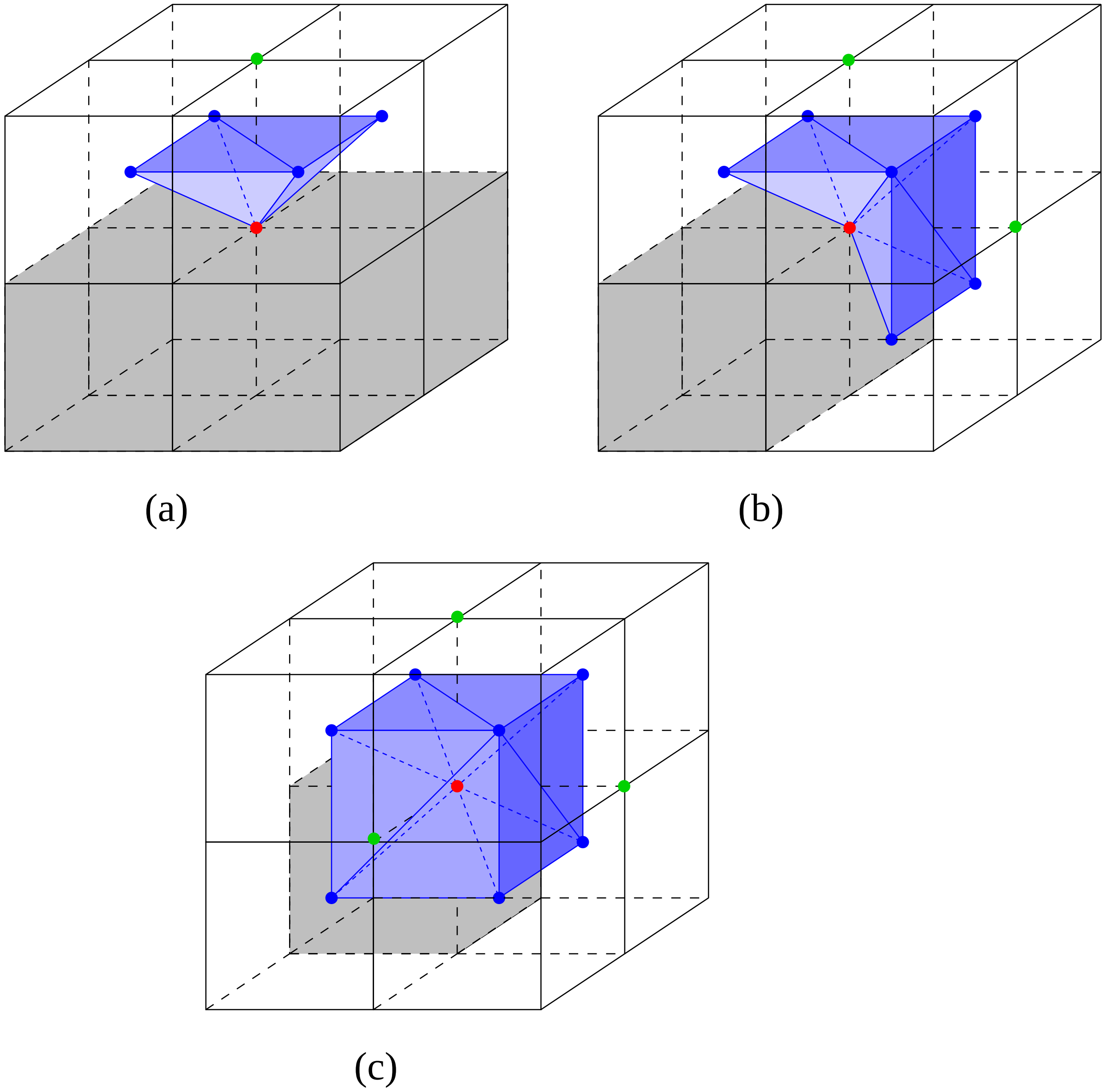


FIGURE 3.7. Uniform tetrahedralization: Possible cases of interior points. The red points mark the interior points, the blue ones are minimizers. The sign change edges connect the interior point and one of the green points. In (a), the interior point belongs to one sign change edge, in (b) to two, and in (c) to three sign change edges. Shaded cells are interior cells.

a way that of two adjacent minimizers one belongs to quad edges and the dividing diagonals, whereas the other one only belongs to quad edges.

3.2. Adaptive Mesh Generation

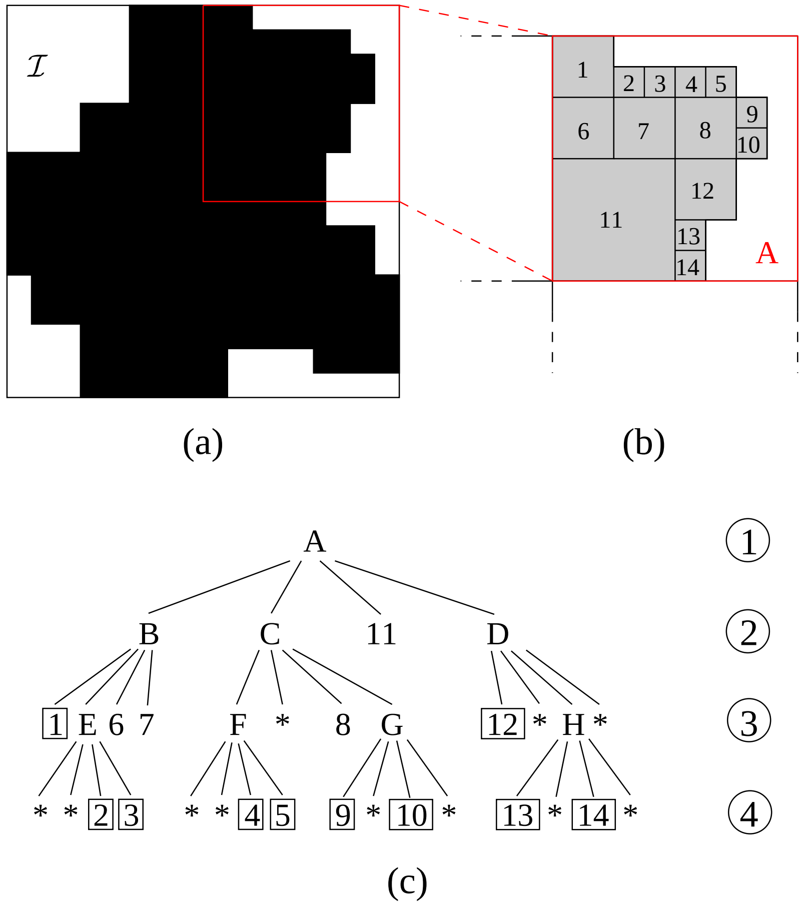
3.2.1. Adaptive 2D Triangulation. The mesh generation methods presented in the previous sections are robust in the sense that a mesh is created under any circumstances, and fast, since meshing of the largest part of the structure consists of assembling pre-computed voxel decompositions, which takes very little computing time. As we will show in section 3.5, the resulting meshes are of high quality. However, this simple approach is hardly applicable to larger geometries and high resolution examples. The resulting numbers of nodes and tetrahedra are very large, leading to long computing times. Thus, it is desirable to reduce the number of nodes and elements. This can be achieved by application of an adaptive mesh generation method.

As in uniform mesh generation, we divide the domain Ω into Ω_{int} and Ω_{bor}. Since our method of boundary condition mapping, which is presented in section 3.4, requires high resolution of the mesh on $\partial\Omega$, we do not allow for adaptivity on the structure's boundary and apply exactly the same mesh generation method for Ω_{bor} as in the uniform case. Thus, adaptivity in mesh generation concerns only Ω_{int}.

For mesh generation in the interior domain Ω_{int}, we use an approach developed by Schulz et al. [**48**]. Originally, this method was designed for generation of tetrahedral meshes from 3D voxel data, e.g., from 3D computer tomography images, and used for numerical homogenization of highly complex microstructures. However, the algorithm is applicable to macrostructures, too, and provides fast and robust adaptive tetrahedral mesh generation for general voxel geometries. In the following, we describe the method in two space dimensions. The corresponding 3D case is discussed in section 3.2.2.

Taking the procedure for uniform mesh generation described in section 3.1.1 as a starting point, the number of grid cells and thus the number of triangles can be reduced by combining neighbored cells to larger blocks, which then are divided into triangles. This is equivalent to constructing a quadtree representation of Ω_{int}.

The concept of a quadtree is well known. For a concise presentation see, e.g., [**45, 46, 31**]. For the sake of continuous reading, we sketch the method of building a standard quadtree: First, we set up a square image $\mathcal{I}$ of size $2^n \times 2^n$ grid cells (so-called picture elements or *pixels*), where n is the smallest natural number such that $\mathcal{I}$ contains Ω_{int} completely. We denote pixels belonging to Ω_{int} by black and those belonging to $\mathcal{I} \setminus \Omega_{int}$ by white. If $\mathcal{I}$ contains both black and white pixels, it is subdivided into four equal-sized quadrants, the *sons*, each of which is represented by a node in the quadtree representation.



(a) (b)

(c)

FIGURE 3.8. Example of a region with enclosing image $\mathcal{I}$ (a), image subdivision (b), and quadtree representation (c). Inner nodes are denoted by letters, leafs by numbers. Numbers in rectangles are boundary leafs. Hierarchy levels are given in circles.

The sons are labeled NW, NE, SW, and SE, according to their position. The subdivision into quadrants is recursively repeated until each son contains only pixels of one color, black or white.

An example for a quadtree subdivision is given in Figure 3.8, which shows a region with the enclosing image $\mathcal{I}$ (3.8(a)), the actual subdivision (3.8(b)), and the corresponding quadtree representation (3.8(c)). This representation looks like an upside down tree, which is the reason for naming this type of structures *tree*. Accordingly, the entire picture $\mathcal{I}$ is called *root* of the tree, whereas sons which are not subdivided further are called *leafs*. Nodes and leafs are grouped according to their resolution level. The root of the tree is assigned level 0, the finest possible leafs, i.e., single pixels, belong to level n. We number leafs with numbers, whereas interior nonleaf nodes are numbered with letters.

In [**64**], a standard quadtree representation is used for construction of adaptive triangular meshes. This allows for adjacent cells the hierarchy levels of which differ by more than one. However, the larger this difference is, the worse the quality of the triangles used in cell decomposition becomes. For example, if the hierarchy levels differ by two, the smallest occurring angle is 12.53 degrees. This makes postprocessing of the mesh necessary. To avoid this, Schulz et al. [**48**] restrict the difference in hierarchy levels to one.

As mentioned above, the results of our method of boundary condition application (cf. section 3.4) are the better the higher the resolution of the mesh on the structure's surface is. Therefore we require that all boundary cells of Ω_{int} must have highest possible resolution.

If this requirement is considered, the approach used in [**48**] leads to a modified quadtree representation an example of which is shown in Figure 3.9. We assume that Figure 3.9(a) represents part of a sufficiently large region which allows for the shown quadtree representation. In Figure 3.8(a), this assumption is not necessary, since the quadtree would be the same, anyway.

Figure 3.10 shows the configurations which are used to subdivide transition cells. Decompositions symmetric to the ones displayed are neglected. In Figure 3.11, the results of a complete adaptive triangulation of $\Omega = \Omega_{int} \cup \Omega_{bor}$ are presented. The underlying quadtree structure is highlighted by the green edges.

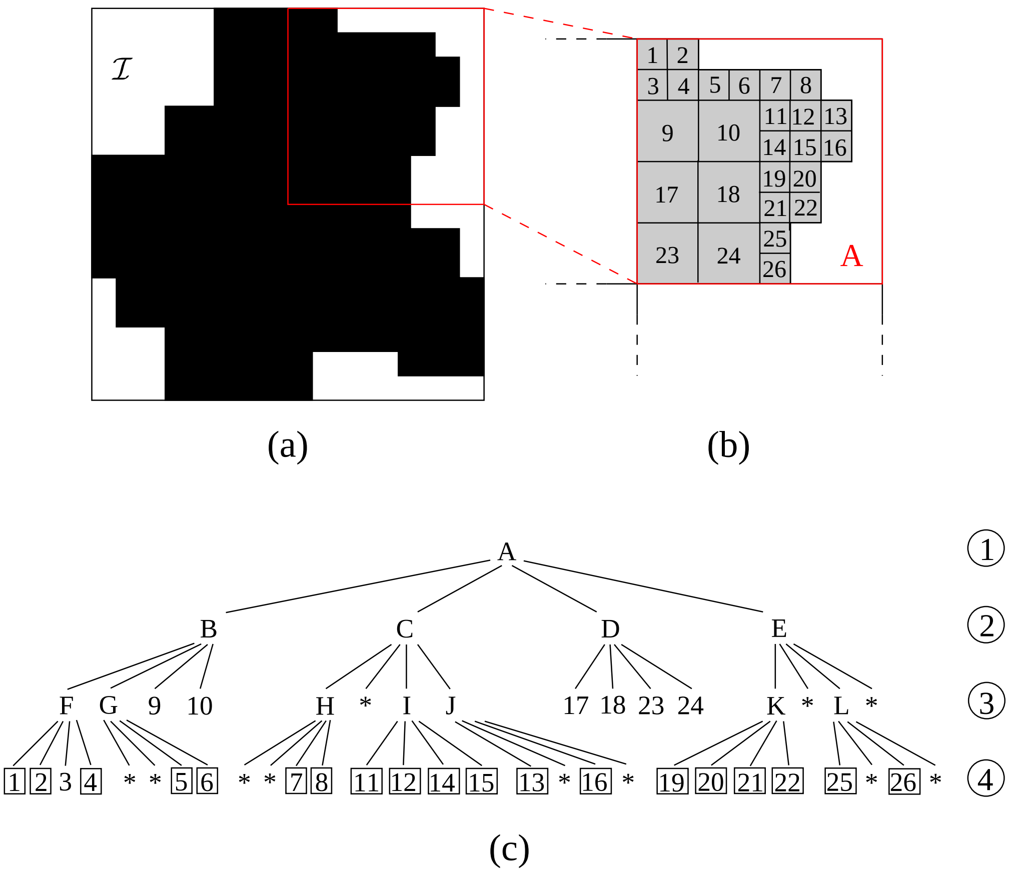
It is obvious that the number of leafs and hence the number of triangles created by [**48**] is larger than in [**64**]. However, we think that the higher mesh quality making extensive postprocessing superfluous easily compensates for this disadvantage.

Remark 3.9. Fixing the width h of the underlying grid $\mathcal{G}_h$ before Ω_{int} is meshed corresponds to truncation-based approximation of Ω (cf. [**45**]).

3.2.2. Adaptive 3D Tetrahedralization.

The basic ideas for adaptive mesh generation in Ω are the same as in the 2D case. High resolution is required on $\partial\Omega$, therefore adaptivity in mesh generation concerns only Ω_{int}, whereas Ω_{bor} is meshed as in the uniform case.

The grid cells belonging to Ω_{int} are grouped to blocks by using the three-dimensional equivalent to the quadtree concept, the octree. Here, the image $\mathcal{I}$ is a cube of size $2^n \times 2^n \times 2^n$, where n is the smallest number of grid cells



(a) (b)

(c)

FIGURE 3.9. Example of a region with enclosing image $\mathcal{I}$ (a), modified image subdivision (b), and the corresponding quadtree representation (c). Inner nodes are denoted by letters, leafs by numbers. Numbers in rectangles are boundary leafs. Hierarchy levels are given in circles.

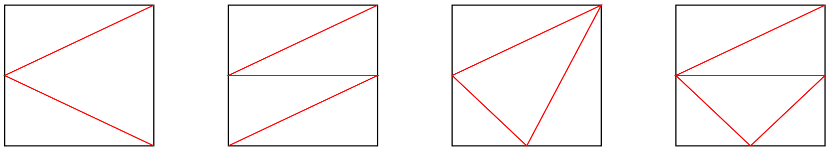


FIGURE 3.10. Configurations for decomposition of blocks having neighbors of lower hierarchy level.

needed to enclose the region to be subdivided completely. The grid cells are called volume elements or *voxels*. If $\mathcal{I}$ contains both black and white voxels, it is subdivided into eight equal-sized cubes, the octants. The decomposition process is repeated recursively until each block contains voxels of one color only.

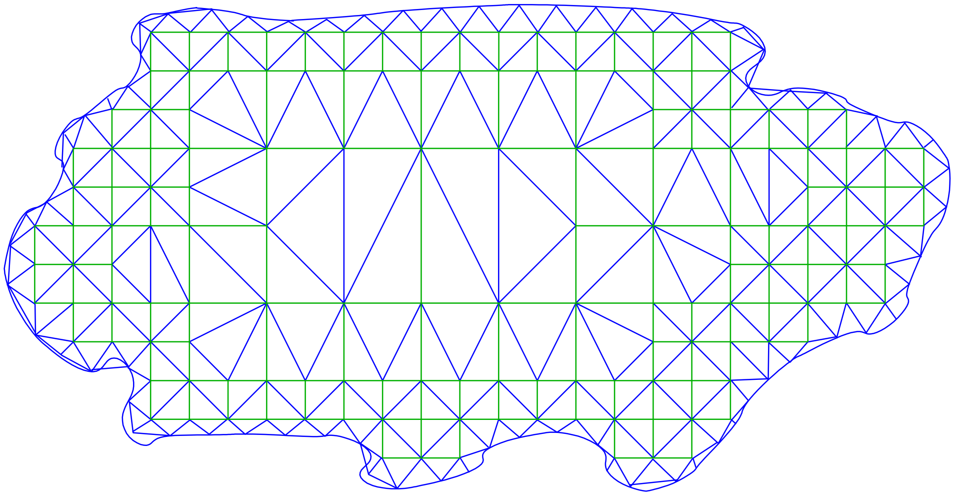


FIGURE 3.11. Adaptive triangulation. The green squares indicate the underlying quadtree structure.

Again, Zhang et al. [**64**] use a standard octree, allowing for neighbored cells the hierarchy levels of which differ by more than one. The faces of these transition elements are subdivided by the algorithm for adaptive triangulation in 2D which was presented in section 3.2.1. To be able to tetrahedralize the elements Steiner points are inserted in the center of the cells. However, the poor quality of the face triangulations leads to relatively low quality of the tetrahedra and enforces postprocessing for quality enhancement.

As in the 2D case, [**48**] do not allow for hierarchy levels differing by more than one for adjacent cells. Due to particular subdivision of elements, they generate consistent tetrahedralizations without insertion of additional points. If symmetric decompositions are included, there are 12,420 possible configurations. Despite this relatively high number, extremely fast mesh generation is possible if these configurations are pre-computed and stored in a look-up table. The quality of the resulting meshes is investigated in section 3.5.

Figure 3.12 shows a cross section of a three-dimensional structure meshed by the algorithm presented by Schulz et al. [**48**]. The adaptive coarsening in the structure's interior is clearly visible.

For the example shown in Figure 3.12, the mesh generated by the adaptive meshing algorithm consists of 39,347 nodes and 178,629 elements, whereas the uniform algorithm produces a mesh of 48,410 nodes and 221,139 elements. Of course, the surface mesh is identical in both cases. The number of nodes has been reduced by a factor of 1.23. For a finer grid width h this effect would be

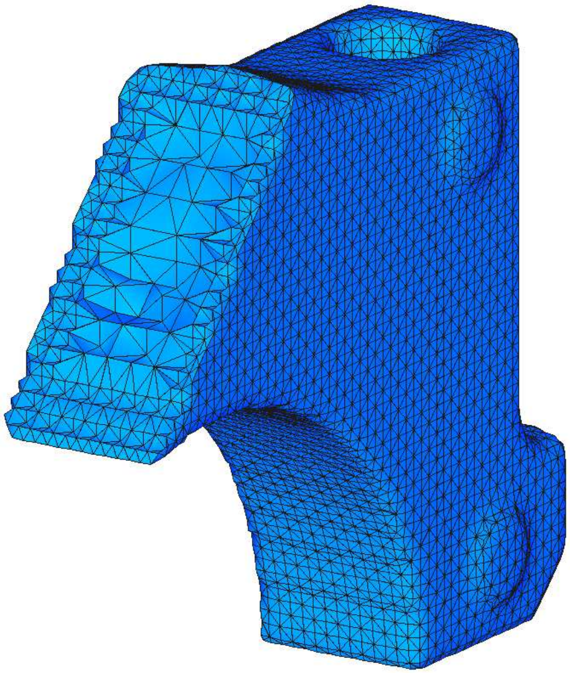


FIGURE 3.12. Cross section of an adaptive tetrahedral mesh.

even stronger, whereas for a larger value of h the ratio of node numbers would be worse.

REMARK 3.10. Naturally, for rather thin structures like shells or plates or in small features like walls, little or no effect of adaptivity can be observed and the resulting mesh is most likely the same as in uniform mesh generation. However, the simplicity and efficiency of the algorithm we use compensates for the high density and number of nodes and tetrahedra which might occur locally or, in extreme cases, in the entire structure.

3.3. Mesh Postprocessing and Improvement

In the previous sections, we described a way of generating uniform and adaptive triangulations and tetrahedralizations, respectively. The computation of minimizers of a quadratic error function e_q plays a crucial role in this approach. We mentioned

$$e_{q1}(x) = \sum_i (\mathbf{n}_i \cdot (x - p_i))^2,$$

which corresponds to the sum of squared distances of the vertex x to the lines or, in three dimensions, planes defined by intersection points of sign change

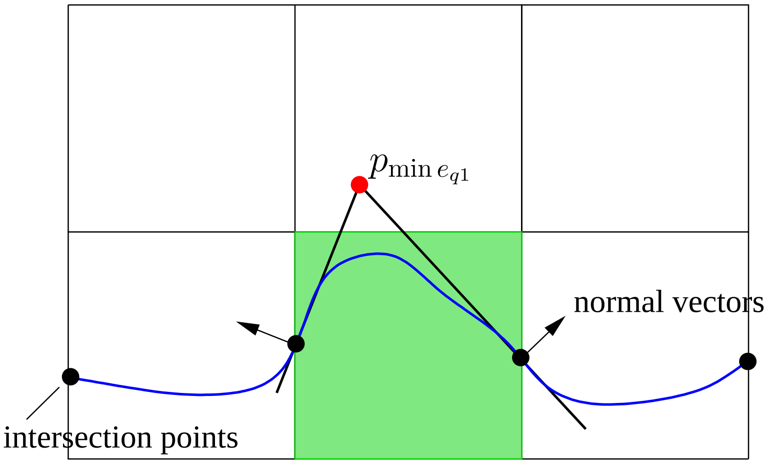


FIGURE 3.13. The red dot is the minimizer of e_{q1} belonging to the highlighted grid cell. Due to consideration of the normal vectors in the intersection points, the boundary feature in this cell is well preserved. However, the minimizer lies outside the cell.

edges and the structure's surface $\partial\Omega$, and as a simpler alternative

$$e_{q2}(x) = \sum_i (x - p_i)^2,$$

which has the arithmetic mean of the intersection points, $p_{min} = \frac{1}{n}\sum_{i=1}^n p_i$, as minimizer.

The use of e_{q1} has the advantage that sharp features and edges can be preserved, whereas e_{q2} smoothes $\partial\Omega$. However, when computing the minimizers p_{min} of e_{q1}, one has to keep in mind that these points can lie outside the cell (cf. Figure 3.13). If $p_{min} \notin \Omega_{int}$, this is not a problem. Otherwise, the described method creates intersecting tetrahedra. One possible way to avoid this is to restrict the set of admissible points, i.e. to determine the minimizer of a boundary cell C as solution of the optimization problem

$$p_{min} = \min_{x \in C} e_{q1}(x).$$

If e_{q2} is used, it is not necessary to restrict the set of admissible points for minimizer determination.

Both minimization of e_{q1} and e_{q2} can lead to triangles and tetrahedra of arbitrarily low quality: The closer p_{min} to an interior edge is, the lower the elements' quality becomes. For the 2D case, this is illustrated in Figure 3.14.

Zhang et al. [**64**] solve this problem by postprocessing the mesh. They employ three quality measures based on which the mesh is corrected. Details on these

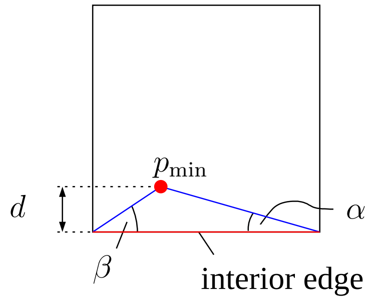


FIGURE 3.14. The smaller the distance d between the minimizer p_{min} and the interior edge is, the more acute the angles α and β between interior edge and the sides of the triangle become.

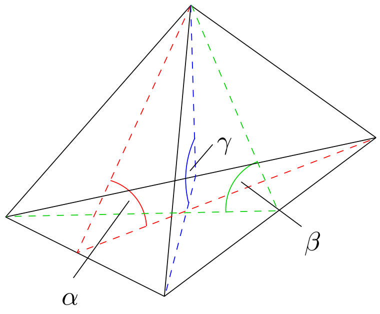


FIGURE 3.15. Definition of dihedral angles for a tetrahedron.

measures can be found in section 3.5. The actual correction is done in two steps, which are applied iteratively until all elements fulfill the prescribed criteria:

- Correction of orientation: If a triangle or tetrahedron does not comply with the right-hand principle, i.e., it has negative volume, two of its nodes are interchanged.
- Sliver removal by edge contraction: If a triangle or tetrahedron has in node A a dihedral angle $\alpha \leq 10°$ or $\alpha \geq 160°$, the shortest of the three edges AB, AC, and AD is deleted by merging of the edge's endpoints (cf. Figure 3.15).

We follow another approach and avoid creation of elements with too low quality completely. Since the problem arises from minimizers placed at unfavorable positions, minimizer computation is the point to be modified. As shown in Figure 3.16, we restrict the domain in which the minimizer can be computed to a sphere S_r of radius $r < \frac{h}{2}$ which has the cell center as midpoint, i.e.,

$$p_{min} = \min_{x \in S_r} e_{q2}(x). \tag{14}$$

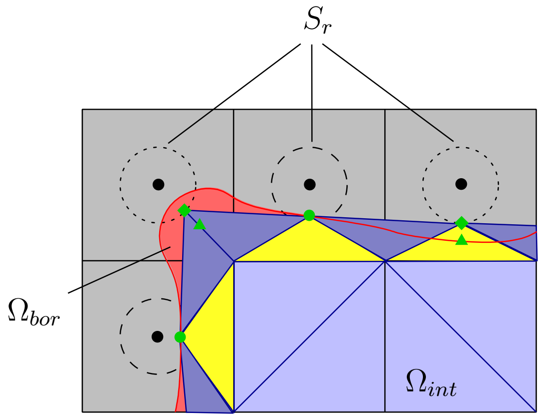


FIGURE 3.16. Restricted domain for minimizer computation in boundary cells. Green triangles indicate the unconstrained minimizer positions, squares show the constrained position. For green circles, constrained and unconstrained minimizer coincide. Active restricted domains are indicated by dotted circles.

If the arithmetic mean of the intersection points of a boundary cell, which we denote by p_{avg} to distinguish it from the solution of (14), lies outside S_r, p_{min} is computed as the orthogonal projection $\mathcal{P}$ of p_{avg} onto S_r :

$$p_{min} = \mathcal{P} p_{avg}.$$

Clearly, the quality of the tetrahedra resulting from the mesh depends on the choice of the radius r of the sphere - the larger r is, the lower the elements' quality can be. The proper choice of r will be discussed in section 3.5.

For further improvement of mesh quality, we apply an iterative smoothing process to the mesh. This was inspired by the work of Persson [**42**], who exploits the analogy of a mesh and a truss structure to construct meshes from implicit geometry descriptions. By considering element edges as springs, Persson assigns a force to each edge in the mesh. This force depends on the actual length l and the unextended length l_0 of the spring. l_0 is controlled by a mesh size function which is defined in the entire domain and depends on the shortest distance of a point to the structure's boundary. However, only repulsive forces are allowed since the initial mesh, which is based on a Delaunay triangulation of a random point cloud, has to be pushed towards the structure's boundary. After that, a nodal force vector $\mathbf{F}$ is computed by summing up all forces associated to edges which contain a given node. Clearly, the nodal force vector depends on the nodal positions, i.e., $\mathbf{F} = \mathbf{F}(\mathbf{p})$. The resulting system of nodal forces is solved for an equilibrium position of the nodes:

$$\mathbf{F}(\mathbf{p}) = \mathbf{0}. \tag{15}$$

Since this minimization problem is rather complicated, [**42**] introduces an artificial time dependence and transforms (15) into the following system of ODEs:

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}(\mathbf{p}), t \geq 0. \tag{16}$$

A stationary solution of (16) fulfills condition (15). It is approximated using the forward Euler method. Thus, a series of nodal positions is computed by

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \Delta t \mathbf{F}(\mathbf{p}_n), \tag{17}$$

where $\mathbf{p}_i$ is the vector of node coordinates at the i-th time step. When equilibrium is reached, the resulting mesh is generally of very high quality.

In contrast to [**42**], we do not start with an implicit description of the structure and a randomly distributed point cloud but with an initial mesh the surface of which fits the structure well. Therefore, we modify the approach used by Persson as follows:

By keeping the positions of nodes belonging to the surface mesh fixed we can allow for both attractive and repulsive nodal forces. In addition, we need not consider exterior forces which prevent nodes from crossing the structure's boundary. Since our initial mesh is already adaptive in element size, the construction of an underlying mesh size function is not necessary.

There is a significant difference in the actual computation of a stationary solution of (16). In contrast to [**42**], we do not set up global vectors $\mathbf{p}$ and $\mathbf{F}$ which contain the data of all nodes and allow for simultaneous correction of all nodal positions, but use a local approach.

For the description of this approach we use the set of *one-ring neighbors*:

Definition 3.11. *For a node P_i of a mesh, the set $\mathcal{N}_1(P_i)$ of* **one-ring neighbors** *is defined as the set of all nodes P_j which are connected to P_i by an edge E_{ij}.*
$n_{\mathcal{N}_1} = n_{\mathcal{N}_1(P_i)} = |\mathcal{N}_1(P_i)|$ *denotes the cardinality of $\mathcal{N}_1(P_i)$, i.e., the number of one-ring neighbors of P_i.*

Applying the non-physical assumption that all springs (edges) in the mesh have unextended length $l_0 = 0$ and using a simple linear model for the spring force, we compute the nodal force in a node P_i as

$$\frac{1}{|\mathcal{N}_1(P_i)|}\sum_{j=1}^{|\mathcal{N}_1(P_i)|}(P_j - P_i).$$

Setting the time step $\Delta t = 1$, we obtain the updated position $P_{i_{n+1}}$ of P_{i_n} by using

$$\begin{aligned} P_{i_{n+1}} &= P_{i_n} + \frac{1}{|\mathcal{N}_1(P_{i_n})|}\sum_{j=1}^{|\mathcal{N}_1(P_{i_n})|}(P_{j_n} - P_{i_n}) \\ &= \frac{1}{|\mathcal{N}_1(P_{i_n})|}\sum_{j=1}^{|\mathcal{N}_1(P_{i_n})|}(P_{j_n}). \end{aligned}$$

This means that the solution of (15) is approximated by sequential computation of nodal position updates.

Experiments have shown that it is not necessary to iterate the smoothing process until convergence is reached. Sufficient element quality is generally reached after five to seven iterations. The overall quality of the resulting meshes is investigated in section 3.5.

To summarize our mesh generation algorithm, we give a concise presentation in pseudocode:

```
START
  define the underlying Cartesian grid $\mathcal{G}_h$
  determine $\Omega_{int}$ and $\Omega_{bor}$ by detection of interior and
      boundary cells
  build a modified octree representation of $\Omega_{int}$
  mesh $\Omega_{int}$ using pre-computed decompositions of blocks
  compute a minimizer for each boundary cell:
      $p_{min} = \min_{x \in S_r} e_{q2}(x)$
  mesh $\Omega_{bor}$:  construct tetrahedra assigned to interior
      faces, edges, and points
  WHILE (mesh quality not high enough)
    smooth the mesh using equation (18)
  END WHILE
END
```

3.4. Boundary Condition Mapping

For practical reasons, specification of the boundary conditions which are applied to the structure is connected to boundary patches described by surface triangulations. Clearly, these triangulations will in general not coincide with the surface mesh resulting from the mesh generation process described earlier in this chapter. In order to make meshing and structural analysis fast, manual interaction has to be avoided wherever possible. Therefore, boundary conditions have to be transferred automatically to the mesh. This is of special importance if the iterative nature of the topology optimization algorithm presented in chapter 4 is considered: If boundary conditions had to be manually assigned to the updated tetrahedral mesh, our algorithm would be practically useless since computations could not be performed in acceptable time and with justifiable effort.

As shown above, we do not apply adaptive coarsening on the boundary of the structure, i.e., the entire surface mesh is of the highest resolution. The maximal possible edge length is $h + 2r$, where r is the radius of the sphere in which a boundary cell's minimizer of the quadratic error function e_q is computed. Together with the fact that surface patches which are subject to boundary conditions are given as triangulated surfaces, this allows for the following procedure of boundary condition mapping, which we give in pseudocode before we explain the details:

```
START
  initialize boundary condition indices with 0
  FOR all faces $c_i$ of the surface mesh
    FOR all faces $c_j^{BC}$ of the boundary condition surface
      IF ((bounding box($c_i$) overlaps with $c_j^{BC}$) AND
        ($\cos\angle(\mathbf{n}_{c_i}, \mathbf{n}_{c_j^{BC}}) \leq 1.0-$ tolerance))
        assign boundary condition index of $c_j^{BC}$ to $c_i$
        break
      END IF
    END FOR
  END FOR
END
```

As shown in the pseudocode description of the boundary condition surface mapping, we compare each face of the surface mesh with the faces of the boundary condition patch. If the bounding box of a face c_i of the surface mesh overlaps with a face c_j^{BC} of the boundary patch and the angle between the normals $\mathbf{n}_{c_i}$ and $\mathbf{n}_{c_j^{BC}}$ of both faces is small enough, we assign the index of the boundary condition surface to the face of the surface mesh. If there are several boundary condition patches, the procedure is repeated until all patches have been treated. However, in the second and later iterations, only faces which have not been assigned a boundary condition index need to be considered.

For large meshes of high resolution and finely discretized boundary condition patches, the number of overlap tests and normal comparisons can become very large. To keep computing times low, we employ a highly efficient algorithm for tests of triangle-box overlap, which was developed by Akenine-Möller [**1**]. This test is based on the *separating axis theorem* by Gottschalk et al. [**27**], but uses axis aligned bounding boxes instead of oriented bounding boxes. The maximal number of axis tests is reduced to 13, compared to 15 in [**27**].

To achieve a higher accuracy, we supplement the triangle-box overlap test by computation of the portion of the triangle's area which intersects with the box. Only if the overlapping area is more than one half of the total area of the triangle, the respective boundary condition index is assigned.

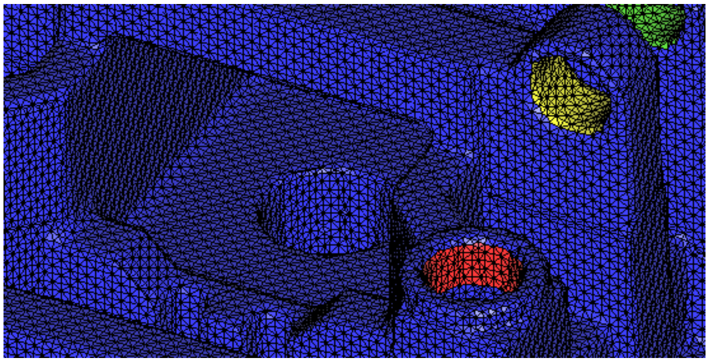


FIGURE 3.17. Mapping of boundary condition patches (red, yellow, and green). The normal of triangles subject to boundary conditions differs significantly from the normals of neighbored triangles which do not belong to a boundary condition patch.

For the classes of problems we consider, completion of the triangle-box overlap test by comparison of normals is proximate. Usually, boundary condition patches are clearly delimited, and often adjacent surface patches have significantly different normals, as illustrated for internal surfaces of bores in Figure 3.17. However, the normal comparison criterion is not completely reliable.

Therefore, we allow for manual correction of boundary condition patches: Single surface element faces can be assigned or unassigned an arbitrary boundary condition index.

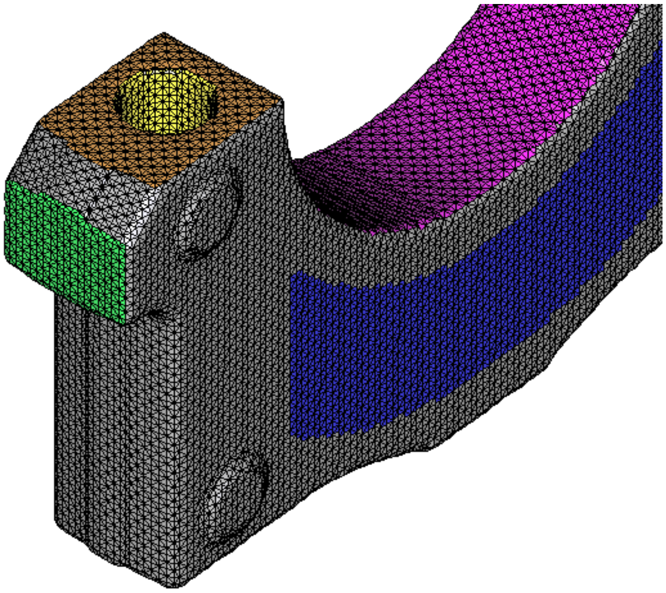


FIGURE 3.18. Mapping of boundary condition patches.

For the example shown in Figure 3.18, all boundary condition patches were properly detected, and no manual correction was necessary.

REMARK 3.12. Inclusion of information about boundary condition surfaces in the mesh generation process would allow for adaptive coarsening on the structure's surface, too. The boundaries of boundary condition patches could be resolved well, whereas the interior of such patches and areas not subject to boundary conditions could be meshed more coarsely. However, this would require a modification of the meshing algorithm. In adaptively meshed surface areas, the width of the boundary layer should be much larger to avoid elements of very low quality.

3.5. Mesh Quality

In finite element computations, the quality of the mesh is crucial both for accuracy of the solution and speed of iterative solvers. In [**53**], Shewchuk shows that interpolation accuracy as well as the condition number of the global stiffness matrix depend on the quality of the mesh.

Concerning interpolation, two types of errors have to be considered: Let $v(p)$ be a continuous scalar function on a mesh $\mathcal{T}$ consisting of elements t and let $w(p)$ denote a piecewise linear approximation to $v(p)$. The errors under consideration are

$$\parallel v - w \parallel_\infty = \max_{p \in \mathcal{T}} \mid v(p) - w(p) \mid$$

and

$$\parallel \bigtriangledown v - \bigtriangledown w \parallel_\infty = \max_{p \in \mathcal{T}} \mid \bigtriangledown v(p) - \bigtriangledown w(p) \mid,$$

i.e., the largest pointwise interpolation error and the largest error in the interpolated gradient, respectively. In structural analyses and simulations of mechanical deformation, the strains $\bigtriangledown v$ are usually of greater interest than the displacements v.

Shewchuk [**53**] gives bounds for both errors:

$$\parallel v - w \parallel_\infty \leq c_t r_{mc}^2 \leq c_t \frac{3}{8} l_{\max}^2$$

and

$$\begin{aligned} \parallel \bigtriangledown v - \bigtriangledown w \parallel_\infty &\leq c_t \frac{\frac{1}{3V} \sum_{1 \leq i < j \leq 4} A_i A_j l_{ij}^2 + 2 \max_i \sum_{j \neq i} A_j l_{ij}}{\sum_{m=1}^4 A_m} \\ &\leq c_t \frac{\sum_{1 \leq i < j \leq 4} A_i A_j l_{ij}^2}{V \sum_{m=1}^4 A_m}, \end{aligned}$$

respectively. Here, r_{mc} is the radius of the *min-containment sphere*, i.e., the smallest sphere which encloses the element completely, c_t is a constant which may vary from element to element, V denotes the element volume, l_{ij} is the length of the edge connecting nodes i and j, and A_i is the area of the tetrahedron's i-th face. The weaker upper bounds are easier to compute. They show that $\parallel v - w \parallel_\infty$ is proportional to the square of the length of the longest edge, but does not depend on the element's shape, whereas both the volume V and the areas A_i occur in the bound of $\parallel \bigtriangledown v - \bigtriangledown w \parallel_\infty$. This explains why *slivers*, i.e., tetrahedra which are nearly flat but do not have an edge which is significantly shorter than the others, should be avoided in mesh generation: Since V is very small, the error of the interpolated gradient can become very large.

Solution time for the system of equations arising in finite element computations usually grows with the condition number $\kappa = \lambda_{\max}^K / \lambda_{\min}^K$, where $\lambda_{\max}^K$ and $\lambda_{\min}^K$ denote the largest and smallest eigenvalue of the stiffness matrix K. In [**24**], Fried gives a lower bound for $\lambda_{\min}^K$ which is proportional to the volume of the smallest element in $\mathcal{T}$. $\lambda_{\max}^K$ is proportional to the largest eigenvalue $\lambda_{\max}^t$ of an element stiffness matrix. If m is the largest number of elements sharing a single node,

$$\max_{t\in\mathcal{T}} \lambda^t_{\max} \leq \lambda^K_{\max} \leq m \max_{t\in\mathcal{T}} \lambda^t_{\max}.$$

For tetrahedral meshes, $\lambda^t_{\max}$ and $\lambda^K_{\max}$ are not scale invariant, i.e., if t is scaled uniformly without changing the shape, $\lambda^t_{\max}$ is not constant. In fact, it grows with the length of the longest edge $l_{\max}$. Therefore, the largest element in a mesh might determine the largest eigenvalue of the global stiffness matrix, and it is important that large elements are well-shaped. This indicates that adaptive mesh coarsening, which reduces the number of nodes and elements, should be applied cautiously and that the number of levels in the hierarchical octree describing the decomposition of Ω_{int} into cells should be limited if the quality of large elements is too low.

3.5.1. Mesh Quality Measures.

There exist a wide variety of quality measures for tetrahedral meshes. To evaluate the quality of the meshes generated by the method above, we adopt the concept of a *fair measure*, which is introduced in, e.g., [**19**].

Definition 3.13. *A function* $m_f : \mathcal{T} \to \mathbb{R}$, $t \in \mathcal{T} \mapsto m_f(t)$, *which assigns a real number to each element of a triangulation is called a* **fair measure** *if it fulfills the following conditions:*

- $m_f(t_{deg}) = 0$ *for all degenerate elements* $t_{deg} \in \mathcal{T}$, *i.e., for all elements of volume 0,*
- $m_f t_i = m_f t_j$ *for similar elements* $t_i, t_j \in \mathcal{T}$. *Elements are called similar if one can be transformed into the other one by an affine map, i.e., rotation, scaling, and translation,*
- m_f *is bounded, i.e.,* $m_f(t) < \infty$ *for all elements* t, *and*
- m_f *is normalized, i.e.,* $0 \leq m_f(t) \leq 1$ *for all elements* t.

The last item in Definition 3.13 indicates that fair measures are easier to compare than other measures.

In particular, we use the following quality measures, which are described in, e.g., [**71, 40, 23**]. Where necessary, we inverted the measure to obtain fair measures.

- the normalized shape ratio $q_{nsr} = \frac{3r}{R}$,
- the aspect ratio $q_\mu = 2\sqrt{6}\frac{r}{L}$,
- $q_\eta = 12\frac{(3V)^{2/3}}{\sum_{i=1}^{6} l_i^2}$, and
- the edge ratio $q_{edgeRatio} = l/L$.

Here, r denotes the radius of the insphere of a tetrahedron, R denotes the circumsphere, V is the tetrahedron's volume, l_i is the length of the i-th edge, and l and L denote the shortest and longest edge of the tetrahedron, respectively. In addition to the popular quality measures shape ratio, aspect ratio, and edge ratio, we consider q_η which takes the volume V into account and therefore allows for detection of slivers, which have acceptable edge lengths but almost zero volume.

The quality measures used for mesh optimization in [**64**] are $q_V = \frac{V}{\widehat{V}}$, where $\widehat{V}$ is the volume of the equilateral tetrahedron with same radius R of the circumsphere as the tetrahedron under consideration, the minimal and maximal angle of the tetrahedron, and the right hand side principle which checks the orientation of a tetrahedron's nodes and interchanges two arbitrary nodes if the volume V is negative.

Alternative measures which are used in various finite element codes are for example the uniformity measure

$$\frac{\min_{t \in \mathcal{T}} V}{\max_{t \in \mathcal{T}} V},$$

which assigns a single value to an entire mesh, the minimal and maximal dihedral angle, the minimum Jacobian

$$((P_0 - P_2) \times (P_1 - P_0)) \cdot (P_3 - P_0),$$

where $P_0, \ldots, P_3$ are the corners of the tetrahedron, and

$$\frac{1}{6^4\sqrt{2}} \frac{(\sum_{i=1}^{6} l_i)^3}{V} + \sum_{i=1}^{6} (\frac{l_i}{h} + \frac{h}{l_i} - 2)$$

with h the desired edge length.

The latter is used in [**47**]. Its effect is twofold: The first term prevents the creation of flat elements with relatively long edges and small volume, whereas the second term penalizes elements with too large or too small volume. However, the quality measure is not applicable in our case since we want to use the adaptive coarsening to some extent.

3.5.2. Mesh Quality Evaluation. In the following, we analyze the quality of the meshes by showing histograms of the four quality measures. Each step of the mesh generation process is evaluated.

The quality of the mesh of Ω_{int} (uniform case) is shown in Figure 3.19. The histograms display the accumulated distribution of element quality values. All

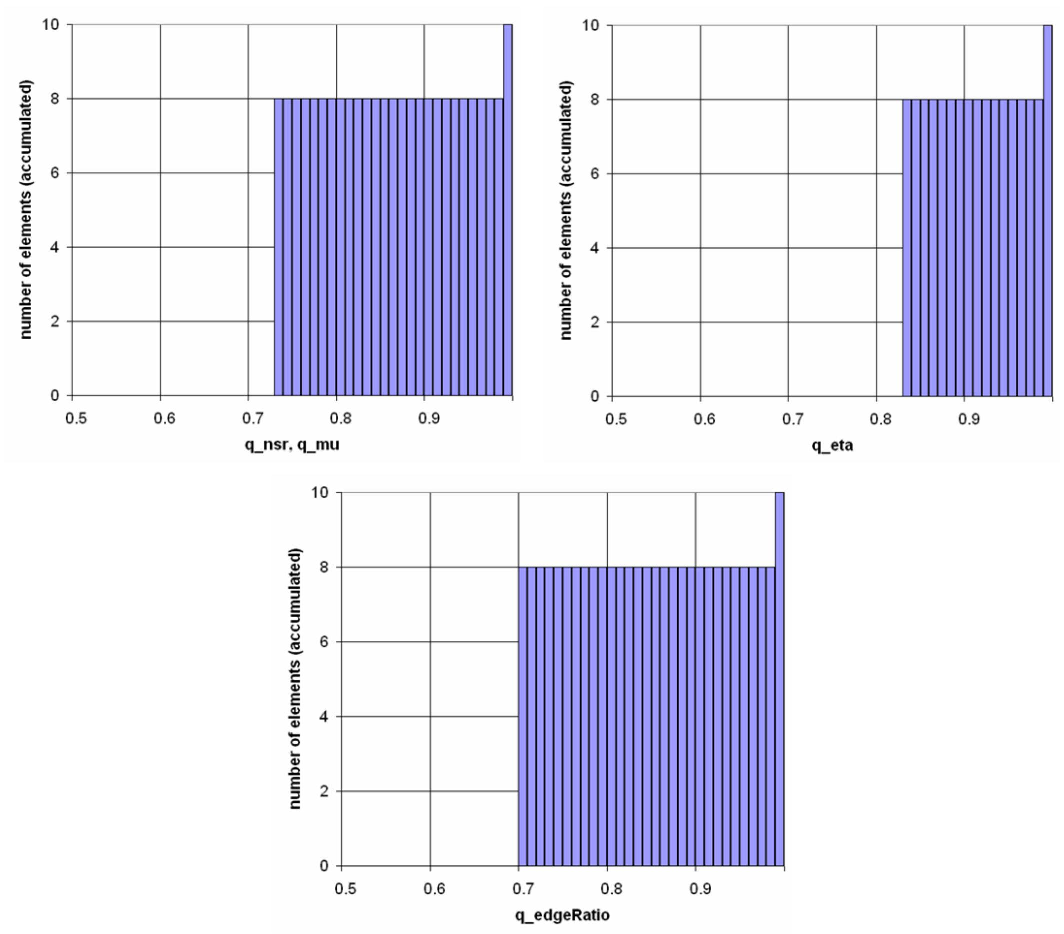


FIGURE 3.19. Accumulated element quality distribution for the mesh of Ω_{int} (uniform case). All elements have quality values higher than 0.7. No smoothing has been applied to the mesh.

ten tetrahedra used in the decomposition of voxels have quality values higher than 0.7 which is very good. The value distributions for q_{nsr} and q_{μ} are identical.

To evaluate the quality of the tetrahedra potentially originating from adaptive decomposition of blocks in Ω_{int}, we constructed a mesh containing all possible configurations. A small part of this mesh is shown in Figure 3.20. The quality values for this mesh are shown in Figure 3.21. Again, we display accumulated data. Compared to the non-adaptive case, there are elements of significantly lower quality. This is natural, since the tetrahedra created from transition cells deviate much more from the ideal equilateral tetrahedron than the ones used in ordinary voxel decomposition. However, no element has a quality value

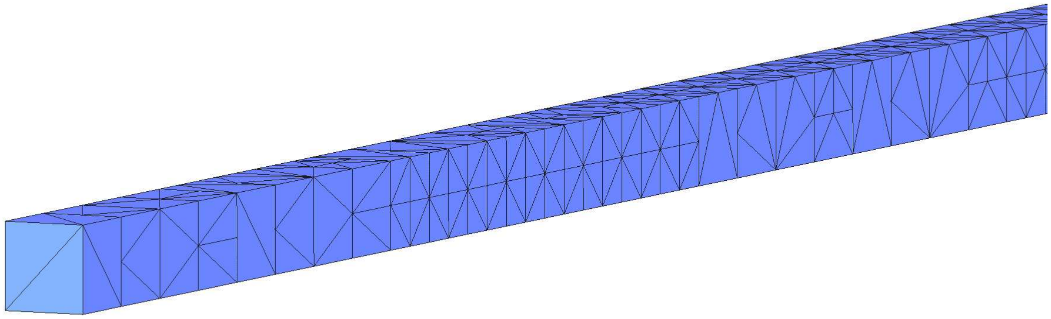


FIGURE 3.20. Part of a mesh containing all possible configurations of voxel decomposition into tetrahedra in Ω_{int} (adaptive case). No smoothing has been applied to the mesh.

lower than 0.3, which is often seen as a bound for admissible element quality. That means that the elements created during mesh generation in Ω_{int} are of good quality.

The quality of the mesh describing the border layer Ω_{bor} depends on the chosen width of the margin. As shown in Figure 3.22, a margin of 28 % of the grid width h has to be chosen to guarantee an element quality not smaller than 0.3. However, smoothing of the mesh allows for a significantly smaller value and thereby for higher precision in domain representation. Figure 3.23 shows a cross section through a typical smoothed adaptive mesh. The quality histogram for this mesh is given in Figure 3.24. The margin was set to 10 % of the grid width, and three smoothing iterations were performed. The mesh consists of 191,456 nodes and 886,280 tetrahedra. It contains only 17 slivers with $q_\eta < 0.3$.

Sliver tetrahedra are a common problem in tetrahedral mesh generation, and all Delaunay mesh generators have to deal with it. As pointed out above, they should be avoided because their Jacobian is close to singular and interpolation

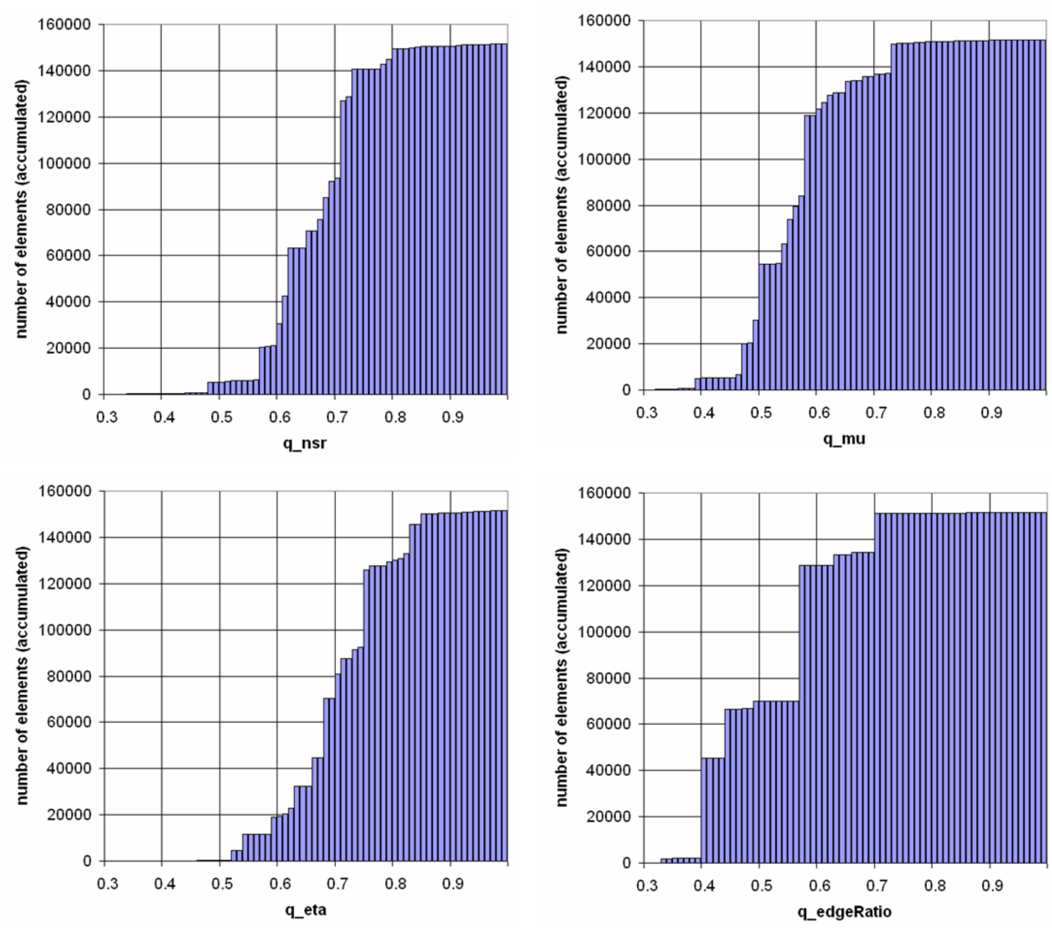


Figure 3.21. Accumulated element quality distribution for the mesh of Ω_{int} (adaptive case). Element qualities are significantly lower than in the uniform case. However, all elements have quality values higher than 0.3. No smoothing has been applied to the mesh.

of derivatives on such elements becomes inaccurate. There are various techniques which have been developed for sliver removal, for example edge flipping and face swapping combined with Laplacian smoothing [**22**]. However, the influence of single bad elements on the solution of finite element computations is not clear [**53**]. For the time being we neglect the few slivers occurring in our meshes and do not apply sliver removal methods. Future work might investigate the effect of the elements having lower quality and possibly enhance the algorithm by sliver removal if necessary.

Remark 3.14. As mentioned above, large elements might have a strong influence on the condition number of the stiffness matrix K and thereby on the solution time of the structural analysis. However, since large elements occur

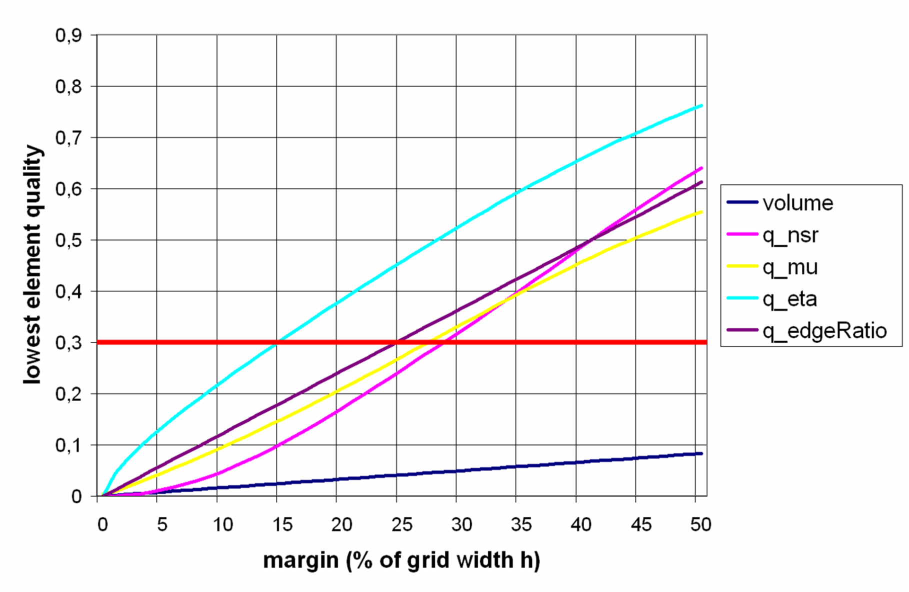


FIGURE 3.22. Correlation of minimizer position restriction and lowest occurring element quality.

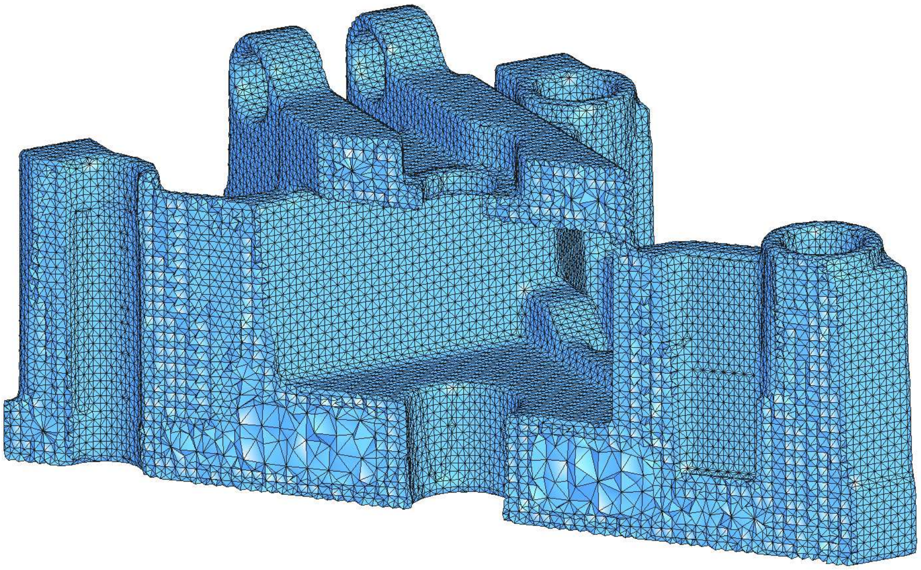


FIGURE 3.23. Cross section through a typical smoothed adaptive mesh consisting of 191,456 nodes and 886,280 tetrahedra. Only 17 elements have a quality $q_\eta < 0.3$.

only in the interior of the structure and are of good quality, it is not necessary to evaluate a uniformity measure and restrict the number of hierarchy levels of the octree structure used in adaptive mesh generation.

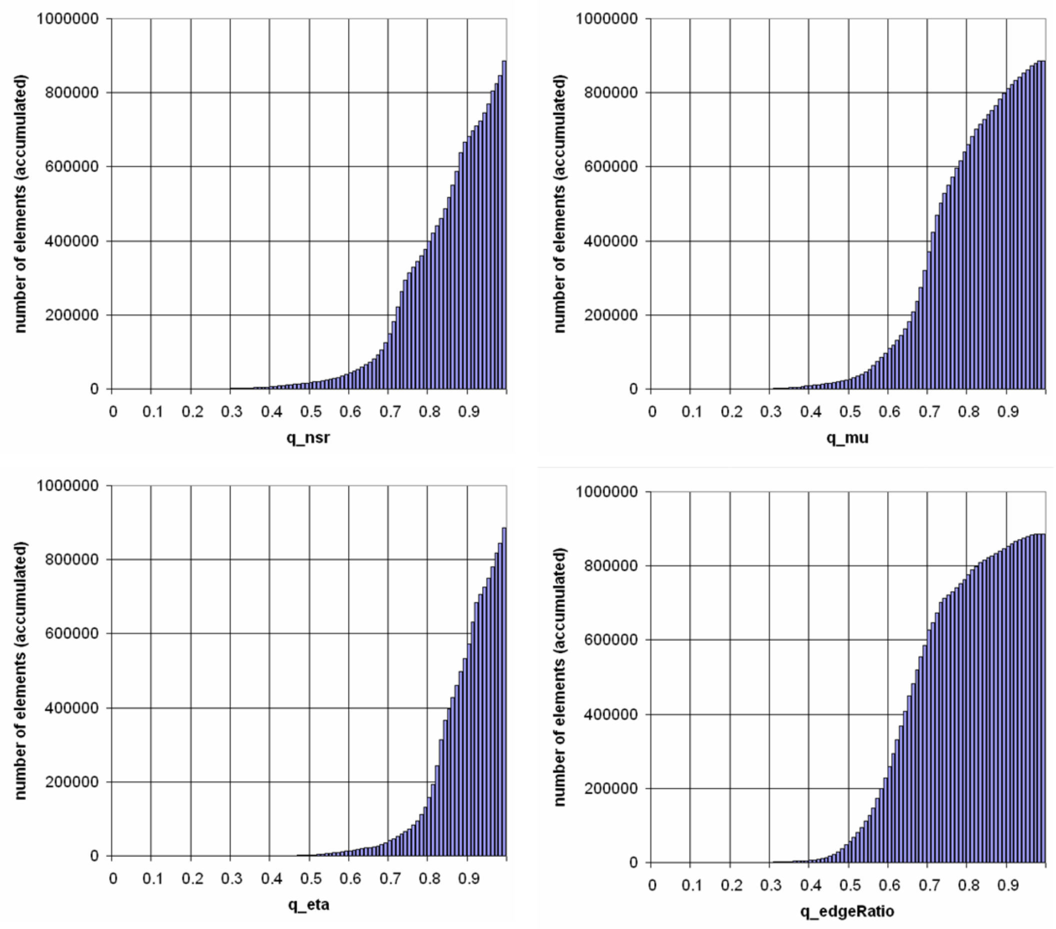


FIGURE 3.24. Typical accumulated element quality distribution for the complete mesh of Ω (adaptive case). The margin for minimizer computation was set to 10%, and three smoothing iterations have been performed.

REMARK 3.15. Inclusion of information about boundary condition surfaces in the mesh generation process would allow for adaptive coarsening on the structure's surface, too. The boundaries of boundary condition patches could be resolved well, whereas the interior of such patches and areas not subject to boundary conditions could be meshed more coarsely. However, this would require a modification of the meshing algorithm since the width of the boundary layer should be much larger in adaptively meshed surface areas in order to avoid large elements of very low quality.

CHAPTER 4

Numerical Algorithm

This chapter contains a detailed description of the numerical algorithm we use for solving the topology optimization problem. A very basic sketch of the algorithm's framework is shown in Figure 4.1. Below, we follow this outline and split the algorithm in its components. In section 4.1, we discuss the choice of the underlying optimization algorithm in general terms. Sections 4.2 and 4.3 describe the topological sensitivity and the structural update, respectively. The latter concerns both the level set function φ_i describing the domain Ω_i in the i-th optimization iteration and the corresponding finite element mesh which is used for the structural analysis. Special weight is put on the simplicity and efficiency of our method. Section 4.4 summarizes and illustrates the complete process chain.

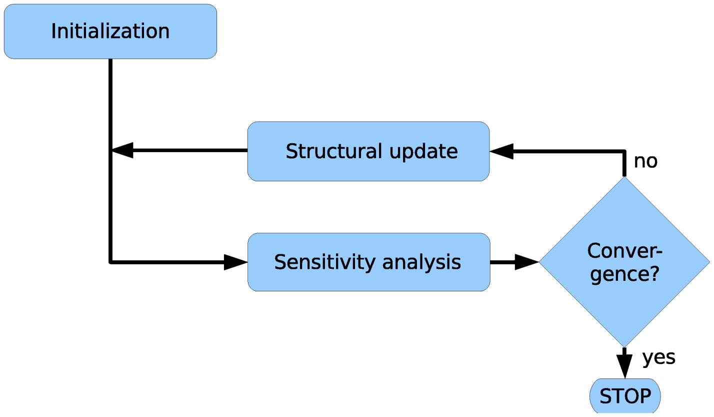


FIGURE 4.1. General structure of topology optimization algorithms. After the initial structure is provided (initialization), sensitivity analysis and structural update are iterated until the algorithm converges.

As stated in chapter 1, the discrete topology optimization problem is

$$\begin{aligned} &\min_{\varphi_\vartheta \in \mathcal{U}_{ad,\vartheta}} \quad J(\varphi_\vartheta, u_h) = \int\limits_{\Omega_{\varphi_\vartheta}} [\mathbb{C} : \varepsilon_h(u_h)] : \varepsilon_h(u_h) dx \\ &\text{subject to} \quad \int\limits_{\Omega_{\varphi_\vartheta}} dx \leq V_{target} \quad \text{and} \\ &\qquad\qquad\quad a_{\Omega_{\varphi_\vartheta}}(u_h, v_h) = l_{\Omega_{\varphi_\vartheta}}(v_h). \end{aligned} \tag{18}$$

φ_ϑ is the design variable, and the set of admissible solutions is given by

$$\begin{aligned} \mathcal{U}^R_{ad,\vartheta} = \{ \varphi_\vartheta \mid \quad & \Omega_{\varphi_\vartheta} \subset D, (\Gamma_{Nth} \cup \Gamma_{Dh}) \subset \partial\Omega_{\varphi_\vartheta}, \\ & \exists \phi_\vartheta \in C^0(D, \mathbb{R}) : \varphi_\vartheta = \widetilde{k}_\vartheta \star (\phi_\vartheta * \widetilde{k}_\vartheta) \Big\} . \end{aligned}$$

For the sake of simplicity, we drop the level set function discretization parameter ϑ in the following and keep in mind that we are working with discrete level set functions.

4.1. The Underlying Optimization Algorithm

For the solution of the optimization problem (18), we employ an iterative scheme. We introduce a set $\omega_t = \{t_0, t_1, ..., t_N\}$ of fictitious time steps and consider a family of domains $(\Omega_{t_i})_{0 \leq i \leq N}$ which is represented by a time-dependent level set function

$$\varphi : \omega_t \times D \to \mathbb{R}, \qquad (t_i, x) \mapsto \varphi(t_i, x)$$

such that

$$\varphi_i(x) = \begin{cases} -c & \text{for } x \in \Omega_{t_i},\ d(x) \geq c, \\ -d(x) & \text{for } x \in \Omega_{t_i},\ d(x) < c, \\ +d(x) & \text{for } x \notin \Omega_{t_i},\ d(x) < c, \\ +c & \text{for } x \notin \Omega_{t_i},\ d(x) \geq c \end{cases},$$

where c is the width of the narrow band and $d(x)$ denotes the distance of a point x to the structure's boundary.

To simplify notation, we define here $\varphi_i(x) := \varphi(t_i, x)$ and denote the domain associated to $\varphi_i(x)$ by Ω_i.

φ_0 represents the initial structure, which might be chosen as the design space D. For $i \geq 1$ the level set function φ_i is computed from φ_{i-1} and the results of the structural analysis of Ω_{i-1}.

One of the simplest iterative optimization algorithms is the *method of steepest descent* with a fixed step size τ. Let φ be the state variable and let $g(\varphi)$ be a yet to be determined generalized derivative of the objective function $j(\varphi_i)$. The choice of $g(\varphi)$ will be substantiated in section 4.2. Starting with an initial guess φ_0, the descent direction $-g(\varphi_0)$ is computed and a step of length τ is taken to determine the point φ_1. This is iterated until a local minimum of j is found:

$$\varphi_{i+1} = \varphi_i - \tau g(\varphi_i), \qquad i \geq 0.$$

The method can be summarized in pseudocode as follows:

```
START
  choose a termination scalar $\varepsilon$, a starting point $\varphi_0$,
       and a step size $\tau$
  WHILE ($||g(\varphi_i)|| \geq \varepsilon$)
    compute the descent direction $\xi_i = -\frac{g(\varphi_i)}{||g(\varphi_i)||}$
    $\varphi_{i+1} = \varphi_i + \tau\xi_i$
    $i = i + 1$
  END WHILE
END
```

The main advantage of this algorithm is its simplicity. However, it is rather costly and requires many iterations, and often no stationary point can be found because the method jumps over the minimum due to the fixed step size.

The number of iterations can be reduced and the solution can be improved if *line search* is used. Instead of using a fixed step size τ, an optimal step size τ_i is computed in each iteration. τ_i is the solution or an approximation to the solution of the optimization problem

$$\min_{\tau \in \mathbb{R}} j(\varphi_i + \tau\xi_i).$$

Usually, this method works quite well at the beginning of the optimization process, but as a stationary point is approached it takes small, nearly orthogonal steps. This leads to a high number of iterations and thus evaluations of the objective function. As the method of steepest descent with fixed step size, line search will in general find a local minimum.

More sophisticated optimization algorithms could be used, for example a *conjugate gradient method* (cf., e.g., [**8, 28, 43, 21**] or any textbook on optimization).

However, since we are dealing with varying domains and do not intend to find a global minimum, we employ a slightly modified version of the line search algorithm. The modification affects only the choice of the step size τ_i. We observed in many numerical tests that a large relative variation of the objective function indicates undesirable changes in the structure such as decomposition into several parts. Therefore, we prescribe an empirically determined constant η and follow the routine given in pseudocode below.

```
START
  choose a termination scalar $\varepsilon$ and a starting point $\varphi_0$
  WHILE ($||g(\varphi_i)|| \geq \varepsilon$)
    compute the descent direction $\xi_i = -\frac{g(\varphi_i)}{||g(\varphi_i)||}$
    set $\tau_i = ||g(\varphi_i)||$
    set $\varphi_{i+1} = \varphi_i + \tau_i \xi_i$
    WHILE ($\frac{j(\varphi_{i+1}) - j(\varphi_i)}{j(\varphi_i)} \geq \eta$)
      $\tau_i = \tau_i / 2$
      $\varphi_{i+1} = \varphi_i + \tau_i \xi_i$
    END WHILE
    $i = i + 1$
  END WHILE
END
```

Using bisection for determination of the step size prevents the structure from breaking apart and yields reliable results.

To summarize, we can write the iterative solution scheme as

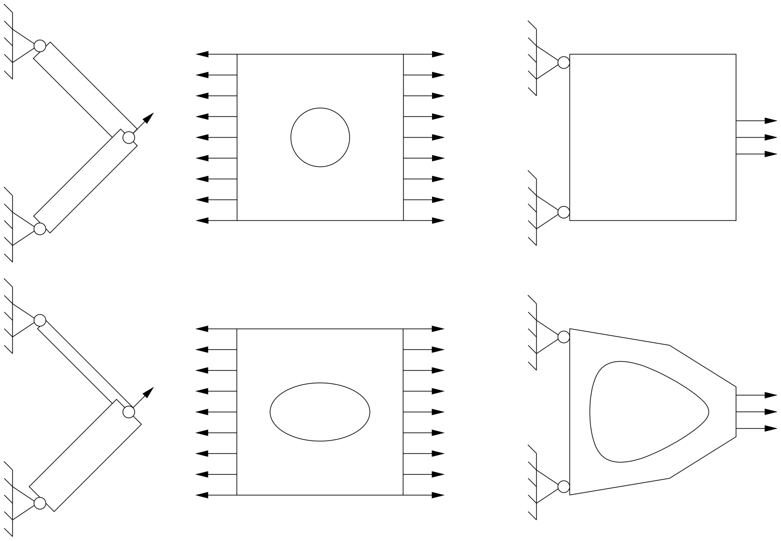
$$\varphi_{i+1} = \varphi_i + \tau_i \xi_i, \tag{19}$$

where φ_i is the level set function representing Ω_i, the step size is denoted by τ_i, and ξ_i is the descent direction computed from a generalized derivative of the objective function.

Due to practical reasons, our software reads the input, i.e., the design space D and the initial structure Ω_0, as surface triangulations. For the actual initialization step, i.e., the construction of φ_0, is based on a marching cube algorithm. This method has been originally developed by Lorensen and Cline [**32**] for construction of surfaces from a three-dimensional field of scalar values. It will not be further discussed in the scope of this work.

4.2. Topological Sensitivity

The objective of this section is to substantiate the generalized derivative g of the objective function which was introduced in section 4.1. Generally speaking, g is a criterion which prescribes how the structure has to be modified in order to decrease the value of the objective function.



(a) (b) (c)

Figure 4.2. Examples for sizing (a), shape optimization (b), and topology optimization (c).

As mentioned in the introduction, the three essential approaches which have been developed in the field of structural optimization are *sizing*, *shape optimization*, and *topology optimization*. Figure 4.2 shows simple examples for the three approaches and illustrates the differences.

Both sizing and shape optimization are important in special application areas such as steel construction or optimization of parts the approximate form of which is prescribed. For our purposes, however, topology optimization is the most suitable approach, since this method is the most flexible one. It admits basically arbitrary starting structures, not least the entire design space D as the simplest possible choice, indicates all necessary topology changes, and incorporates shape optimization.

The sensitivity used in topology optimization is called *topological sensitivity* or *topological gradient*. This concept can be traced back to Schumacher et al. [**49, 50**]. It was further developed by Masmoudi et al. [**26, 33**], and thereafter widely adopted in structural optimization ([**55, 14, 36, 7**] and many more).

The topological gradient is mathematically justified and can be computed for arbitrary elliptic systems of differential equations. For systems with piecewise constant coefficients, it can be computed analytically, which makes very efficient computation of the criterion possible.

In the following we give an account of the fundamental ideas behind the topological sensitivity as presented in [**26**]. For this purpose, we drop the notation using the domain parametrization by a level set function φ and denote the domain under consideration by Ω. A detailed alternative presentation and proof of the topological sensitivity can be found in [**34**]. There, the focus is put on inclusion of the casting process simulation into topology optimization.

The aim of the topological sensitivity is to describe how the objective function $j(\Omega)$ varies if Ω is perturbed, for example through creation of a small hole. Let $0 < \rho \in \mathbb{R}$, $x_0 \in \Omega$, and $\omega \subset \mathbb{R}^3$ be a fixed open and bounded subset containing the origin. Then Ω_ρ is defined as the set obtained by removing a small part $x_0 + \rho\omega$ from Ω, i.e., $\Omega_\rho = \Omega \backslash \overline{(x_0 + \rho\omega)}$. The physical interpretation of the perturbation ω depends on the type of boundary conditions which are applied on the boundary of the perturbation: If a homogeneous Neumann boundary condition is imposed, ω represents a perforation. A homogeneous Dirichlet condition would imply a weld or rivet at $x_0+\rho\omega$. Thus, we can restrict ourselves to Neumann boundary conditions.

In [**26**], an asymptotic expansion of j of the form

$$j(\Omega_\rho) - j(\Omega) = f(\rho)g(x_0) + o(f(\rho)) \tag{20}$$

is derived. Here, f is proportional to the volume of the hole: $f \propto \rho^3$. Furthermore, $\lim_{\rho\to 0} f(\rho) = 0$ and $f(\rho) > 0$.

The function $g(x)$ is called the *topological sensitivity* or *topological gradient.* It can be interpreted as a generalized derivative:

$$g(x_0) = \lim_{\rho\to 0} \frac{j(\Omega_\rho) - j(\Omega)}{f(\rho)}.$$

For the computation of $g(x_0)$ and $f(\rho)$ in equation (20), Masmoudi et al. [**26**] employ a generalized adjoint method and a truncation technique. For three-dimensional problems and Neumann boundary conditions on the boundary of the perturbation they obtain the following results, which we quote without proof:

The topological sensitivity is

$$g(x_0) = -(20\mu\sigma(u) : \varepsilon(p) + (3\lambda - 2\mu)\mathbf{tr}\sigma(u)\mathbf{tr}\varepsilon(p))(x_0), \tag{21}$$

and $f(\rho)$ is given by

$$f(\rho) = \frac{(\lambda + 2\mu)}{\mu(9\lambda + 14\mu)}\pi\rho^3,$$

where μ and λ are the Lamé coefficients, $\mathbf{tr}$ is the trace operator, u is the solution of the variational formulation of the elasticity problem (5), and p is the solution of the adjoint problem

PROBLEM 4.1. **Adjoint elasticity problem**

$$\begin{array}{ll}\textbf{Find} & p \in \mathcal{V}_{\Omega_R} = \{w \in (H^1(\Omega_R))^3 : w|_{\Gamma_D} = 0\} \\ \textbf{such that} & a(w,p) = -DJ(u) \qquad \forall w \in \mathcal{V}_{\Omega_R}.\end{array}$$

Here, D denotes the Fréchet derivative of the objective function with respect to u.

If the compliance is used as objective function, the problem is selfadjoint and we have $p = -2u$, i.e.,

$$g(x_0) = (40\mu\sigma(u) : \varepsilon(u) + (6\lambda - 4\mu)\mathbf{tr}\sigma(u)\mathbf{tr}\varepsilon(u))(x_0).$$

That means that $g(x)$ can be computed by a single structural analysis, which makes the method very efficient.

According to equation (20), the topological gradient describes how the function j will change if a small hole is created at x_0. Indeed, removing material where $g(x) < 0$ will decrease the objective function.

REMARK 4.2. The topological gradient is defined via creation of a single hole of arbitrary shape. Since nucleation of a hole can change the results of the structural analysis in the entire domain, it cannot be taken for granted that the approach is valid if several holes are created. We did not investigate this point. However, since we never experienced an increase of the objective function due to creation of disjoint holes, we set aside measures preventing multiple holes and allow for large step sizes in the optimization algorithm.

Obviously, $g(x)$ defines the following optimality criterion: When

$$g(x) \geq 0 \qquad \forall x \in \Omega,$$

j cannot be further reduced and a local minimum has been reached.

$g(x)$ gives the step size τ in the iterative scheme (19) implicitly. The modified structure Ω_{i+1} can be obtained from Ω_i by removing all points x with $g(x) < 0$:

$$\Omega_{i+1} = \{x \in \Omega_i | g(x) \geq 0\}.$$

Alternatively, the authors of [**26**] propose the definition of a decreasing sequence $(m_k)_{k\geq 0}$ of volume constraints with $m_0 = \text{meas}(\Omega_0)$. This, in turn, generates a sequence $(c_k)_{k\geq 1}$, where c_{k+1} is chosen such that

$$\begin{cases} \Omega_{k+1} = \{x \in \Omega_k | g_k(x) \leq c_{k+1}\}, \\ \text{meas}(\Omega_{k+1}) = m_{k+1}. \end{cases}$$

In this case, holes are created in the points which have the lowest values of $g(x)$.

As mentioned above we observed that a large relative variation of the objective function often indicates a break up of the structure, which is not desirable. Therefore, we choose an empirically determined constant η, prescribe a maximal admissible volume change per iteration ΔV_{max}, and define c_k as the largest constant c fulfilling

$$\begin{cases} \Omega_{k+1} = \{x \in \Omega_k | g_k(x) \leq c\}, \\ \dfrac{\text{meas}(\Omega_k) - \text{meas}(\Omega_{k+1})}{\text{meas}(\Omega_k)} \leq \Delta V_{max}, \\ \dfrac{j(\Omega_{k+1}) - j(\Omega_k)}{j(\Omega_k)} < \eta. \end{cases}$$

Usually, the relative volume change is the decisive criterion for determination of c_k. This is essential for computational efficiency, since evaluation of the second criterion, namely the relative change of the objective function, requires a complete structural update, mesh generation, and structural analysis. In our sample computations, we obtained generally good results for $\Delta V_{max} \leq 0.10$.

4.3. Structural Update

For the structural update basically any combination of a generalized gradient or descent direction and a method to modify the structure's boundary can be chosen. As mentioned earlier, we employ the topological gradient g as descent direction. Due to restrictions imposed by incorporation of the casting process into the optimization algorithm, we perform the structural update on the level set function, which gives us the exact position of the boundary at any time. Thereafter, the FE mesh is adapted to the new structure on the basis of the updated level set function.

An overview of the various approaches used for structural update in topology optimization has been given in section 0.1.

4.3.1. Update of the Level Set Function. During the last years, the use of level set functions in topology optimization has become very popular. Originally, this type of boundary representation has been developed by Osher and Sethian [**38**] for modeling of physical phenomena involving propagating interfaces as for example fluid dynamics or combustion processes.

There, boundary movement is controlled by the time dependent initial value problem

$$\begin{aligned} &\phi_t + F|\nabla\phi| = 0, \\ &\phi(x, t = 0) \quad \text{given}, \end{aligned} \tag{22}$$

where ϕ_t is the time derivative of the level set function, F is a so-called speed function, and $\nabla\phi$ denotes the spatial derivative of the level set function.

In the context of topology optimization, F usually depends on the stresses σ in the structure and on the structure's boundary, respectively (see, e.g., [**14, 52, 4, 2**]).

The benefits of the application of level set functions are flexibility, robustness and efficiency. All kinds of topology modifications such as nucleation or merging of holes can be performed. The use of a Cartesian grid combined with a narrow band technique leads to fast algorithms.

Our approach employs level set functions, too, and takes advantage of these strong points. However, we do not use a velocity as described above but base the structural modification directly on the topological gradient. Thus, we manage without solution of a Hamilton-Jacobi equation as (22). Our method stands out due to its simplicity and efficiency.

As described in section 4.1, we use a time-dependent level set function, introduce the set $\omega_t = \{t_0, t_1, ..., t_N\}$ of fictitious time steps, and update the structure by the iterative scheme

$$\varphi_{i+1} = \varphi_i + \tau_i g_i,$$

where $\varphi_i(x)$ is a short notation for $\varphi(t_i, x)$ and denotes the level set function associated to the domain Ω_i.

Our realization of the modification of the structure is based on the following considerations:

- In the domain Ω_i, we have to increase the level set function in points where $g_i(x) < 0$ in order to create holes in these points.
- The actual value of the level set function in a point x is of interest only if $\min_{y\in\partial\Omega_i} |x - y| < c$, i.e., x is close to the structure's boundary. Otherwise, the sign of φ is sufficient. This comes from the fact that only points closer than c are needed to determine the exact position of the zero level set by interpolation in the discretized problem.

Therefore, instead of evolving the level-set function in time by a Hamilton-Jacobi equation propagating the interface $\Gamma_0(t)$, we propose the following way of updating the level set function φ_i:

Assume $g(t_i, x_0)$ indicates that we have to remove material at the point x_0. Since we use the signed distance function, it is sufficient for creation of a hole of radius $r < c$ to set $\varphi_{i+1}(x_0) = r$ and to adapt $\varphi_i(x)$ in the surrounding points $x \in \mathcal{N}_{x_0}(r)$, such that φ_{i+1} remains a correct signed distance function. Here,

$\mathcal{N}_{x_0}(r) = \{x \in D \; s.t. \; |x - x_0| < 2r\}$ is the neighborhood of radius r around x_0. This approach implements merging and deletion of holes as well as trouble-free creation of new holes at arbitrary points, including boundary points. Thus, it provides both shape and topology optimization.

For a formal description of this procedure we introduce a new level set function $\psi_{x_0,r} \hat{\equiv} B_r(x_0)$ representing the hole of radius r to be created at position x_0. Furthermore, we define a special addition operator for level set functions:

Definition 4.3. *The* **level set function addition operator**

$$\oplus : C^0(D) \times C^0(D) \to C^0(D)$$

is defined as the maximum of its arguments:

$$(\varphi \oplus \psi_{x_0,r})(x) = \max(\varphi(x), \psi_{x_0,r}(x)).$$

Then we obtain the updated level set function φ_{i+1} as the sum of φ_i and the functions describing the holes to be created:

$$\varphi_{i+1} = \varphi_i \oplus \sum_{(x,r) \in \mathcal{B}_i} \psi_{x,r},$$

where

$$\begin{aligned} \mathcal{B}_i &= \mathcal{B}(g_i, V_{t_i}) \\ &= \left\{ (\hat{x}, \hat{r}) \in \Omega \times \mathbb{R} \text{ s.t. } | \bigcup_{(\hat{x},\hat{r}) \in \mathcal{B}_i} B_{\hat{r}}(\hat{x})| = V_{t_i} \text{ and } g_i(\hat{x}) < g_i(x) \; \forall x \notin \mathcal{B}_i \right\} \end{aligned}$$

is the set of center points of the holes which have to be created such that the desired amount V_{t_i} of material is removed and $\sum$ is to be understood as a short notation for multiple application of the summation operator $\oplus$ defined above.

Close to the boundary $\partial\Omega_{i+1}$ of the updated domain, φ_{i+1} is the signed distance function. In all points x with distance to $\partial\Omega_{i+1}$ greater than c, $\varphi_{i+1}(x)$ is constant:

$$\begin{aligned} \varphi_{i+1}(x) &= -c \quad \text{for } x \in \Omega_{i+1} \quad \text{and} \\ \varphi_{i+1}(x) &= +c \quad \text{for } x \in D \setminus \Omega_{i+1}. \end{aligned}$$

This means that φ_i has to be adapted only in a small neighborhood of the modified free boundary. In addition, the approach is completely robust. Problems such as sharp corners, ambiguously defined normals, or self-intersection as

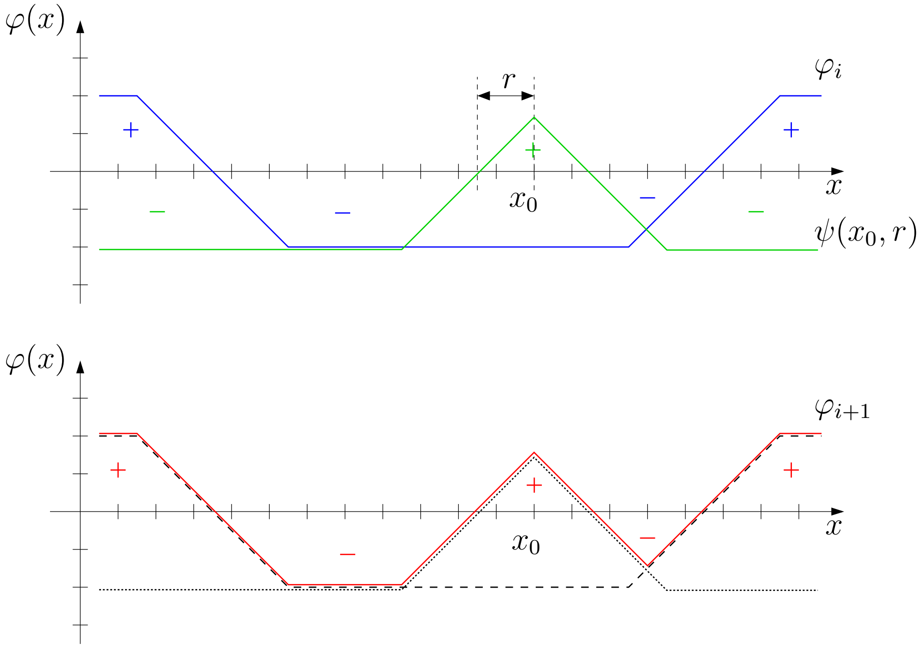


FIGURE 4.3. Update of a one-dimensional continuous level set function. In the upper figure, the blue line is the original level set function φ_i, whereas the green line is represents a hole of radius r which is centered at x_0. The lower part of the figure shows the updated level set function (red) resulting from addition of φ_i (dashed line) and $\psi_{x_0,r}$ (dotted line).

described in [**51**] for interface evolution by a Hamilton-Jacobi equation cannot occur.

Having presented the update of the continuous level set function, the discrete case is straightforward. All occurring level set functions (φ_i, φ_{i+1}, and the holes $\psi_{x,r}$) are discretized on the same Cartesian grid as described in section 2.2. The constant c is defined as two times the width of the narrow band. The summation of level set functions is performed pointwise, and φ_ϑ has to be modified only in a small tube around the modified boundary which makes the method highly efficient.

In a small neighborhood of areas where boundary conditions are applied, the values of the level set function are fixed. Here, removal of material is prohibited. Thereby, we guarantee that the load cases under consideration remain unmodified throughout the entire optimization process. Besides this approach we have implemented the possibility to explicitly define free surface areas. In this case, structural modifications may only be performed in the structure's interior and the designated surface patches. Erosion and dilation are applied as described in section 2.3 to prevent development of too fine features and holes. Again, boundary condition patches are excluded from the process.

Our level set-based realization of the structural update goes very well in line with the iterative nature of the optimization algorithm. Even large models can be handled efficiently and accurately, and frequent modifications of the structure are no problem. It becomes clear that the solution of the elasticity problems is the most time-consuming part of the algorithm and in some sense a limiting factor. We want to underline that high efficiency is not achieved at the expense of stability.

4.3.2. Modification of the FE Mesh. In order to evaluate the objective function and to determine whether further iterations in the optimization process have to be performed, we have to solve the equations of elasticity in the updated domain. This requires creation of a finite element mesh approximating Ω_{i+1}. One possible way is to apply the mesh generation algorithm described in chapter 3. Since our mesh generation method is very fast, this would not lead to an unacceptable increase in computing time. However, it is possible to achieve a significant speed-up by exploiting the knowledge of the points in which the updated level set function φ_{i+1} differs from φ_i.

The computationally most expensive part of mesh generation is the computation of minimizers of the quadratic error function e_q. Therefore, this is the point where it is most profitable to save CPU time.

It is obvious that large parts of $\partial\Omega_i$ and $\partial\Omega_{i+1}$ coincide. This is true at least for all surface patches where boundary conditions are applied, but in general for large portions of the free boundary, too. Thus, it is proximate to keep as much of the old finite element mesh as possible and to remesh only those parts where the structure has been modified.

The differences between mesh generation and mesh modification are very small. Let us denote the union of the cells of the Cartesian grid $\mathcal{G}_h$ in which φ_i has been changed by Ω_{mod}. Then we only have to delete all tetrahedra in Ω_{mod} (in the uniform case). The rest of the mesh, especially the computationally expensive minimizers in $\Omega_{bor} \setminus \Omega_{mod}$, will not be modified. After this clean-up, we simply use our mesh generator restricted to Ω_{mod} to complete the partially erased mesh and to adapt it to the updated level set function.

Since mesh modification and generation are almost identical and the latter has been described in great detail in chapter 3, we restrict ourselves to a pseudocode representation of the mesh modification algorithm for the uniform case.

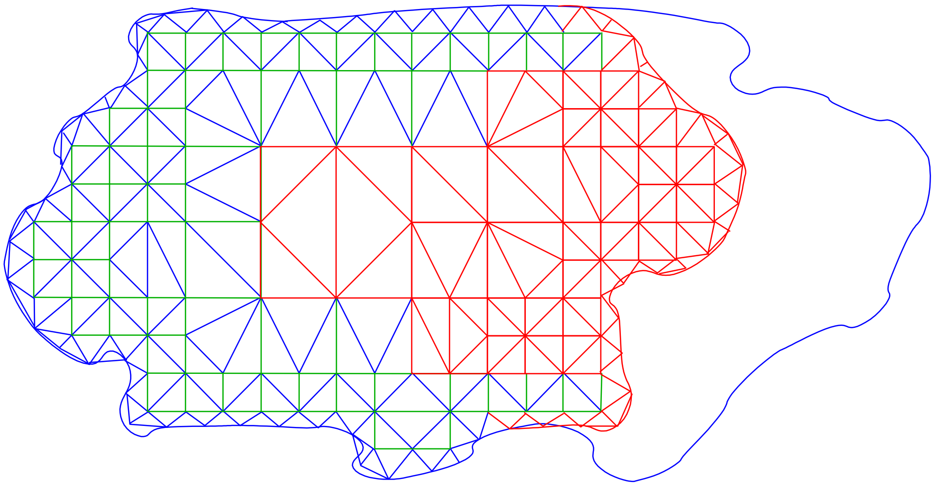


Figure 4.4. Mesh modification. The red portion of the structure's boundary has been modified, and only the red part of the mesh has been reconstructed. Most of the minimizers and a large part of the interior mesh remain unchanged, which keeps the computing time low.

```
START
  delete all tetrahedra in the cells belonging to $\Omega_{mod}$
  divide the grid cells of $\Omega_{mod}$ into interior and boundary
      cells, respectively
  adapt the modified octree representation of $\Omega_{int}$
  mesh $\Omega_{int} \cap \Omega_{mod}$ using pre-computed decompositions of blocks
  compute a minimizer for each cell of $\Omega_{bor} \cap \Omega_{mod}$:
      $p_{min} = \min_{x \in S_r} e_{q2}(x)$
  mesh $\Omega_{bor} \cap \Omega_{mod}$:  construct tetrahedra assigned to interior
      faces, edges, and points
  WHILE (mesh quality not high enough)
    smooth the mesh of $\Omega_{mod}$ using equation 18
  END WHILE
END
```

In the case of adaptive mesh generation, the entire mesh of Ω_{int} has to be deleted because the modified octree representation may vary significantly from the original one. In the uniform case, the new octree is simply a subset of the old octree representation.

Figure 4.4 illustrates the process in two dimensions. The outer blue line is the boundary of the original domain Ω_i. Green squares indicate the unmodified portion of the underlying quadtree structure. The modified part of both quadtree and FE mesh is given in red. Even in this extreme example which displays an exceptionally large step from Ω_i to Ω_{i+1}, about two thirds of the minimizers are carried over from the original mesh.

Naturally, element quality evaluation yields exactly the same good results for the modified mesh as for meshes generated from scratch.

Since φ is fixed in areas where boundary conditions are applied, the mesh is not modified in these regions. Hence, the boundary condition mapping has to be performed only in the initial mesh generation but not in mesh modification. Besides the reduction of computing time, this guarantees that the boundary condition patches are exactly the same in all iterations. This is crucial for reliable evaluation and comparability of the objective function.

4.4. The Complete Numerical Algorithm

In order to summarize the presentation of the numerical algorithm, we give a flow chart describing the entire process chain in Figure 4.5.

The processes can be grouped in three major units, which are indicated by underlying grey boxes: Initialization, sensitivity analysis, and structural update. Clearly, these three steps correspond to the iterative scheme (cf. equation (19))

$$\varphi_{i+1} = \varphi_i + \tau_i g_i.$$

Starting from an initial guess φ_0, we can iteratively compute improved structures with the help of the topological gradient g_i and a step size τ_i.

The initialization step consists of three parts: First, the input data has to be specified. For practical reasons, the actual initial guess, which may be chosen as the entire design space, has to be provided as a surface triangulation (stl format). In addition, a triangulation of each boundary condition surface must be available, but these triangulations can differ from the one describing the entire structure. Furthermore, the applied boundary conditions must be specified in a text file. Since the load cases are transferred automatically to the FE mesh, no mesh dependency like element or node numbers is contained in the boundary condition file. The second and third step are initialization of the level set function with a marching cube-based algorithm and generation of the initial FE mesh with the algorithm described in chapter 3, respectively. Here,

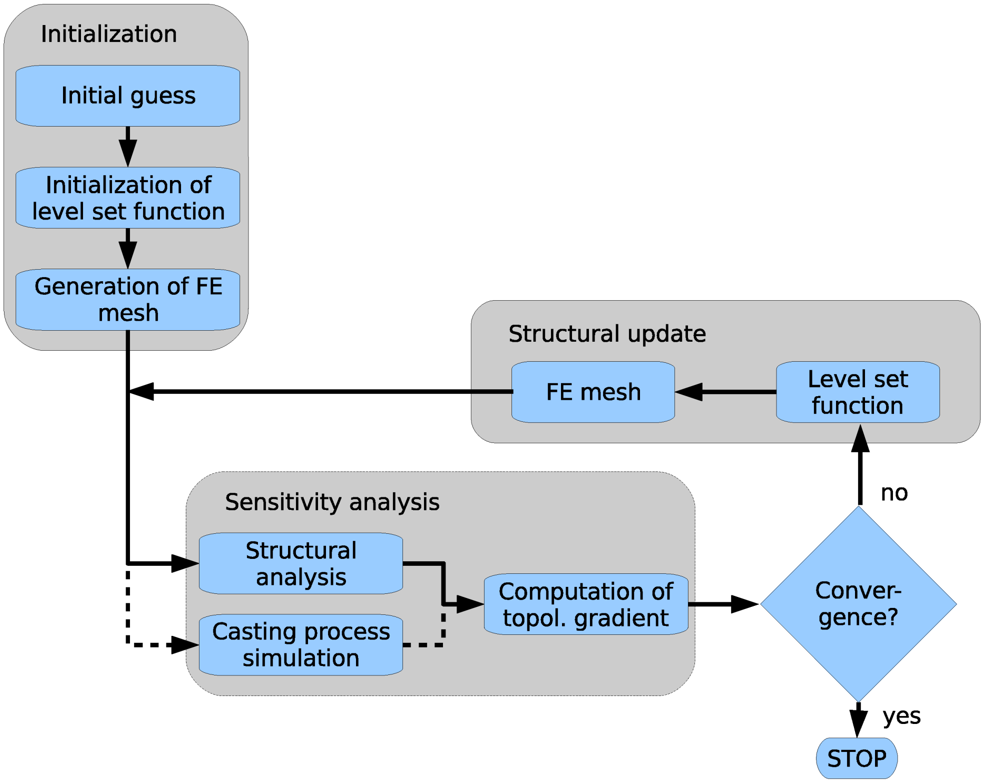


FIGURE 4.5. Flow chart of the processes in our topology optimization algorithm.

the width of the Cartesian grids for discretization of the level set function, ϑ, and FE mesh generation, h, are chosen. Naturally, the entire initialization step has to be done only once.

Sensitivity analysis comprises the processes which are required for the computation of the topological gradient, i.e., the criterion which prescribes how the structure has to be modified. In the standard case, only a structural analysis has to be performed, i.e., the variational formulation of the equations of elasticity (8) has to be solved. For this purpose we employ the parallel finite element solver DDFEM [**67**], which has been developed at the Fraunhofer Institute for Industrial Mathematics. The solver is based on domain decomposition and scales very well with the number of processors. Parallelization of this computationally intensive process reduces computing time significantly. If in addition eigenstresses have to be considered, we have to perform a casting process simulation besides the structural analysis. Here, we use the software package MAGMA [**68**], which is widely used in foundries for simulation of cooling down, solidification, and the development of eigenstresses. The topological gradient is computed according to equation (21). Consideration of eigenstresses introduces an additional term in the formula for the topological gradient. Then, one obtains

$$g(x_0) = -(20\mu\sigma(u) : \varepsilon(p) + (3\lambda - 2\mu)\mathbf{tr}\sigma(u)\mathbf{tr}\varepsilon(p))(x_0) - 2|\omega|b(\varepsilon_0)u_0(x_0),$$

where $|\omega|$ is the volume of the created hole, ε_0 are the strains in the unperturbed domain, and $u_0(x_0)$ is the displacement at point x_0 (the center of the hole) in the unperturbed domain. For the derivation of this formula and more information on the importance of eigenstresses and the casting process itself, we refer to [**34**].

If the topological gradient $g(x) > 0$ for all $x \in \Omega_i$, a local optimum has been reached and the optimization process is stopped. If this is not the case, the structure has to be modified and another iteration has to be performed.

The structural update concerns both the level set function representing the boundary $\partial\Omega_i$ of the domain Ω_i and the FE mesh associated to this domain. These processes have been described in sections 4.3.1 and 4.3.2, respectively. The boundary conditions are automatically transferred to the modified FE mesh.

CHAPTER 5

Numerical Examples

5.1. Bridge

The first example is the expansion to 3D of a two-dimensional problem that has been treated, e.g., in [**2, 3, 4**]. The design space D is the cuboid $3.0 \times 2.0 \times 1.0$ with zero displacement at the bottom left side and zero vertical displacement at the bottom right side. In the middle of the bottom face, a distributed vertical load is applied (see Figure 5.1).

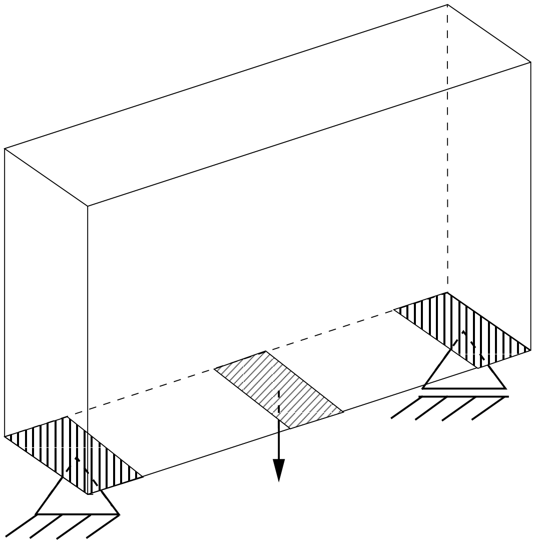


FIGURE 5.1. Schematic view of the loads applied to the cuboid. The marked areas at corners of the bottom face are subject to zero displacement (left) and zero vertical displacement (right), respectively. On the shaded area in the middle, a distributed vertical force is applied.

In general, topology optimization results cannot be transferred from 2D to 3D. However, our results resemble very much what is known from similar problems in 2D. Indeed, the optimized structure's projection onto the x-z-plane gives the typical structure of a circular arc which is connected by spokes to the point of application of the load.

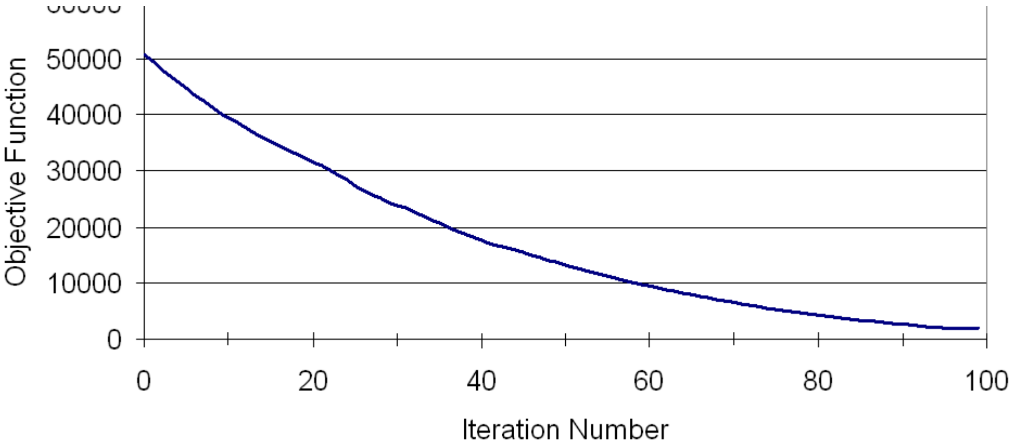


FIGURE 5.2. Plot of the objective function. The algorithm converges after 99 iterations

The computation was performed using the following input data: The width h of the underlying Cartesian grid was determined by (13) with $c = 20$. The resulting initial mesh consisted of 61,809 nodes and 294,252 elements. The target volume was set to 20 % of the initial volume, which is a very low value and usually only applicable if one accepts a significant increase of the compliance. The maximum volume change per iteration was set to 1 %. This number is always related to the structure's volume in the current optimization iteration, not to the initial volume. Naturally, such a small volume change increases the number of iterations needed to reach the target volume significantly. In the displayed example, the algorithm converged after 99 iterations. The total computation time was 3 hours, 15 minutes and 25 seconds on a machine with two Intel(R) Xeon(R) Quad Core CPUs (2.33GHz) and 16 GB RAM.

Figure 5.2 shows a plot of the objective function against the iteration numbers. The results of the structural analysis of the initial structure are shown in Figure 5.3, some steps of the optimization are shown in Figures 4(a) to 4(e). Figure 5.5 depicts the final result. In all examples we display the von Mises equivalent stress as a quantity relevant in practice. The step from the initial design to iteration 8 demonstrates clearly that material is removed in areas of low stress indicated by dark blue colors in the upper corners of the cuboid. In these areas, both von Mises stress and the topological gradient attain the lowest values. Since the color scale is identical for all results, it is clearly visible that the maximum von Mises stress increases as the amount of material is reduced. Symmetry is not preserved in the course of the optimization process, since the

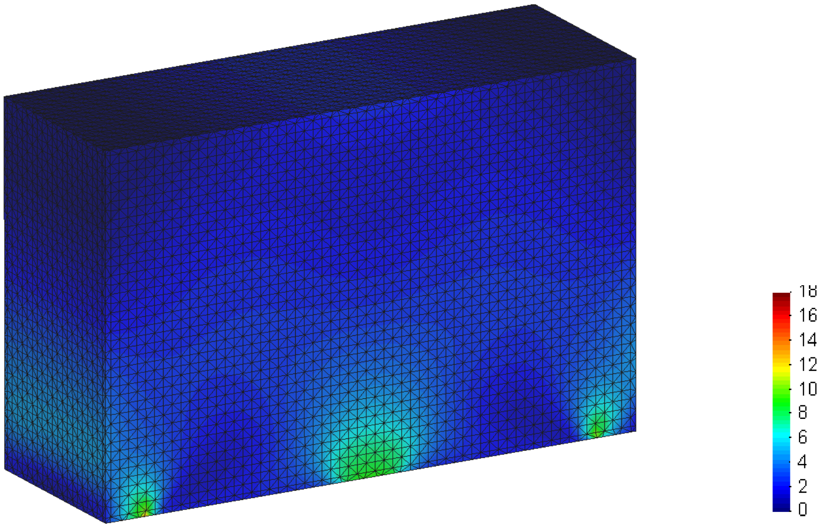
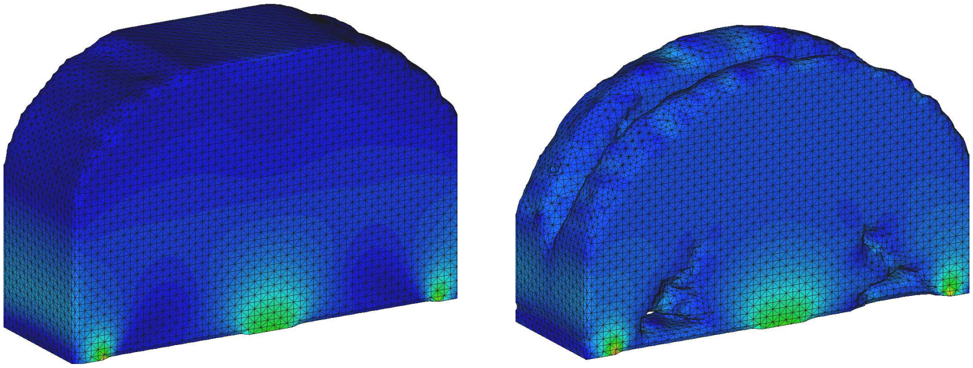


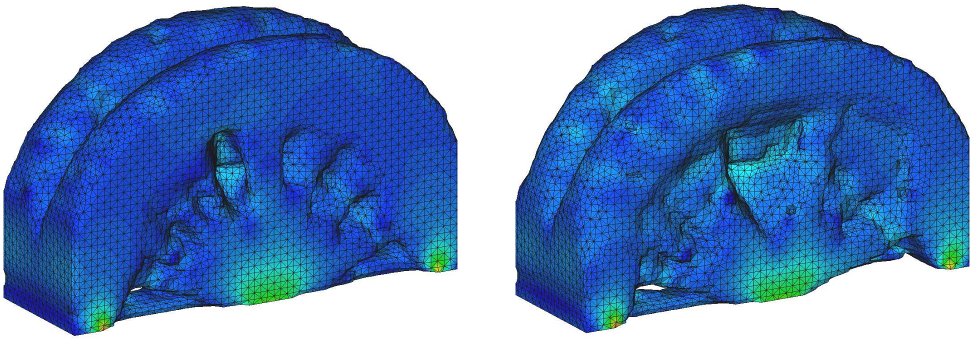
Figure 5.3. Von Mises stress on the initial structure.

finite element meshes are not symmetric, which leads to slightly asymmetric results of the structural analysis.

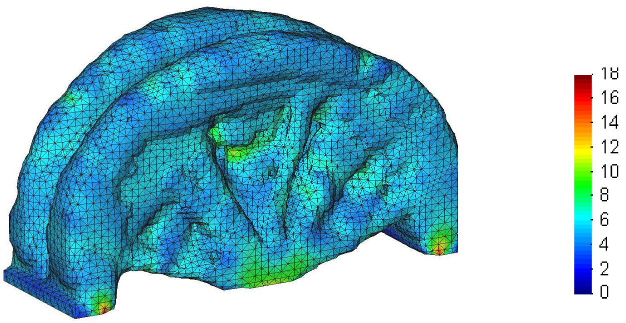
In order to illustrate that not only shape optimization but topology optimization is performed, we give a transparent view of the results of iteration 12 in Figure 5.6. The two lower holes are connected to the exterior, whereas the hole in the upper part of the structure is completely enclosed by the domain Ω. Only as the hole grows in later iterations, it connects to the exterior and thereby separates the two arcs of the bridge structure.



(a) 8th iteration. (b) 39th iteration.



(c) 45th iteration. (d) 53rd iteration.



(e) 76th iteration.

Figure 5.4. Some intermediate results of the optimization process.

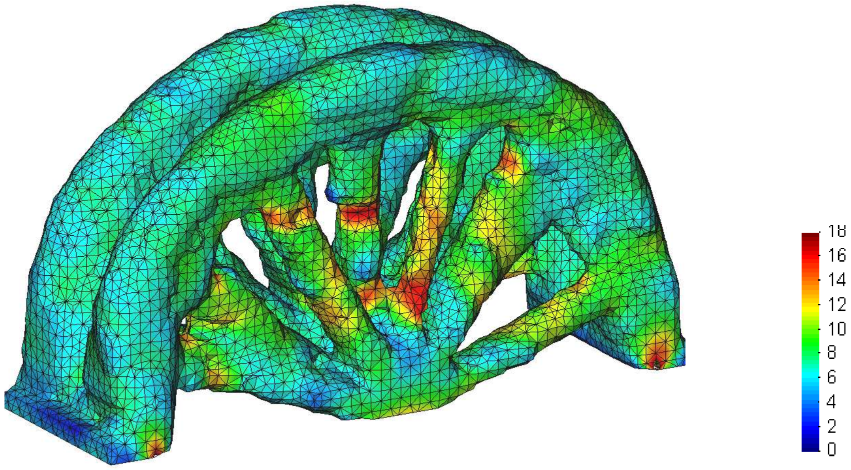


Figure 5.5. 99th iteration. The volume of the structure has been reduced to 29.2 % of the initial volume.

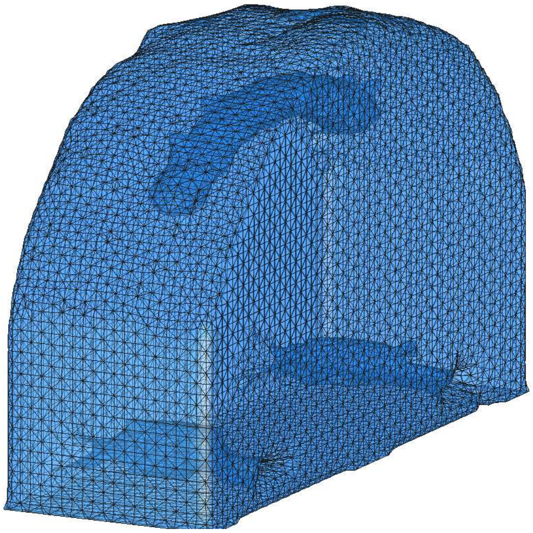


Figure 5.6. Transparent view of the structure after 12 iterations. The upper hole is not connected to the exterior, which means that both shape and topology optimization is performed.

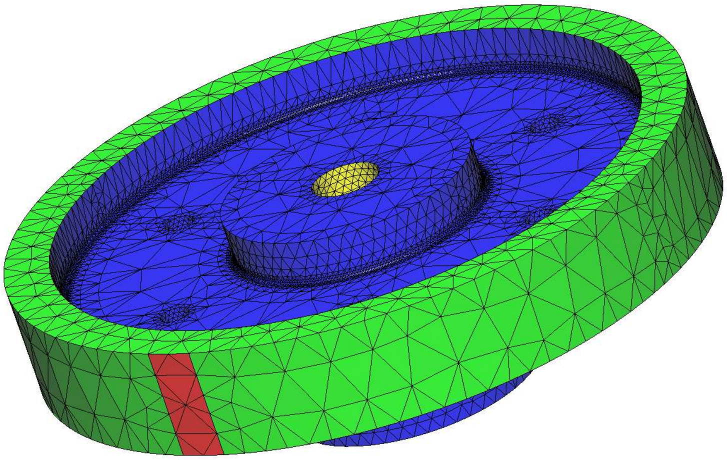


Figure 5.7. Load case of the dial plate. Yellow and green areas are subject to homogeneous Dirichlet conditions. A distributed tangential force is applied on the red rectangle. Only blue areas can be modified.

5.2. Dial Plate

Our second example is a dial plate which is used for mounting various tools for metal machining. Usage and resulting load case can be explained with the help of Figure 5.7. The rotary plate is clamped to a hub in the center. In order to guarantee a maximal contact surface between hub and plate, a homogeneous Dirichlet condition is applied at the yellow area, i.e., the center must not be modified in the course of the optimization. The mounted tool is simulated by a tangential force which is distributed over the red rectangle. In order to prevent too large an unbalance, the entire rim (green) is considered a fixed but unloaded area. Thus, only the blue areas can be altered.

The following parameter values were chosen: The constant c in (13) was set to 30, which resulted in an initial mesh consisting of 50,500 nodes and 224,995 tetrahedra. The target volume was set to 50 % of the initial value, and the maximum volume reduction per iteration was 3 %.

The objective function, which is plotted in Figure 5.8, decreases monotonically. After 16 iterations, the target volume was reached and the algorithm terminated. The final structure is shown in Figure 12(b). However, this design might not be chosen for production since the compliance value has increased

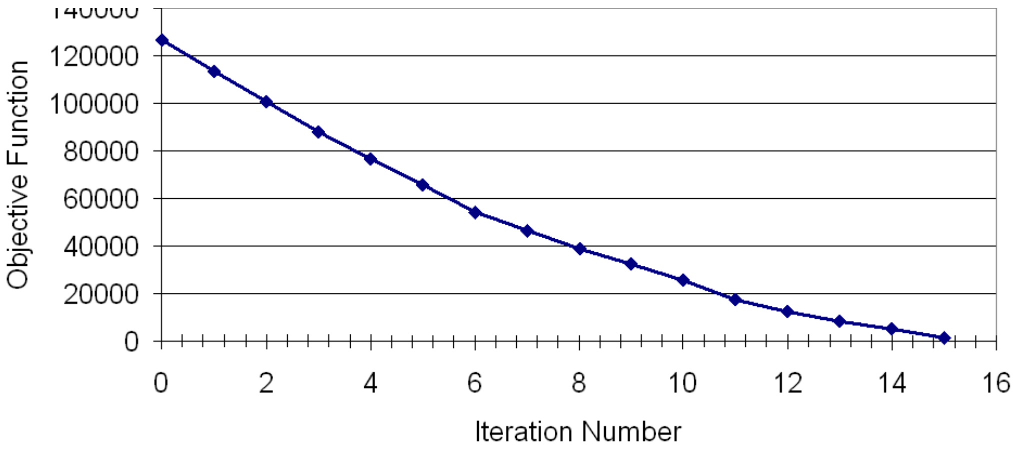


FIGURE 5.8. Plot of the objective function.

by 135 %. A more realistic shape might be given by the result of the fifth iteration (cf. Figure 11(d)), which combines a 20 % volume reduction with an increase of compliance smaller than 5 %. Naturally, the final choice depends on the required manufacturing precision.

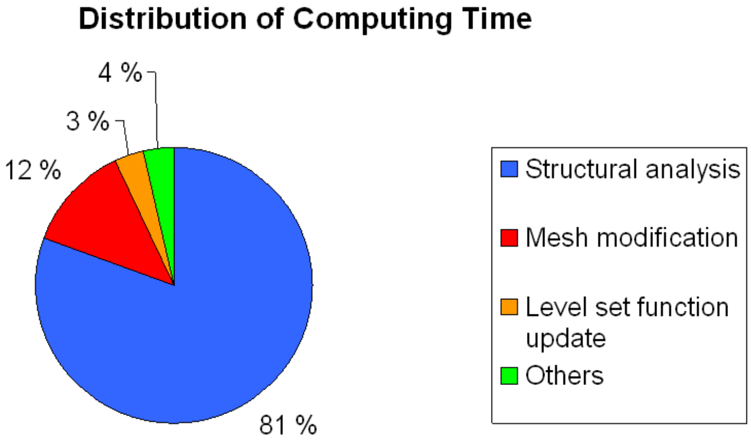


FIGURE 5.9. Distribution of computing time. The repeated structural analysis takes the by far largest part of the time. The structural update, which includes modification of both the level set function and the finite element mesh, takes only 15 % of the total computing time. Input, output, and internal data management account for 4 % of the time.

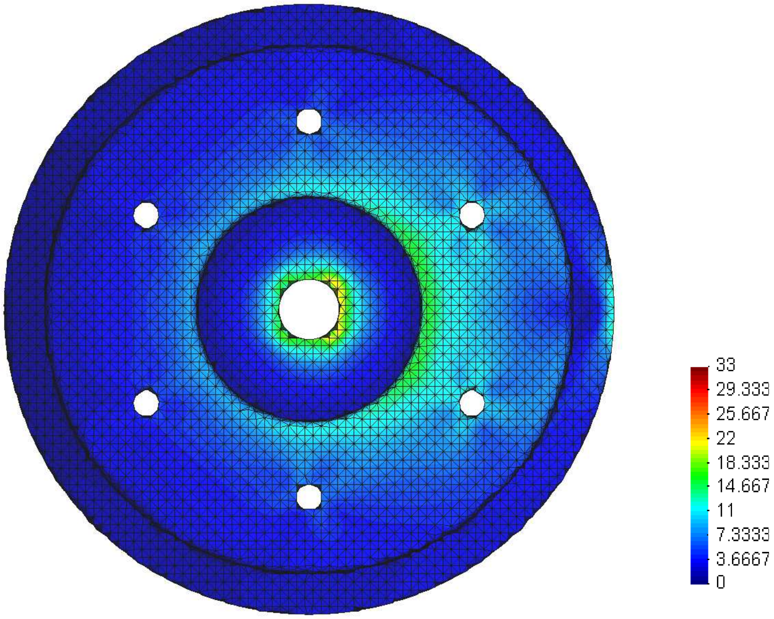
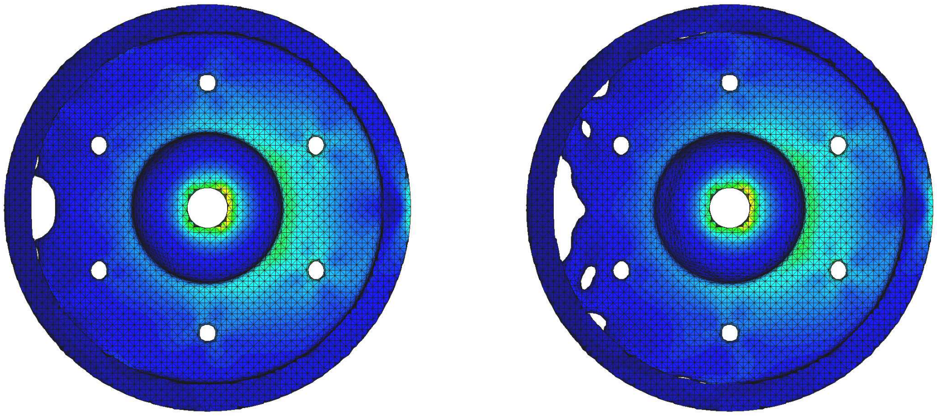


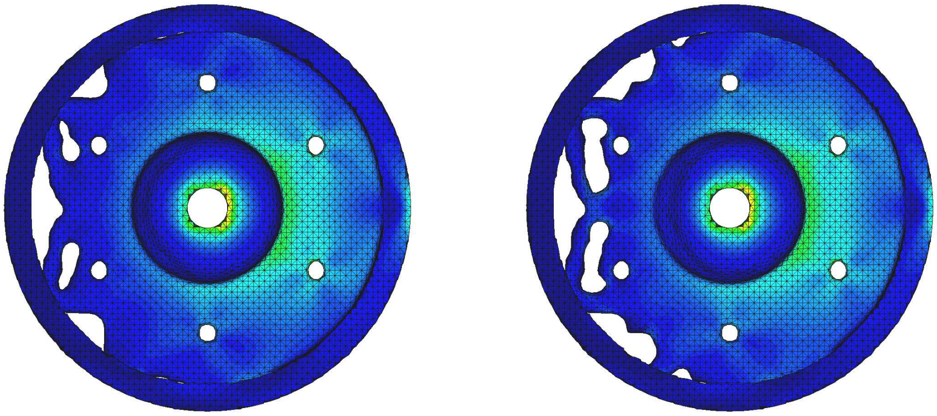
FIGURE 5.10. Von Mises stress on the initial configuration.

The total computing time for this example was 1 hour, 1 minute and 25 seconds. Again, we used a machine with two Intel(R) Xeon(R) Quad Core CPUs (2.33GHz) and 16 GB RAM. The distribution of computing time between the main processes is shown as a pie chart in Figure 5.9. As expected, the solution of the finite element problem takes the by far largest part of the time, whereas the structural updates and mesh modifications of all iterations together with input, output, and internal data management consume only 15 % of the total computing time. This ratio becomes even larger if one considers the fact that the parallelized FE solver uses two CPUs with four cores each, whereas the rest of the program is run on a single core. Thus, the distribution of computing time underlines the efficiency of the algorithms for the structural update we presented in this thesis.

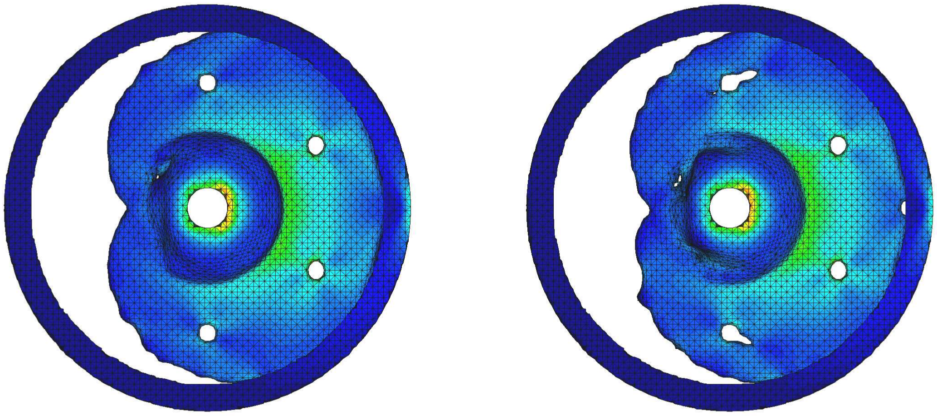
As for the academic example in section 5.1, we show the von Mises stress on the structure's surface at several iterations. Figure 5.10 is the original configuration, Figures 11(a) to 12(a) show intermediate results, and Figure 12(b) is the final result fulfilling the volume constraint but having an extreme increase in compliance. The color scale is the same for all iterations. Since this is not clearly visible due to symmetry of the initial part, we want to point out that the area of application of the tangential load is located at the rightmost point of the pictures, at the same height as the dial table's center.



(a) 2nd iteration. (b) 3rd iteration.

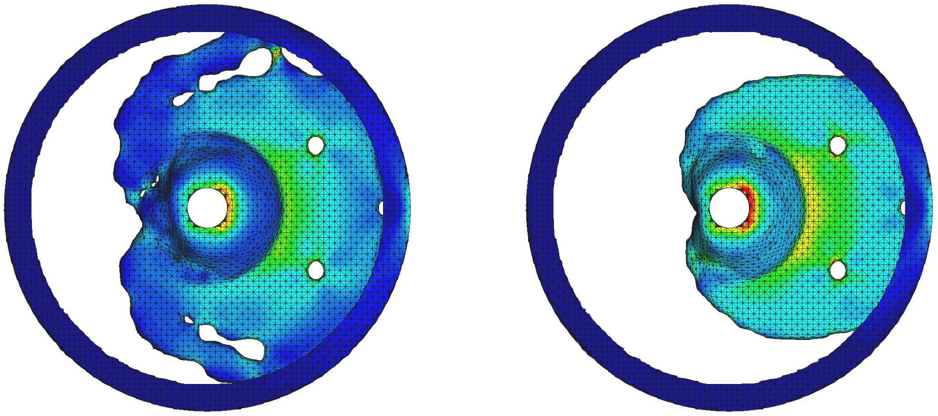


(c) 4th iteration. (d) 5th iteration.



(e) 9th iteration. (f) 10th iteration.

FIGURE 5.11. Some intermediate results of the optimization process.



(a) 11th iteration.

(b) 16th iteration.

Figure 5.12. Some intermediate results of the optimization process.

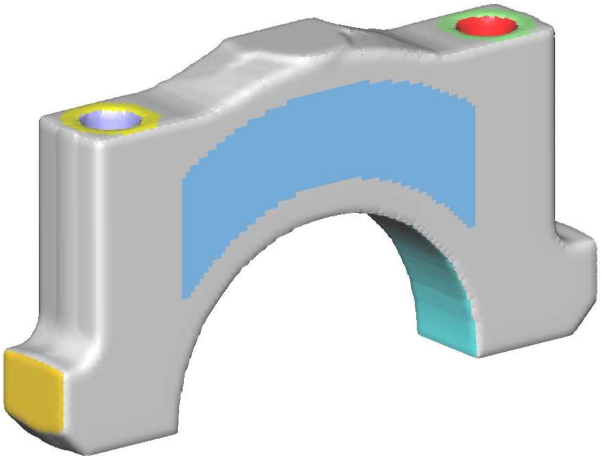


Figure 5.13. Load case for the bearing cap.

5.3. Bearing Cap

The last example we want to consider in this thesis is a bearing cap which is used in car manufacturing to secure the shaft transmitting the rotational movement from the motor to the traction axle.

The boundary condition surfaces are highlighted in Figure 5.13. There is a distributed load acting normal to the wall of the semicylinder holding the shaft (turquoise area) and the part is mounted with two bolts running through the drilling holes and exerting vertical, downwards pointing forces at the rings on top of the structure. The walls of the drilling holes and the small vertical patches at the narrow faces are fixed. In a conservative scenario prescribed by the manufacturer, only the large light blue surface patch on the front side and a corresponding surface patch on the opposite face of the bearing cap are defined as free boundaries. Therefore, the effects of topology optimization are restricted to the volume above the semicylinder.

For this example, we want to compare two computations differing only in the maximal admissible volume change per iteration (3 % and 10 %, respectively). The initial mesh consisted of 34,624 nodes and 156,629 elements. The target volume was 80 % of the initial volume.

The plot of the objective functions (cf. Figure 5.14) shows that our algorithm tends to run into local minima. Reducing the structure's volume significantly in each iteration decreases the objective function much more than the optimization with a small step size. Despite the fact that the volume of the resulting structure for the large step size is slightly lower than for the other computation

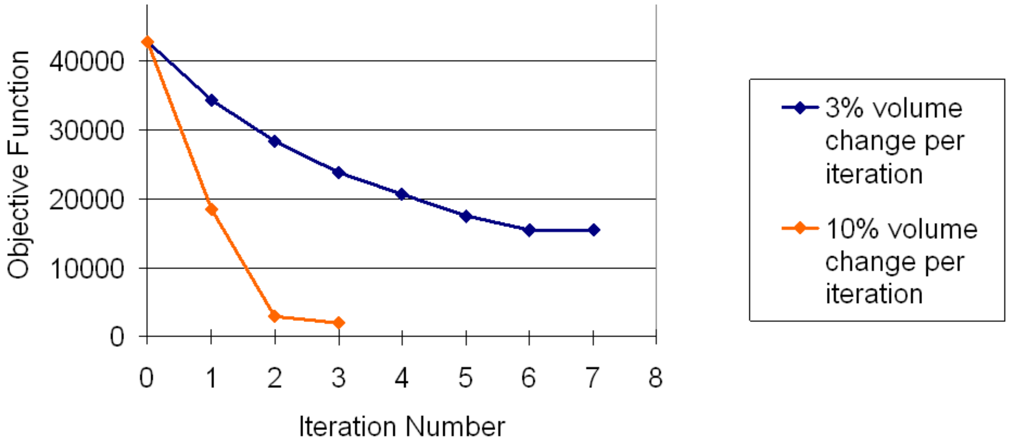


FIGURE 5.14. Plot of the objective function.

(79.25 % of the initial volume compared to 83.70 %), the compliance is lower (102.38 % of the value of the original structure compared to 106.64 %).

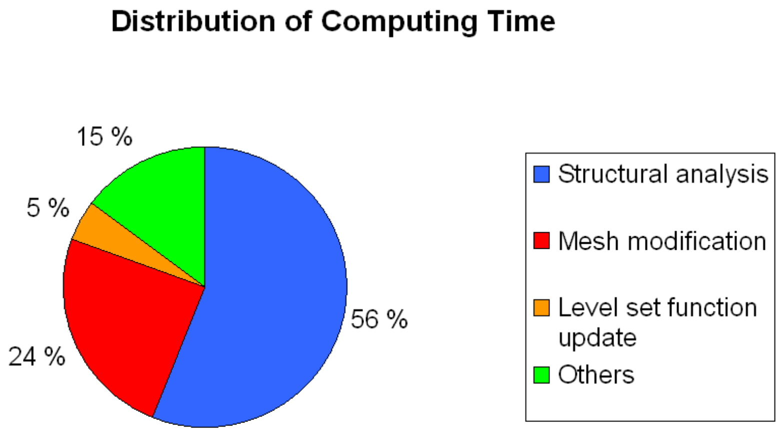
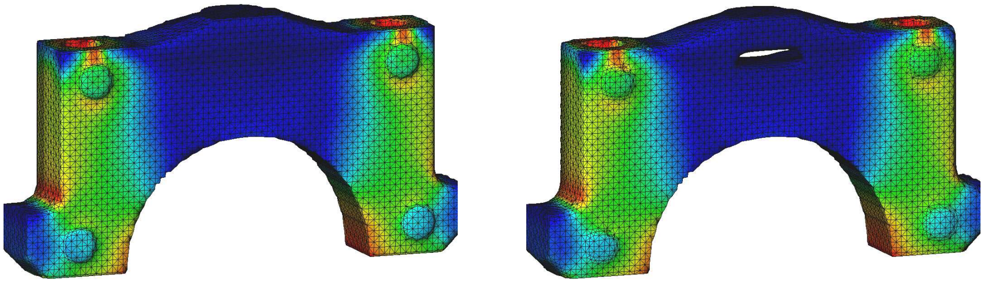
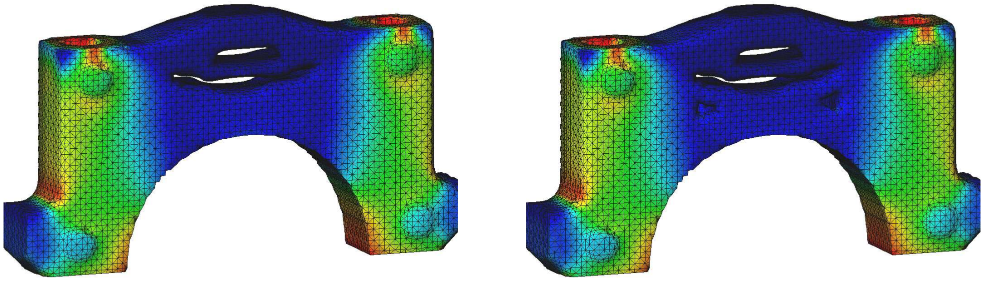


FIGURE 5.15. Distribution of computation time, 7 iterations.

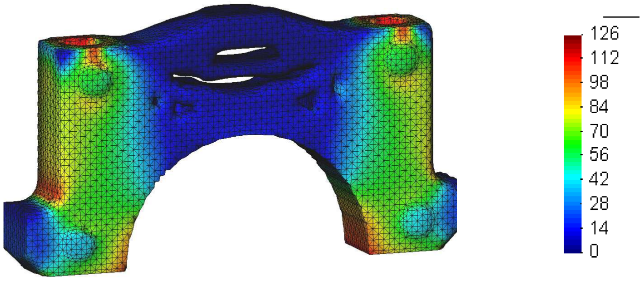
The computing times were exactly 17 minutes (3 % volume reduction) and 7 minutes, 25 seconds (10 % volume reduction), respectively. The distribution of computing time for the case is shown in Figure 5.15. It confirms the results described in section 5.2. For the second case, the distribution is almost identical.



(a) Initial configuration. (b) 1st iteration.



(c) 2nd iteration. (d) 3rd iteration.



(e) 4th iteration.

Figure 5.16. Some intermediate results of the optimization process.

The initial configuration and intermediate results of the optimization using the smaller step size are shown in Figure 5.16. Iteration 5 is omitted since no differences to the final structure are visible from the outside.

The final optimized structures for both step sizes are displayed in Figure 5.17 for direct comparison. To visualize the different moldings in areas not visible from the outside, we added transparent views of the results in Figure 5.18.

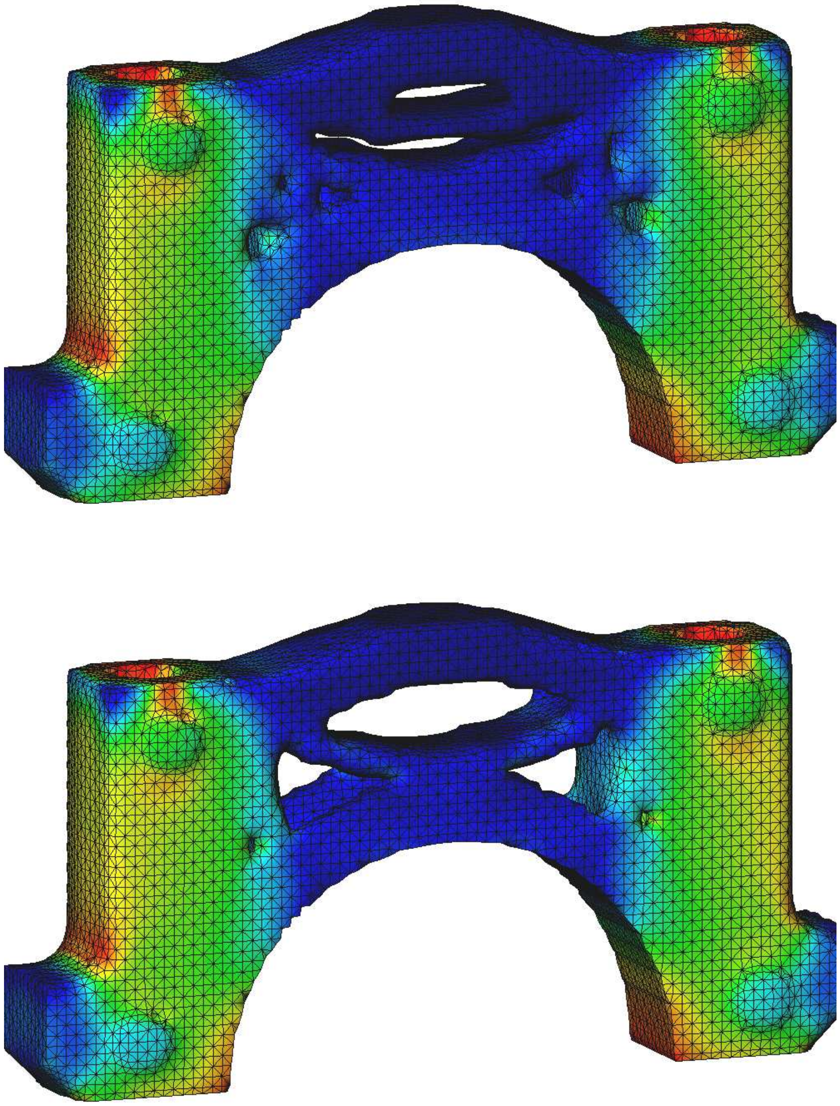


FIGURE 5.17. Top: Optimized structure after 6 iterations with a maximal volume reduction of 3 % per iteration. Bottom: Result after 3 iterations with 10 % volume reduction per iteration. The color bar is the same as in Figure 5.16.

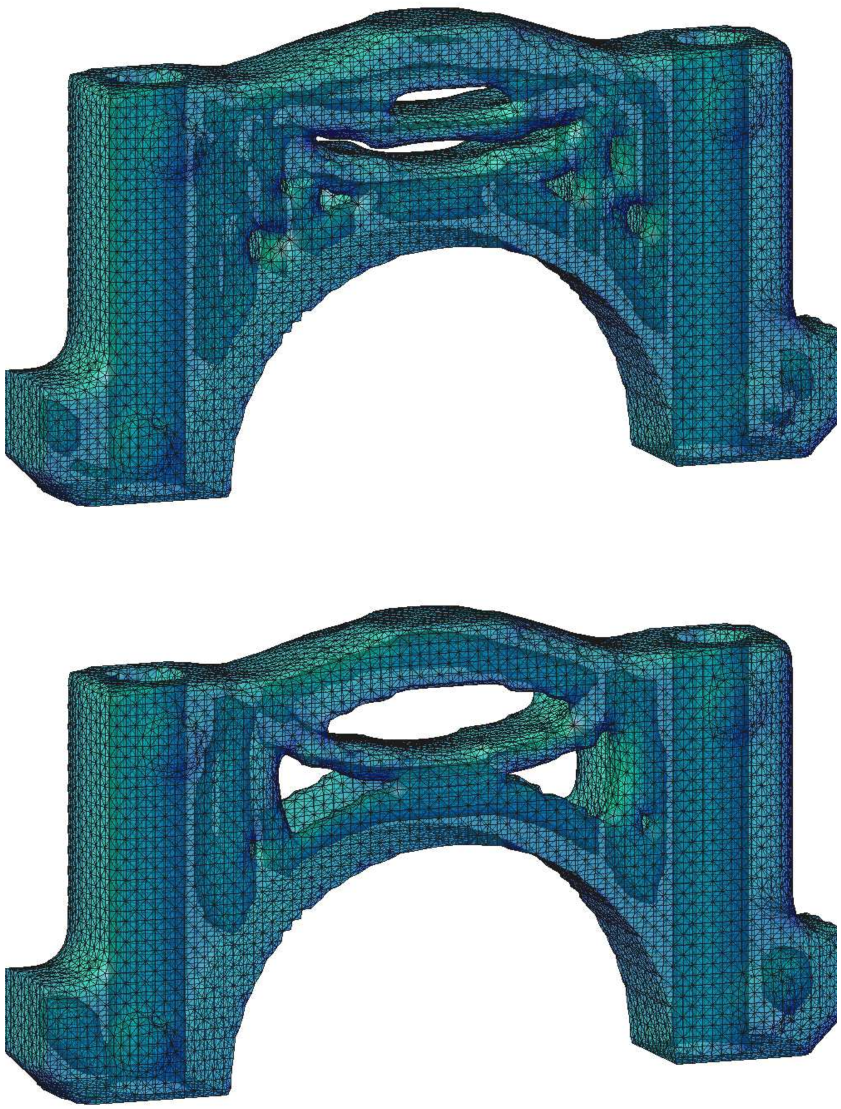


FIGURE 5.18. Transparent views of the optimized structures. Clearly, the smaller step size (top) lead to more filigree branches and holes in the structure's interior than the larger step size (bottom). In both cases there are holes in the feet of the bearing cap which are not connected to the exterior.

Summary and Future Work

In this thesis, we presented a new algorithm for the solution of 3D topology optimization problems. Contrary to other methods, our approach allows for the inclusion of a casting simulation in the optimization process. This can yield better optimization results, since unfavorable eigenstresses are considered when the sensitivity analyses are performed and the strength of structures is evaluated.

We used a twofold structural update and thereby satisfied the two main challenges imposed by the casting process simulation: On the one hand, the use of a level function for boundary representation ensures that the exact position of the boundary is known at any time. On the other hand, a level set based tetrahedral mesh generator constructs the complete FE model that is required in each iteration of the optimization process.

To make optimization of industrially relevant components possible, we focused on efficiency of the update process. We simplified the level set update significantly by representing holes as simple level set functions and using an addition operator for the update (section 4.3.1). With regard to the mesh generation and update part (chapter 3 and section 4.3.2, respectively), we ensured computational efficiency by combining an octree-based algorithm with clever re-use of the mesh from the previous iteration: Large parts of the computationally far more expensive mesh close to the structure's boundary are preserved, and only modified areas have to be remeshed. The structure's interior is meshed by precomputed configurations, which reduces computing time significantly. We showed that the resulting meshes are of high quality (section 3.5.2), which is the basis for reliable results.

The capabilities of our algorithm were demonstrated with the help of industrial-scale numerical examples (sections 5.2 and 5.3). Investigation of the distribution of computing time showed that our structural update requires significantly significantly less time than the solution of the elasticity problem, even though the FE solver was run on eight cores and the update on a single core. Thus, remeshing is no longer the bottleneck in topology optimization and can be successfully applied in each iteration.

Repeated generation of a complete FE model is made possible by automatic mapping of boundary condition surfaces (section 3.4). This feature is important for a smooth and efficient execution of the complete algorithm - if manual interaction were necessary, the process would take way too much time.

Future work can be split in two parts. Concerning theoretical aspects, we see a need for investigation of the effects of single parameters as for instance the width of the underlying Cartesian grid, the number of mesh smoothing iterations, or the amount of material that can be removed in a single iteration. Investigations in this direction might reveal criteria for an optimal and automated determination of parameter values that could lead to even better optimization results.

On the more practical side, some features which are relevant for foundries could be included. Here, we want to mention the so-called minimum member size, which prescribes a minimum thickness of for instance walls or trusses of cast parts and is needed to guarantee producability. In our opinion, the level set representation of the boundary provides a good basis for the implementation of this criterion.

Another attribute which is important for the actual casting process is the direction of removal from the mold. This can impose geometrical restrictions to the design space and should be considered to ensure that optimal designs can be manufactured.

Finally, a persistent parallelization of the code and especially the meshing process would further reduce computation time and make the algorithm even more attractive. Also here we see a good potential, since the deterministic nature of the surface mesh seems to facilitate effective distribution of computational effort among several processors.

Bibliography

[1] T. Akenine-Möller, *Fast 3D Triangle-Box Overlap Testing*, Department of Computer Engineering, Chalmers University of Technology, 2001.

[2] G. Allaire, F.D. Gournay, F. Jouve, A.-M. Toader, *Structural optimization using topological and shape sensitivity via a level set method*, Internal Report 555, Ecole Polytechnique, France, 2004.

[3] G. Allaire, F. Jouve, *A level-set method for vibration and multiple loads structural optimization*, Comput. Methods Appl. Mech. Engrg. 194, pp. 3269–3290, 2005.

[4] G. Allaire, F. Jouve, A.-M. Toader, *Structural optimization using sensitivity analysis and a level-set method*, J. Comp. Phys. 194, pp. 363–393, 2004.

[5] G. Allaire, R.V. Kohn, *Optimal design for minimum weight and compliance in plane stress using extremal microstructures*, European Journal of Mechanics/A 12, pp. 839–878, 1993.

[6] G. Allaire, R.V. Kohn, *Topology optimization and optimal shape design using homogenization*, In M.P. Bendsoe, C.A. Moa Soares (editors) Topology Design of Structures, volume 227 of NATO ASI Series, Series E, pp. 207–218, Kluwer, 1993.

[7] S. Amstutz, H. Andrä, *A new algorithm for topology optimization using a level-set method*, Journal of Computational Physics Volume 216, Issue 2, pp. 573–588, 2006.

[8] M. Bazaraa, H. Sherali, C. Shetty, *Nonlinear Programming Theory and Algorithms*, Wiley, 2006.

[9] M.P. Bendsoe, *Optimal shape design as a material distribution problem*, Structural Optimization 1, pp. 193–202, 1989.

[10] M.P. Bendsoe, N. Kikuchi, *Generating optimal topologies in structural design using a homogenisation method*, Computer Methods in Applied Mechanics and Engineering 71, pp. 197–224, 1988.

[11] M.P. Bendsoe, O. Sigmund, *Material interpolations in topology optimization*, Archive of Applied Mechanics 69, pp. 635–654, 1999.

[12] S.C. Brenner, L.R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, New York, 1994.

[13] M. Burger, *A framework for the construction of level-set methods for shape optimization and reconstruction*, Interfaces and Free Boundaries 5, pp. 301–329, 2003.

[14] M.Burger, B.Hackl, W.Ring, *Incorporating topological derivatives into level set methods*, J. Comp. Phys. 194, pp. 344–362, 2004.

[15] G.D. Cheng, N. Olhoff, *An investigation concerning optimal design of solid elastic plates*, International Journal of Solids and Structures 16, pp. 305–323, 1981.

[16] P.G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland Publishing Company, Amsterdam, 1978.

[17] M.C. Delfour, J.P. Zolésio, *Shapes and geometries. Analysis, differential calculus, and optimization*, SIAM, Philadelphia, 2001.

[18] A.R. Diaz, M.P. Bendoe, *Shape optimization of structures for multiple loading conditions using a homogenization method*, Structural Optimization 4, pp. 17–22, 1992.

[19] D.A. Field, *Qualitative measures for initial meshes*, International Journal For Numerical Methods In Engineering, Vol 47, pp. 887–906, John Wiley, 2000.

[20] J.E. Flaherty, *FINITE ELEMENT ANALYSIS*, Lecture Notes: Spring 2000.

[21] R. Fletcher, C.M. Reeves, *Function minimization by conjugate gradients*, Computer Journal 7:149, 1964.

[22] L.A. Freitag, C. Ollivier-Gooch, *Tetrahedral mesh improvement using swapping and smoothing*, Internat. J. Numer. Methods Engrg., 40(21) 3979–4002, 1997.

[23] P.J. Frey, P.-L. George, *Mesh Generation*, Hermes Science Publishing, Oxford & Paris, 2000.

[24] I. Fried, *Condition of finite element matrices generated from nonuniform meshes*, AIAA J. 10, pp. 219–221, 1972.

[25] M. Garland, P.S. Heckbert, *Simplifying Surfaces with Color and Texture using Quadric Error Metrics*, IEEE Visualization '98 Proceedings, pp.263–270, 1998.

[26] S. Garreau, P. Guillaume, M. Masmoudi, *The topological Asymptotic for PDE systems: the elasticity case*, SIAM J. Control Optim. 39(6), pp. 1756–1778, 2001.

[27] S. Gottschalk, M.C. Lin, D. Manocha, *OBBTree: A Hierarchical Structure for Rapid Interference Detection*, Computer Graphics (SIGGRAPH 96 Proceedings), pp. 171–180, 1996.

[28] F. Jarre, J. Stoer, *Optimierung*, Springer, Berlin [u.a.], 2004.

[29] T. Ju, F. Losasso, S. Schaefer, J. Warren, *Dual Contouring of Hermite Data*, Siggraph 2002, Computer Graphics Proceedings, pp.339–346, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2002.

[30] R.V. Kohn, G. Strang, *Optimal design and relaxation of variational problems*, Communications on Pure and Applied Mathematics 39, pp. 1–25, 139–182, 353–377, 1986.

[31] H.R. Lewis, L. Denenberg, *Data Structures and Their Algorithms*, Addison-Wesley Longman Publishing Co., 1997.

[32] W.E. Lorensen, H.E. Cline, *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, Computer Graphics (Proceedings of SIGGRAPH '87), Vol. 21, No. 4, pp. 163–169, 1987.

[33] M. Masmoudi, *The Toplogical Asymptotic*, Computational Methods for Control Applications, R. Glowinski, H. Kawarada and J. Periaux eds., GAKUTO Internat. Ser. Math. Sci. Appl. Vol. 16, pp. 53–72, 2001.

[34] I. Matei, *Development of Adequate Structure Optimization Methods for Foundries*, PhD thesis, Technische Universität Kaiserslautern, Dept. of Mathematics, in preparation.

[35] F. Murat, *Contre-exemples pour divers problmes o le contrle intervient dans les coefficients*, Ann. Mat. Pura Appl., 112, pp. 49–68, 1977.

[36] A.A. Novotny, R.A. Feijóo, E. Taroco, C. Padra, *Topological sensitivity analysis*, Computer Methods in Applied Mechanics and Engineering Volume 192, Issues 7-8, pp. 803–829, 2003.

[37] S. Osher, F. Santosa, *Level-set methods for optimization problems involving geometry and constraints: frequencies of a two-density inhomogeneous drum*, J. Comp. Phys. 171, pp. 272–288, 2001.

[38] S. Osher, J.A. Sethian, *Front propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comp. Phys. 78, pp. 12–49, 1988.

[39] S.J. Owen, *A Survey of Unstructured Mesh Generation Technology*, Proceedings, 7 th International Meshing Roundtable, pp. 239–267, 1998.

[40] V.N. Parthasarathy et al., *A comparison of tetrahedron quality measures*, Finite Elem. Anal. Des., 15: pp. 255–261, 1993.

[41] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, O.C. Zienkiewicz, *Finite Element Euler Computations in Three Dimensions*, International Journal For Numerical Methods in Engineering, John Wiley & Sons, Ltd, Vol 26, pp. 2135–2159, 1988.

[42] P.-O. Persson, *Mesh Generation for Implicit Geometries*, PhD thesis, Massachusetts Institute of Technology, Dept. of Mathematics, 2005.

[43] E. Polak, *Computational Methods in Optimization*, Academic Press, New York, 1971.

[44] J. Ruppert, *A Delaunay refinement algorithm for quality 2-dimensional mesh generation*, J. Algorithms, 18 (3), pp. 548–585, 1995.

[45] H. Samet, *Applications of Spacial Data Structures*, Addison-Wesley Publishing Company, New York, 1990.

[46] H. Samet, *The Quadtree and Related Hierarchical Data Structures*, ACM Comput. Surv., vol. 16, nr. 2, pp. 187–260, ACM Press, 1984.

[47] J. Schöberl, *NETGEN - An advancing front 2D/3D mesh generator based on abstract rules*, Computations in Visualization and Science 1, pp. 41–52, 1997.

[48] V. Schulz, H. Andrä, K. Schmidt, *Robuste Netzgenerierung zur μFE-Analyse mikrostrukturierter Materialien*, NAFEMS Magazin Nr. 02/2007, 7. Ausgabe, 2007.

[49] A. Schumacher, *Topologieoptimierung von Bauteilstrukturen unter Verwendung von Lochpositionierungskriterien*, PhD thesis, Universität-Gesamthochschule-Siegen, 1995.

[50] A. Schumacher, V.V. Kobolev, H.A. Eschenauer, *Bubble method for topol- ogy and shape optimization of structures*, Journal of structural optimization no. 8, pp. 42–51, 1994.

[51] J.A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1998..

[52] J.A. Sethian, A. Wiegmann, *Structural boundary design via level-set and immersed interface method*, J. Comp. Phys. 163, pp. 489–528, 2000.

[53] J.R. Shewchuk, *What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures*, Proceedings, 11th International Meshing Roundtable, Sandia National Laboratories, pp. 115–126, 2002.

[54] O. Sigmund, J. Petersson, *Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima.*, Struct. Multidisc. Optim., 16:68–75, 1998.

[55] J. Sokolowski, A. Zochowski, *On the topological derivative in shape optimization*, SIAM J. Control Optim. 37, pp. 1241–1272, 1999.

[56] J. Sokolowski, J.-P. Zolésio, *Introduction to shape optimization: shape sensitivity analysis*, Springer Series in Computational Mathematics vol. 10, Springer, Berlin, 1992.

[57] E.H. Spanier, *Algebraic Topology*, McGraw-Hill, New York, 1966.

[58] K. Suzuki, N. Kikuchi, *A homogenization method for shape and topology optimization*, Computer Methods in Applied Mechanics and Engineering 93, pp. 291–381, 1991.

[59] G. Taubin, *A signal processing approach to fair surface design*, SIGGRAPH 95 Proceedings, pp. 351–358, 1995.

[60] M.Y. Wang, X. Wang, D. Guo, *A level-set method for structural topology optimization*, Comput. Meth. Appl. Mech. Engrg. 192, pp. 227–246, 2003.

[61] R.E. White, *An Introduction to the Finite Element Method with Applications to Nonlinear Problems*, John Wiley and Sons, New York, 1985.

[62] G. Xu, *Convergent Discrete Laplace-Beltrami Operators over Triangular Surfaces*, 2004 Geometric Modeling and Processing, pp.195–204, 2004.

[63] M.A. Yerry, M.S. Shephard, *Three-Dimensional Mesh Generation by Modified Octree Technique*, International Journal for Numerical Methods in Engineering, vol 20, pp. 1965–1990, 1984.

[64] Y. Zhang, C. Bajaj, B.-S. Sohn, *Adaptive and quality 3D meshing from imaging data*, SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications, pp.286–291, ACM Press, Seattle, 2003.

[65] O.C. Zienkiewicz, R.L. Taylor, *The Finite-Element Method*, Volumes 1-3, fifth ed. Butterworth-Heinemann, Oxford, 2000.

Web-based Resources

[66] FE-DESIGN, , http://www.fe-design.de/tosca/tosca.html.

[67] Fraunhofer Institut für Techno- und Wirtschaftsmathematik, Competence Center High Performance Computing and Visualization, *DDFEM - parallel code for three-dimensional structure mechanics*, http://www.itwm.fraunhofer.de/en/hpc_parallelisierung_ddfem/ddfem/.

[68] MAGMA Giessereitechnologie GmbH, *MAGMA. Committed to casting excellence.*, http://www.magmasoft.de/.

[69] J. Schöberl, *NETGEN - automatic mesh generator*, http://www.hpfem.jku.at/netgen/ .

[70] H. Si, *A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, http://tetgen.berlios.de/.

[71] C.J. Stimpson, C.D. Ernst, P. Knupp, P.P. Pébay, D. Thompson, *The Verdict Library Reference Manual*, http://www.vtk.org/Wiki/images/6/6b/VerdictManual-revA.pdf.

CURRICULUM VITAE

14.03.1980	geboren in Speyer
1986 – 1990	Grundschule Heiligenstein
1990 – 1999	Gymnasium am Kaiserdom, Speyer
Okt. 1999	Beginn des Studiums im Fach Technomathematik mit Nebenfach Maschinenbau an der Technischen Universität Kaiserslautern
Juli 2001	Vordiplom in Technomathematik
Aug. 2002 – Jan. 2003	Studium der Mathematik an der Chalmers Tekniska Högskola in Göteborg im Rahmen des ECMI-Programms
Aug. 2004	Diplom in Technomathematik
Oct. 2004 – Aug. 2007	Stipendiat des Graduiertenkollegs Technomathematik an der Universität Kaiserslautern