

Régis Santos Thedin

**Topology Optimization using
Polyhedral Meshes**

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE ENGENHARIA MECÂNICA

**Programa de Pós-Graduação
em Engenharia Mecânica**

Rio de Janeiro
July 2014

Régis Santos Thedin

**Topology Optimization using
Polyhedral Meshes**

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Engenharia Mecânica of the Departamento de Engenharia Mecânica do Centro Técnico Científico da PUC–Rio, as partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica.

Advisor : Prof. Ivan Fábio Mota de Menezes
Co–Advisor : Prof. Marcio da Silveira Carvalho

Rio de Janeiro
July 2014



Régis Santos Thedin

Topology Optimization using Polyhedral Meshes

Dissertation presented to the Programa de Pós-Graduação em Engenharia Mecânica of the Departamento de Engenharia Mecânica do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica. Approved by the following commission:

Prof. Ivan Fábio Mota de Menezes

Advisor

Department of Mechanical Engineering — PUC-Rio

Prof. Marcio da Silveira Carvalho

Co-Advisor

Department of Mechanical Engineering — PUC-Rio

Prof. Eduardo Gomes Dutra do Carmo

COPPE — Federal University of Rio de Janeiro (UFRJ)

Prof. Glaucio Hermogenes Paulino

University of Illinois at Urbana-Champaign (UIUC)

Prof. Jose Eugenio Leal

Head of the Centro Técnico Científico — PUC-Rio

Rio de Janeiro, July 30th, 2014

All rights reserved. It is forbidden partial or complete reproduction without previous authorization of the university, the author and the advisor.

Régis Santos Thedin

Regis holds a bachelor of science degree from Pontifical Catholic University of Rio de Janeiro, obtained in July 2013. His final year thesis also focused on topology optimization.

Bibliographic data

Thedin, Régis S.

Topology Optimization using Polyhedral Meshes / Régis Santos Thedin; advisor: Ivan Fábio Mota de Menezes; co-advisor: Marcio da Silveira Carvalho. — 2014.

126 f.: il. (color); 30 cm

1. Dissertação (Mestrado em Engenharia Mecânica) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2014.

Inclui bibliografia

1. Engenharia Mecânica – Teses 2. Otimização topológica. 3. Malhas poliédricas. 4. Método dos elementos finitos. 5. Diagramas de Voronoi. 6. Métodos diretos e iterativos. I. Menezes, Ivan Fábio Mota de. II. Carvalho, Marcio da Silveira. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

To my parents, Ademir and Elenídia,
and brothers, Raphael and Rodolpho.

Acknowledgments

First and foremost, I'd like to express my profound gratitude to my advisor, Prof. Ivan Menezes. Through his kindness, patience, motivation, constant support and lots of encouragement I was able to conclude this work on time. I attribute much of this work's quality to our frequent meetings and his close, friendly supervision. My gratitude also extends to my co-adviser, Prof. Marcio Carvalho.

My sincere thanks also goes to Dr. Anderson Pereira. Due to almost-daily discussions and enlightenment from him I was able to achieve a better work. His guidance also helped me in several critical moments. Thank you!

I'm also very grateful to Prof. Glaucio Paulino for the collaboration opportunity with the University of Illinois at Urbana-Champaign. I also thank him for the suggestion of the topic and the feedback he gave throughout this work.

I also need to acknowledge some of my friends at PUC-Rio and from before PUC. In the past year they have been my source of daily strengths and my way out when stressed. They know who they are and how great friends they were.

I thank my family for their support and understanding when I had decided to pursue higher level education. Their encouragement has been essential to me.

Last, but most certainly not the least, I thank God for everything.

I acknowledge CNPq for the financial support and Tecgraf for the facilities and the welcoming environment.

Abstract

Thedin, Régis S.; Menezes, Ivan Fábio Mota de; Carvalho, Marcio da Silveira. **Topology Optimization using Polyhedral Meshes**. Rio de Janeiro, 2014. 126p. Dissertação de Mestrado — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Topology optimization has had an impact in various fields and has the potential to revolutionize several areas of engineering. This method can be implemented based on the finite element method, and there are several approaches of choice. When using an “element-based” approach, every finite element is a potential void or actual material, whereas every element in the domain is assigned to a constant design variable, namely, density. In an Eulerian setting, the obtained topology consists of a subset of initial elements. This approach, however, is subject to numerical instabilities such as one-node connections and rapid oscillations of solid and void material (the so-called checkerboard pattern). Undesirable designs might be obtained when standard low-order elements are used and no further regularization and/or restrictions methods are employed. Unstructured polyhedral meshes naturally address these issues and offer flexibility in discretizing non-Cartesian domains.

In this work we investigate topology optimization on polyhedra meshes through a mesh staggering approach. First, polyhedra meshes are generated based on the concept of centroidal Voronoi diagrams and further optimized for finite element computations. We show that the condition number of the associated system of equations can be improved by minimizing an energy function related to the element’s geometry. Given the mesh quality and problem size, different types of solvers provide different performances and thus both direct and iterative solvers are addressed. Second, polyhedrons are decomposed into tetrahedrons by a tailored embedding algorithm. The polyhedra discretization carries the design variable and a tetrahedra sub-discretization is nested within the polyhedra for finite element analysis. The modular framework decouples analysis and optimization routines and variables, which is promising for software enhancement and for achieving high fidelity solutions. Fields such as displacement and design variables are linked through a mapping. The proposed mapping-based framework provides a general approach to solve three-dimensional topology optimization problems using polyhedrons, which has the potential to be explored in applications beyond the scope of the

present work. Finally, the capabilities of the framework are evaluated through several examples, which demonstrate the features and potential of the proposed approach.

Keywords

Topology optimization; Polyhedral meshes; Finite element method; Voronoi diagrams; Direct and iterative solvers.

Resumo

Thedin, Régis S.; Menezes, Ivan Fábio Mota de; Carvalho, Marcio da Silveira. **Otimização Topológica usando Malhas Poliédricas**. Rio de Janeiro, 2014. 126p. Dissertação de Mestrado — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

A otimização topológica tem se desenvolvido bastante e possui potencial para revolucionar diversas áreas da engenharia. Este método pode ser implementado a partir de diferentes abordagens, tendo como base o Método dos Elementos Finitos. Ao se utilizar uma abordagem baseada no elemento, potencialmente, cada elemento finito pode se tornar um vazio ou um sólido, e a cada elemento do domínio é atribuído uma variável de projeto, constante, denominada densidade. Do ponto de vista Euleriano, a topologia obtida é um subconjunto dos elementos iniciais. No entanto, tal abordagem está sujeita a instabilidades numéricas, tais como conexões de um nó e rápidas oscilações de materiais do tipo sólido-vazio (conhecidas como instabilidade de tabuleiro). Projetos indesejáveis podem ser obtidos quando elementos de baixa ordem são utilizados e métodos de regularização e/ou restrição não são aplicados. Malhas poliédricas não estruturadas naturalmente resolvem esses problemas e oferecem maior flexibilidade na discretização de domínios não Cartesianos.

Neste trabalho investigamos a otimização topológica em malhas poliédricas por meio de um acoplamento entre malhas. Primeiramente, as malhas poliédricas são geradas com base no conceito de diagramas centroidais de Voronoi e posteriormente otimizadas para uso em análises de elementos finitos. Demonstramos que o número de condicionamento do sistema de equações associado pode ser melhorado ao se minimizar uma função de energia relacionada com a geometria dos elementos. Dada a qualidade da malha e o tamanho do problema, diferentes tipos de resolvedores de sistemas de equações lineares apresentam diferentes desempenhos e, portanto, ambos os resolvedores diretos e iterativos são abordados. Em seguida, os poliedros são decompostos em tetraedros por um algoritmo específico de acoplamento entre as malhas. A discretização em poliedros é responsável pelas variáveis de projeto enquanto a malha tetraédrica, obtida pela subdiscretização da poliédrica, é utilizada nas análises via método dos elementos finitos. A estrutura modular, que separa as rotinas e as variáveis usadas nas análises de deslocamentos das usadas no processo de otimização, tem se mostrado promissora tanto na melhoria da eficiência computacional como na qualidade das soluções que foram obtidas neste

trabalho. Os campos de deslocamentos e as variáveis de projeto são relacionados por meio de um mapeamento. A arquitetura computacional proposta oferece uma abordagem genérica para a solução de problemas tridimensionais de otimização topológica usando poliedros, com potencial para ser explorada em outras aplicações que vão além do escopo deste trabalho. Finalmente, são apresentados diversos exemplos que demonstram os recursos e o potencial da abordagem proposta.

Palavras-chave

Otimização topológica; Malhas poliédricas; Método dos elementos finitos; Diagramas de Voronoi; Métodos diretos e iterativos.

Contents

List of Figures	12
List of Tables	14
List of Symbols	15
1 Introduction	20
1.1 Motivation	20
1.2 Objectives and Scope of Current Work	23
1.3 Dissertation Outline	25
2 Topology Optimization	26
2.1 Introduction	26
2.2 Formulation and Problem Statement	28
2.3 Density-based Methods	34
2.4 Other Methods	35
2.5 Optimizer	37
2.6 Regularization Techniques	40
2.7 Discretization Choice	42
2.8 Concluding Remarks	44
3 Embedding of Elements	46
3.1 Motivation	46
3.2 The Embedding Approach	48
3.3 Displacement Field	49
3.4 Density Field	52
3.5 “Super-elements”	52
3.6 Mapping Between Fields	53
3.7 Concluding Remarks	57
4 Mesh Generation	58
4.1 Implicit Representation	59
4.2 Delaunay Triangulation and Voronoi Tesselations	60
4.3 Mesh Optimization Procedure	63
4.4 Results and Concluding Remarks	69
5 Solving the System of Equations	74
5.1 Introduction	75
5.2 Direct Sparse Solvers	76
5.3 Iterative Solvers	84
5.4 Packages	88

5.5	Performance Results and Conclusions	91
6	Numerical Results	100
6.1	Three-dimensional Validation	100
6.2	Benchmark Problem	101
6.3	Edge Supported Cantilever Beam	108
6.4	Michell-like Domain Subject to a Torsional Load	108
6.5	Shear Loaded Thin Disk	111
6.6	Beam Subject to a Torsional Load	112
6.7	Final Remarks	113
7	Conclusions and Extensions	114
7.1	Suggestions for Future Work	116
	Bibliography	118

List of Figures

1.1	Example of numerical instabilities in conventional finite elements	21
1.2	Motivation of the super-element approach: Displacement mesh and a rotated density mesh together to generate a modified Q4 element	22
1.3	Higher resolution results with modified elements	22
1.4	Solution for the Michell problem using polyhedrons and quadratic triangle elements	24
2.1	Categories of structural optimization	27
3.1	Multiresolution topology optimization scheme	47
3.2	Higher resolution topologies obtained using embedding technique on structured meshes	48
3.3	Two-level mesh representation	48
3.4	The embedding process on a simple mesh	50
3.5	Tetrahedron nomenclature	50
3.6	Meshes distinction	53
3.7	Element association between meshes	54
3.8	Mapping between elements	55
3.9	Map between the two discretizations	56
4.1	Geometric representation of a domain	60
4.2	Delaunay triangulation and corresponding Voronoi diagram	61
4.3	Initial seeds distributions and their CVTs	62
4.4	Two-dimensional CVT mesh, 50×50 domain	63
4.5	Two-dimensional PolyMesher approach dealing with small edges	64
4.6	Origin of a small Voronoi edge	64
4.7	Meshing optimization procedure	68
4.8	Circumcenters are shifted towards the incenters	69
4.9	Histograms of edges lengths for the unitary cube	71
5.1	Runtime breakdown of a typical topology optimization problem for different meshes size	74
5.2	Example of post-embedding numbering	78
5.3	Matrix pattern after reordering	79
5.4	Special embedding performed on regular hexahedron, with no new node creation	92
5.5	Runtimes for each main part for the PARDISO package	93
5.6	Runtimes for each main part for the UMFPACK package	93
5.7	Number of PCG iteration with different preconditioner, as well as condition number at each optimization iteration	96
5.8	Relative increase in the number of PCG iterations needed in a typical problem using unstructured polyhedral meshes	96
5.9	Ersatz parameter ε influence in the conditioning of the stiffness matrix as the topology optimization evolves	97

5.10	Performance of different solver in the solution of differently sized problems	98
6.1	Michell beam problem: domain, loads, boundary conditions and known analytical solution	100
6.2	Michell beam problem converged topology	101
6.3	Cantilever beam subject to a transverse load problem	102
6.4	Converged topology for the beam problem with volume fraction constraint of 8.5%	102
6.5	Converged topology for the beam problem: Alternative views	103
6.6	Compliance convergence history for differently-sized fixed-beam problem	107
6.7	Cantilever Beam supported by two edges	108
6.8	Three-dimensional Michell-like domain subject to a torsional load	109
6.9	Michell original problem of the torsion ball	110
6.10	Thin disk subject to equidistant shear loads problem	111
6.11	Box under torsional loads domain	112
6.12	Other results for the box under torsional load	113

List of Tables

4.1	Mesh statistics for the unitary cube domain, $\Omega = (0, 1)^3$	70
4.2	Effectiveness in removing small edges	72
4.3	Condition number associated with the stiffness matrix of the linear elasticity problem	73
5.1	Mesh information	94
5.2	Number of iterations for the solution of $\mathbf{Ax} = \mathbf{b}$ using the PCG solver	95
5.3	Average number of iterations for the solution of the first 10 iterations of a topology optimization problem using the PCG solver	95
6.1	Details of the meshes used for the construction of Table 6.2	103
6.2	Filtering and embedding investigation results	104
6.3	Comparison between brick and polytopes elements	106

List of Symbols

Upper-case Roman

\hat{H}	Weighting factor for filters
\mathbf{B}	Strain-displacement matrix
\mathbf{C}	Material tensor matrix
\mathbf{C}^0	Material tensor matrix for $\rho = 1$
\mathbf{F}	Global load vector
\mathbf{K}	Global stiffness matrix
$\mathbf{K}^{(e)}$	Element stiffness matrix
\mathbf{L}	Lower triangular matrix
\mathbf{N}	Shape function matrix
\mathbf{P}	Mapping matrix
\mathbf{U}	Global displacement vector; Upper triangular matrix
A_i	Area
D	Displacement mesh
E	Young's modulus
$E(\mathbf{x})$	Energy function
E^0	Young's modulus for $\rho = 1$
G	Constraint function
M	Density mesh
N	Shape function
P	Nodal load
R	Radius of the circumsphere
S_i	Set of FE elements sharing density element i
V	Volume
V_S	Specified maximum volume
J	Jacobian

Lower-case Roman

$\mathbf{u}^{(e)}(\mathbf{x})$	Displacement field
--------------------------------	--------------------

\mathbf{x}	Location of a point in the domain
c	Compliance of the design
$d(\mathbf{x})$	Distance function
f	Objective function, displacement element
h	Step size in the mesh optimization algorithm
m	Move limit in the OC method
p	SIMP penalty parameter
r	Radius of the insphere
r_{\min}	Radius of the filter
u, v, w	Components x, y, z of the displacement
x, y, z	Cartesians coordinates
d	Density element

Upper-case Greek

Φ	PDE function
Ω	Design domain
Ω_S	Optimized structure domain
$\partial\Omega$	Domain boundary

Lower-case Greek

δ	Kronecker delta
η	Damping parameter in the OC method
γ	Shear strain
κ	Condition number
λ	Lagrange multiplier; eigenfrequency
σ	Stress vector
ε	Strain vector
ν	Poisson's ratio
ϕ	Phase field; angle in Michel's sphere problem
ϕ_a	Barycentric shape functions
ρ	Density

ρ_{min}	Density lower bound
τ	Shear stress
ε	Ersatz parameter, minimum step size in the mesh optimization algorithm

Other Symbols

(e)	Iteration e ; element-wise
$(\)^*$	Optimal point
\mathbb{R}^n	Set of real numbers in dimension n

Acronyms

2D	Two-dimensional
3D	Three-dimensional
BC	Boundary Condition(s)
BEM	Boundary Element Method
BESO	Bi-directional ESO
BiCGSTAB	Biconjugate Gradient stabilized
BLAS	Basic Linear Algebra Subprograms
CG	Conjugated Gradient
CPU	Core Processing Unit
CVT	Centroidal Voronoi Tessellation
DOF	Degree(s) of Freedom
DT	Delaunay Triangulation
ESO	Evolutionary Structural Optimization
FE	Finite Element
FEA	Finite Element Analysis
FEM	Finite Element Method
FVM	Finite Volume Method
GPU	Graphics Processing Unit
I/O	Input/Output
IC	Incomplete Cholesky preconditioner
LAPACK	Linear Algebra Package

MBB	Messerschmitt-Bölkow-Blohm
MKL	Math Kernel Library
MMA	Method of Moving Asymptotes
MTOP	Multi-resolution Topology Optimization scheme
NAND	Nested Analysis and Design
OC	Optimality Criteria
PARDISO	Parallel Direct Sparse Solver Interface
PCG	Preconditioned Conjugated Gradient
PDE	Partial Differential Equation
Q4	Quadrilateral element with 4 nodes
RAM	Random Access Memory
RAMP	Rational Approximation of Material Properties
RCM	Reverse Cuthill-McKee
SAND	Simultaneous Analysis and Design
SAO	Sequential Approximate Optimization
SIMP	Solid Isotropic Material with Penalization
SLP	Sequential Linear Programming
SQP	Sequential Quadratic Programming
T3	Triangular element with 3 nodes
T6	Triangular element with 6 nodes
UMFPACK	Unsymmetric-pattern Multifrontal Package
VEM	Virtual Element Method

*“I don't know anything, but I do know that
everything is interesting if you go into it
deeply enough”*

Richard Feynman, *1965 Physics Nobel*.

1

Introduction

The search for optimal design under a variety of specified conditions is an important and challenging subject. The structural optimization field has a long history, beginning in the early twentieth century with the Michell's research on optimal truss layouts [65]. Starting in the past century, the efficient use of materials became increasingly important in many different settings. In this context, structural topology optimization is a current growing field with many applications in several different industries, such as aerospace [59], automotive [18], civil [9] and medical [92].

In its most general setting, topology optimization of continuum structures consist of a determination, for every point in design space, if there is material in that point or not [11]. In other words, the goal is to find the material distribution of the system. The unknown is the topology of the structure.

The complexity of typical problems in engineering generally requires the use of unstructured meshes in order to accurately discretize the domain and capture the boundary condition, therefore ensuring reliable solutions. Polyhedral elements provides not only the flexibility in discretizing complex domain but also lead to optimal topologies that are not biased by the domain discretization.

This dissertation investigates topology optimization on polyhedral meshes. We present a technique based on staggering of elements. We employ a sub-discretization of the polyhedral mesh in order to perform finite element analysis on common elements, namely, tetrahedrons. The final topology is given as a subset of the initial polyhedral discretization.

We investigate in detail important steps in the topology optimization approach, from three-dimensional polyhedral mesh generation to the solution of the associated system of linear equations.

1.1

Motivation

In some practical engineering problems, two-dimensional approximations may give adequate and more economical numerical models. In the field of

topology optimization, however, three-dimensional problems not always can be approximated by two-dimensional representations. Moreover, more attention had been given to three-dimensional investigations recently in the topology optimization community.

In an “element-based” approach, the problem parametrization and its approximate solution are closely linked to the discretization of the domain. Each element in the domain is assigned to a constant design variable. However, such approach limits the resolution of the topology and precision of response, specially if regular elements are used. The final structure connecting the imposed boundary conditions and external loads is found as a subset of all the elements of the initially chosen set of finite elements. Thus the characteristics of the element used in domain discretization plays a major role in topology optimization problems. Checkerboard patterns are common when low-order Lagrangian elements are used and no further regularization technique is employed. Checkerboard instability consists in rapid oscillation of solid and void material. Additionally, structured meshes are subject to mesh biased solutions (see Fig. 1.1). Such numerical instabilities, detailed in the work of Sigmund [84] and Díaz [28], may, eventually, restrict the areas in which topology optimization can be used.

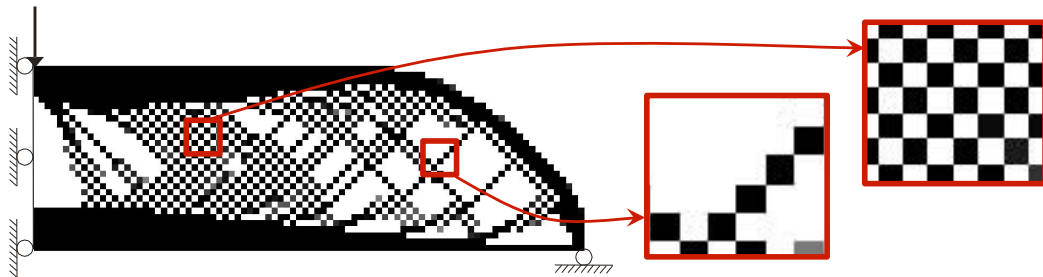


Figure 1.1: Example of numerical instabilities in conventional finite elements. MBB problem [75] using linear quad elements, showing one-node connections and checkerboard pattern.

In order to avoid numerical instabilities using the element-based approach, additional restrictions should be applied to the design space or higher order finite elements should be used. In this setting, polygonal finite elements have been studied [100]. Talischi et al. [98] have shown that stable solutions can be achieved using polygonal finite elements.

Authors such as Sukumar [90] and Talischi et al. [98] have studied convex two-dimensional polygonal finite elements. Fully unstructured meshes reduce the influence of the simplex geometry on topology optimization solutions. This is accomplished by means of polygonal meshes based on Voronoi tessellations, which in addition to possessing higher degree of geometric isotropy

allow for flexibility in discretization. Due to its geometry, such elements do not allow checkerboard and/or one-node connections. Polygonal 2D finite elements showed to be a reliable element not only in topology optimization problems. Other applications of polygonal finite elements includes fracture mechanics [89], biomechanics and coupled electromechanical problems [53]. Additionally, Gain [35] demonstrated the capabilities of unstructured polyhedral meshes in providing pathology-free topologies.

In the present work we deal with unstructured polyhedral-based meshes for compliance minimization topology optimization problems. We present an algorithm based on a staggering of elements, containing two separate fields.

The separation of fields have been investigated in the concept of multiple meshes by Paulino et al. [69] and Nguyen et al. [67]. In [69], a rotated Q4 design variable mesh and a Q4 displacement mesh are superpositioned (Fig. 1.2 and 1.3); whereas in [67], a design variable mesh is nested within a displacement mesh. Such investigations yielded higher resolution designs.

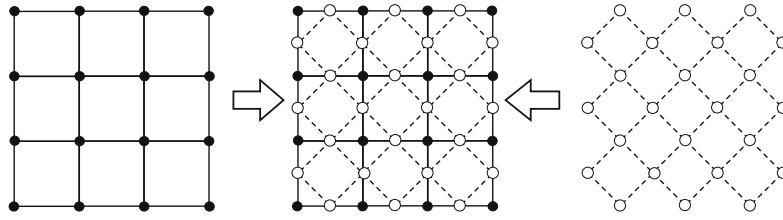


Figure 1.2: Motivation of the super-element approach: Displacement mesh (left) and a rotated density mesh (right) together to generate a modified Q4 element (center) [69].



Figure 1.3: Modified elements can be used to achieve higher resolution results with same mesh size. In this example is shown a modified Q4 element [69].

Thus this work aims in solving three dimensional topology optimization problems using special elements in the interest of avoiding numerical instabilities commonly found in standard three dimensional elements, such as the 8-node bricks. These special elements, called here *super-elements*, are constructed embedding tetrahedrons within polyhedrons. Tetrahedrons are used for finite element analysis and polyhedrons carry the design variables. The resulting topology is found to be a subset of the initial polyhedral mesh. The

inherently unstructured nature of Voronoi-based polyhedral meshes is valuable for topology optimization. Polytope meshes alleviate the problem of one-node or one-edge (sometimes referred to as “hinges”) connection between elements in the final topology.

1.2

Objectives and Scope of Current Work

This work deals with arbitrary polyhedrons for representation of the design domain. Topology optimization is carried out using a density-based method, in which the density is updated on the polyhedral elements, whereas displacement analysis occurs in a sub-discretization, consisting of tetrahedral elements.

Three-dimensional extension of shape functions still presents restriction on the topology of admissible elements (such as convexity and maximum valence count). The development of polygonal interpolants has its roots in the work of Wachspress [103], who proposed rational basis functions on polygonal elements. The construction of polygonal interpolants have been investigated by several authors [88, 90, 103], but reliable extension to three-dimensions is still under investigation by the finite element community (see section 2.7.1). Although some recent work is at an advanced stage, the method is still not well-established [63]. Finite element computation of arbitrary polyhedrons using their shape functions is still a novel technique.

In this setting, this dissertation investigates the decomposition of polyhedra into tetrahedra and subsequent parametric mapping between them. Thereby, finite element analysis is carried out in well-known elements, namely, tetrahedrons. The use of a second discretization is also motivated by multi-resolution approaches, outlined in the previous section.

In order to deal with some of the issues presented, we introduce a technique that involves the staggering of elements, using two separate computational domains within the same physical domain, with decoupled routines and variables. A framework based on a mapping between design and displacement variables is developed. The polyhedra-based mesh is decomposed into tetrahedral elements by means of an embedding algorithm. Polyhedrons are used to carry the design variable and represent the final topology, whereas tetrahedrons are used to carry out finite element analysis. Both fields are linked through a mapping. We called this approach *embedding of elements*.

In most topology optimization approaches available in the literature, regularization schemes are employed with the objective of obtaining mesh independent solutions. However, regularizations schemes often smooth the

final density field, resulting in a design that is not solid/void, which requires further care. Filters are commonly used as a regularization technique, and are usually employed when a topology with minimum (or maximum) length scale is required. Minimum length scale filters ensure the structure does not converge to different topologies when using finer meshes, which defines a mesh independent formulation. Here we investigate optimal solutions with no imposed minimum length scale, seeking the analytical solution. As mentioned, discretizations with commonly used low-order elements exhibit undesired features when no regularization is employed and they frequently display checkerboard patterns.

In usual structured discretization, the orientation of the elements may influence the geometry of the final structure. Discretizations that present favored direction may lead to biased non-optimal solution and may exhibit one-node connections. Polyhedral elements naturally address these issue (see Fig. 1.4).

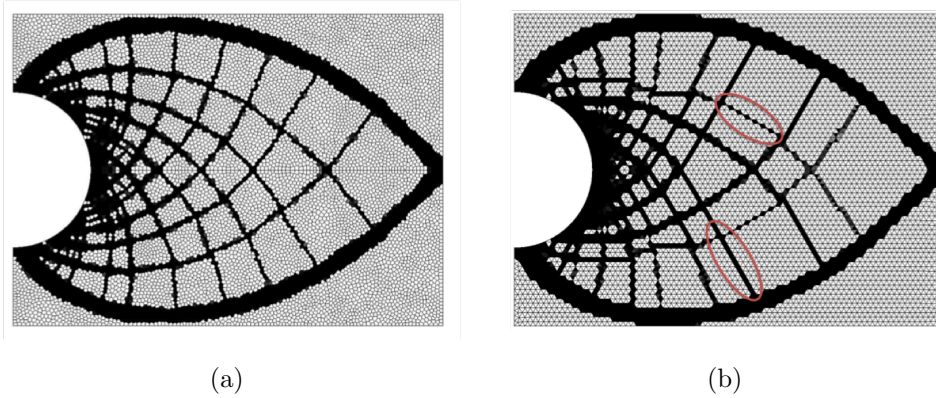


Figure 1.4: Solution for the Michell problem [93] using (a) 10,000 polyhedrons; (b) 10,220 quadratic triangle element [98]. Note biased structural members and node to node connection.

The polyhedral discretization of the domain is carried out based on the concept of Voronoi diagrams and a Centroidal Voronoi Tessellation (CVT) is obtained [99, 101]. However, here we seek to further optimize this mesh, in a sense that small edges are penalized and no embedded tetrahedron is greatly deformed with respect to each other. Small edges are effectively penalized by decreasing a carefully designed energy function related to the element's geometry. We aim in reducing the condition number of the stiffness matrix associated with the linear elasticity problem, which, in turn, yields better performance on iterative solvers. Different aspects related to the numerical solution of the system of equations are also addressed in this work. Both direct and iterative (including the use of preconditioning) solvers are discussed and their performance in solving typical topology optimization problems is

evaluated.

The examples investigated here are restricted to compliance minimization. We note that it is possible to expand to different objective functions, as well as adaptations to handle different physics (such as fluid flow problems). Results from structural topology optimization using the embedding approach obtained in this work are promising in achieving high fidelity solutions. Our framework achieved solutions that matches analytical results and consistently indicates presence of features (such as orthogonal structural members) that are expected in such optimal shapes.

Our embedding scheme is quite general and expandable. Its modular architecture completely separates optimization routines from specific choice of optimizer. Additionally, regularization methods can be applied with very few modification, if a topology with minimum length scale is desired.

Concluding, this work consists in an extension of the PolyTop framework [100], developed by Talischi et al. The original MATLAB code has been rewritten in C++ [29]. In this work we expanded in order to consider three-dimensional problem and incorporate the mapping-based embedding algorithm.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows: Chapter 2 introduces the topology optimization discipline, current status, issues and set the context of the present work. In Chapter 3, we present the embedding of elements algorithm and how it is employed towards the solution of a topology optimization problem. Chapter 4 describes the polyhedral meshing approach, issues, mesh optimization procedures and algorithms, and provides results for efficiency assessment of the mesh generator. Chapter 5 deals with performance issues regarding the solution of the system of equations that arises from the finite element discretization. Different types of solvers, numerical factorization, reordering algorithms and preconditioners are discussed. Numerical results obtained with the developed framework are finally presented in Chapter 6, including comparisons with standard topology optimization formulations and commonly used elements. Final conclusions are summarized in Chapter 7, as well as suggestions for future work.

2

Topology Optimization

Structural optimization is a mathematical and computational approach that optimizes a measure of performance while satisfying a set of constraints. Topology optimization is one of the branches of structural optimization: it is a method for finding the material distribution that generates the optimal layout structure in such a way that maximizes (or minimizes) a certain objective function. Topology optimization, differently from shape optimization, allows the introduction and/or elimination of holes in the structure, amounting to a true optimal design tool.

This Chapter gives the reader a review of topology optimization theory and methods, and set the context for the present work.

2.1

Introduction

A general optimization problem can be represented in the following way: Given a function $f: A \rightarrow \mathbb{R}$, in which \mathbb{R} represents the set of real numbers, find an element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A (minimization problem), or such that $f(x_0) \geq f(x)$ for all x in A (maximization problem).

Usually, A is a subset of the Euclidean space \mathbb{R}^n , commonly specified by a set of constraints, equalities and inequalities, that members of A must satisfy. The domain of A is called *search space* or *choice set*, whereas elements of A are called *candidate solutions* or *feasible solutions*. The function f is called *objective function*. A feasible solution set that minimizes/maximizes the objective function is called optimal solution

One common objective function f is the minimization of mass (i.e., volume) of the structure, while satisfying equilibrium equations and others geometrical constraints. In structural optimization problems, the final shape, or geometry, of the structure is the unknown. Structural optimization can be divided into three categories of problems: *sizing*, *shape* and *topology* optimization [19].

Each of them addresses different aspects of structural design. In a *sizing* problem, the objective may be to find the optimal thickness distribution of a

plate or the optimal members area in a truss structure. A specified physical quantity is minimized, while some constraints are satisfied. The main feature of a sizing problem is that the domain of the design is known *a priori* and is fixed throughout the optimization process. On the other hand, the objective in a *shape* optimization problem is to find the optimum shape of this domain, that is, the shape problem is defined on a domain which is now the design variable. The *topology* optimization problems are more general and involves the determination of features such as the number, location and shape of holes and the connectivity of the domain [11]. The concept of each structural optimization discipline is illustrated in Fig. 2.1.

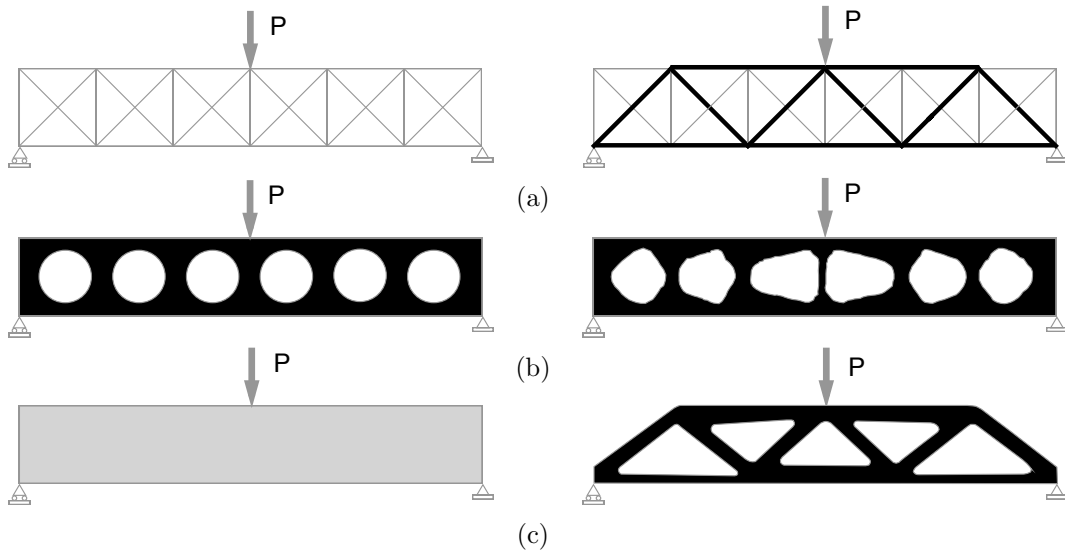


Figure 2.1: Categories of structural optimization. (a) Sizing optimization; (b) shape optimization; and (c) topology optimization (with imposed minimum length scale; not actual Michell's optimal solution). Initial problem shown on the left and solution on the right. From [11].

Differently from other methods, finding an optimal topology means finding the connectedness, shape, number and form of holes in the structure. Additionally, the design domain is not fixed a priori and the topology optimization formulation allows the introduction and elimination of holes (essentially voids), providing flexibility in converging to the optimal solution.

A common constraint in topology optimization is the amount of material that may be removed. There are a number of objectives functions used in topology optimization, but here we focus our attention in compliance minimization problems. In such problems, the objective function not only depends on the shape explicitly but also on the physical response of the system (e.g. deformation). It is clear the presence of two competing effects: we're trying to minimize the external work done by the applied load, while having a

constraint on the volume of the structure, minimizing the amount of material used. Minimizing the compliance amounts to maximizing the stiffness.

2.2

Formulation and Problem Statement

A general topology optimization problem may be stated as follows: find the material distribution ρ for every point \mathbf{x} , resulting in a subdomain Ω_S within a fixed domain Ω that minimizes the objective function f , subject to a prescribed upper bound on the volume of the structure V_S and to m others constraints $G_i \leq 0$, $i = 1, \dots, m$. Mathematically,

$$\begin{aligned} \min_{\rho} : \quad & f = f(\mathbf{U}(\rho(\mathbf{x})), \rho(\mathbf{x})) \\ \text{s.t.} : \quad & G_0(\rho) = \int_{\Omega} \rho(\mathbf{x}) \, dV - V_S \leq 0 \\ & G_i(\mathbf{U}(\rho), \rho) \leq 0, \quad i = 1, \dots, m, \end{aligned} \quad (2.1)$$

where we note \mathbf{U} as the displacement vector. We emphasize that the objective function depends on the response of the system as well as on the actual shape.

We employ a continuous parametrization of the domain by means of a density function ρ :

$$\rho(\mathbf{x}) = \begin{cases} 0, & \text{if void} \\ 1, & \text{if structural member.} \end{cases} \quad (2.2)$$

The optimization problem must also satisfy the binary nature of the density, defined in (2.2) for every point \mathbf{x} in extended domain Ω . The continuous problem (2.1) is usually solved by discretizing the domain Ω into N elements (or nodal) design variables¹, usually by the finite element method. Using an element-based approach and assigning a density value to each element in the domain, Eq. (2.1) can be rewritten as:

$$\begin{aligned} \min_{\boldsymbol{\rho}} : \quad & f(\mathbf{U}(\boldsymbol{\rho}), \boldsymbol{\rho}) \\ \text{s.t.} : \quad & G_0(\boldsymbol{\rho}) = \sum_{i=1}^N \rho_i v_i - V_S \leq 0 \\ & G_j(\mathbf{U}(\boldsymbol{\rho}), \boldsymbol{\rho}) \leq 0, \quad j = 1, \dots, m \\ & \rho_i = 0 \text{ or } 1, \quad i = 1, \dots, N, \end{aligned} \quad (2.3)$$

where $\boldsymbol{\rho}$ is now a N -sized design variable vector.

It is well known that the problem stated in Eqs. (2.1–2.3) presents fundamental difficulties: it is ill-posed and lack solutions in general [84]. For

¹For now, the design variable can be thought of being, essentially, the density.

several types of problems, closedness of the design space is also an issue, as ever increasing the number of holes will decrease the objective function. Shapes with fine features, i.e., fine structural members, are favored, leading to a non-convergent solution that exhibits rapid oscillations, making no sense from a physical point of view. This issue can also be thought in a way that more holes can arbitrarily appear in finer discretization, defining the mesh-dependency phenomena. More holes usually refers to physically inadmissible solutions. That is, finer discretizations lead to finer structural members. Regularization and restriction methods are generally employed to address these issues.

In order to efficiently use gradient based optimization algorithms (the popular optimality criteria, for example), a continuous density is desired. The set of admissible densities is relaxed to allow the appearance of intermediate values, ranging from 0 to 1, leading to some sort of “grey regions”:

$$0 \leq \rho \leq 1. \quad (2.4)$$

In density-based methods, another critical aspect is the selection of interpolation function and penalization technique that express the physical property of the problem as a function of the design variables. These methods are also used to ensure the solution is directed towards solid/void results. A commonly used approach in topology optimization is the *Solid Isotropic Material with Penalization* (SIMP) material model, first described by Zhou and Rozvany [108] and extensively analyzed by Bendsøe [13] and references therein. In the SIMP method, the design variables are penalized with a power law (with finite value) and multiplied onto physical quantities such as material stiffness. Hence, the material density is embedded into the material tensor. The relationship between the density $\rho(\mathbf{x})$ and the material tensor $\mathbf{C}_{ijkl}(\mathbf{x})$ in the equilibrium analysis is written as:

$$\mathbf{C}_{ijkl}(\mathbf{x}) = [\rho(\mathbf{x})]^p \mathbf{C}_{ijkl}^0, \quad (2.5)$$

where p is the SIMP penalization factor ($p \geq 1$) and \mathbf{C}_{ijkl}^0 is the material tensor (constitutive matrix) for the solid phase, corresponding to $\rho = 1$. The material tensor is function of only two material properties (for an isotropic material): the Poisson ratio and the Young’s modulus (see Eq. 2.17). By imposing the penalization over the densities values, intermediate values of stiffnesses are effectively penalized, and the solution converges to the optimal shape, with grey regions less pronounced [11].

One can note that using the density bounds from Eq. 2.4, the SIMP model gives $\mathbf{C}_{ijkl}(\rho = 1) = \mathbf{C}_{ijkl}^0$, which is correct and makes sense from a physical point of view. The value of zero density, however, is often associated

with numerical issues and singular matrices when performing FEA. Therefore, $\mathbf{C}_{ijkl}(\rho = 0) = \mathbf{0}$ leads to numerical problems. To avoid such problems, one option is to impose a positive lower bound on the density:

$$0 < \rho_{\min} \leq \rho \leq 1,$$

or, we can also overcome this issue and ensure well-posedness of the governing equations by replacing regions with compliant material of stiffness $\varepsilon \mathbf{C}_{ijkl}$. The property ε is usually referred to as Ersatz parameter in the level-set literature and its value is such that avoids numerical zero, $0 < \varepsilon \ll 1$. Rewriting Eq. (2.5):

$$\mathbf{C}_{ijkl} = [(1 - \varepsilon)\rho^p + \varepsilon] \mathbf{C}_{ijkl}^0. \quad (2.6)$$

Penalized topology optimization problems are non-convex, often having a large number of local minima. To convey this issue and increase the chance of converging to the global optimal solution, continuation methods are commonly used [11]. Continuation methods slowly increase the effect of penalization on the design variables over the course of optimization iterations. This is achieved by the gradual increment of the exponent p in the SIMP formulation, usually from 1 to 3.0 or 4. Although Stolpe and Svanberg [87] have demonstrated that continuation methods does not guarantee a complete 0–1 design², Bendsøe [11] showed that nonetheless they perform very well in practical applications, notably if used together with a regularization scheme, such as filtering. Additionally, It had been shown that for compliance minimization problems in discrete form, for a value of p large enough, there exists a globally optimal solution in the 0–1 form [74]. In this work, we have used the SIMP approach and it was observed that increasing p from 1 to 3.5 with increments of 0.5 yielded good results.

Topology optimization problems can have a variety of objective functions, for example, eigenfrequency maximization for structural dynamics, compliant mechanism design. As mentioned, the cases considered in this work are associated with compliance minimization, where the objective is to find the stiffest structure composed of a fixed volume of material, subjected to a set of loads and supports. To minimize the compliance means to minimize the strain energy of the structure, or, equivalently, maximize its stiffness. The objective function f is then:

$$f(\rho) = \mathbf{F}^T \mathbf{K}^{-1}(\rho) \mathbf{F}, \quad (2.7)$$

²0–1 design means that no intermediate density is present. Any point of the structure either has material or not. That indicates a clear, defined boundary $\partial\Omega$.

where \mathbf{F} is the global load vector and \mathbf{K} is the global stiffness matrix. Note that the stiffness matrix depends upon the density, i.e., the topology of the structure, the actual material distribution. The objective function changes from iteration to iteration, creating a different problem at each design loop. For compliance minimization problem, we need the equilibrium equations (for linear elasticity), solved for the displacements \mathbf{U} , given by the finite element method:

$$\mathbf{KU} = \mathbf{F}. \quad (2.8)$$

Plugging the Eq. (2.8) into Eq. (2.7), we obtain:

$$\begin{aligned} f(\rho) &= \mathbf{F}^T \mathbf{K}^{-1}(\rho) \mathbf{K}(\rho) \mathbf{U} \\ &= \mathbf{F}^T \mathbf{U} \equiv c, \end{aligned} \quad (2.9)$$

defining the compliance, c . Alternatively, the compliance can be written as:

$$c = \mathbf{U}^T \mathbf{KU}. \quad (2.10)$$

Therefore, the topology optimization problem of minimum compliance in the discrete form may now be formulated. We present two formulations, the Simultaneous Analysis and Design (SAND) [47] and Nested Analysis and Design (NAND). In NAND, only the design variables are taken as optimization variables, and state equations are solved within each iteration (which can result in expensive state equations evaluation). In SAND, the equilibrium equations are solved simultaneously with the optimization problem. In other words, they are also taken as optimization variables. In our formulation, this amounts to the minimization of the design variables, as well as the displacements \mathbf{U} :

$$\begin{aligned} \min_{\rho, \mathbf{U}} \quad & \mathbf{F}^T \mathbf{U} \\ \text{s.t.:} \quad & \mathbf{K}(\rho) \mathbf{U} = \mathbf{F} \\ & V(\rho) = \int_{\Omega} \rho \, dV \leq V_s \\ & \rho = \rho(\mathbf{x}), \forall \mathbf{x} \in \Omega; \quad \rho(\mathbf{x}) \in [\rho_{\min}, 1]. \end{aligned} \quad (2.11)$$

The NAND formulation is presented in Eq. (2.12):

$$\begin{aligned} \min_{\rho} \quad & \mathbf{F}^T \mathbf{U}(\rho) \\ \text{s.t.:} \quad & V(\rho) = \int_{\Omega} \rho \, dV \leq V_s \\ & \rho = \rho(\mathbf{x}), \forall \mathbf{x} \in \Omega; \quad \rho(\mathbf{x}) \in [\rho_{\min}, 1]. \end{aligned} \quad (2.12)$$

where $\mathbf{x} \mapsto \mathbf{U}(\mathbf{x})$ is an implicit function defined through the equilibrium equations $\mathbf{KU} = \mathbf{F}$. The equilibrium equations are implicitly solved for

displacements \mathbf{U} in a nested loop in order to evaluate the objective function and perform sensitivity analysis. We note that using the NAND formulation, the problem is written in terms of the design variables only.

Both NAND and SAND formulations considers two constraints: density bounds and maximum volume constraint. Note that the dependence of the stiffness matrix on $\boldsymbol{\rho}$ is based on the relationship of the material distribution on the actual structure and its resulting stiffness. Finally, we quickly remark that the bounds of $\boldsymbol{\rho}$ may also be defined by the Ersatz parameter (see Eq. (2.6)).

We note that during our formulation, we have used the density function ρ extensively. The optimization process takes place in the *design variables* rather than on the density. The density is then updated relying for the most part on the design variable, that is, the final density distribution is a function of the design variable field. In usual formulations and discretizations using standard elements, the direct assignment of density from the design variables may result in issues of checkerboarding (and/or mesh dependency). The separation of fields, namely density and design, allows the application of regularization schemes (discussed in section 2.6) as well as independent refinement, leading to a higher resolution topology (see, for example, [67]).

We now give a summary of the finite element method and then discuss common approaches to topology optimization problems.

2.2.1

A Brief Review of the Finite Element Method for Linear Elasticity

The system of linear equations $\mathbf{KU} = \mathbf{F}$, where \mathbf{K} is the stiffness matrix, arises when the finite element method is employed. The system is then solved numerically for the displacement \mathbf{U} .

Let the displacement field \mathbf{u} in each element be approximated by $\mathbf{u}^{(e)}$, such that

$$\mathbf{u}^{(e)}(\mathbf{x}) = \sum_{i=1}^n u_i N_i(\mathbf{x}) = \mathbf{N}\hat{\mathbf{u}}^{(e)}, \quad (2.13)$$

where N_i are the shape functions, $\hat{\mathbf{u}}^{(e)}$ represents a listing of nodal displacements for element e and n is the number of nodes in element e .

Once the displacements at the points within an element are obtained, the three-dimensional *strains* can be determined by means of the following relationship:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \end{bmatrix} = \mathbf{S}\mathbf{u}^{(e)}, \quad (2.14)$$

where u , v and w are the x , y and z components of the displacement, respectively, and \mathbf{S} is a suitable linear operator. From Eq. (2.13), the above equation can be approximated as:

$$\boldsymbol{\varepsilon} = \mathbf{S}\mathbf{N}\hat{\mathbf{u}}^{(e)}. \quad (2.15)$$

The matrix product $\mathbf{S}\mathbf{N}$ represents the strain-displacement matrix, also called \mathbf{B} and can be expressed as:

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 & \frac{\partial N_i}{\partial y} & 0 & \frac{\partial N_i}{\partial z} \\ 0 & \frac{\partial N_i}{\partial y} & 0 & \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial z} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix}^T.$$

Three-dimensional *stresses* can be written as:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{zx} \end{bmatrix}^T = \mathbf{C}(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) + \boldsymbol{\sigma}_0, \quad (2.16)$$

where $\boldsymbol{\sigma}_0$ are the initial residual stresses, $\boldsymbol{\varepsilon}_0$ are the initial strains and \mathbf{C} is the material constitutive matrix (also called stiffness tensor). The \mathbf{C} matrix can be derived from stress-strain relationship, and, isotropic materials³, is given as:

³In this work we consider only isotropic materials. For anisotropic materials the placement of principal directions of the material should be considered as design variable as well (optimization of principal directions and/or microstructure may be solved as a topology optimization problem on its own).

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}, \quad (2.17)$$

where ν and E are the Poisson ratio and the Young's modulus, respectively.

Defining nodal forces \mathbf{f} , it can be demonstrated that:

$$\mathbf{f}^{(e)} = \mathbf{K}^{(e)} \mathbf{u}^{(e)} + \mathbf{q}^{(e)}, \quad (2.18)$$

where \mathbf{q} are the elementary forces due to body forces, traction forces, initial strain and stress and $^{(e)}$ indicates element-wise. Therefore, the quantity \mathbf{q} is given by:

$$\mathbf{q}^{(e)} = - \int_V \mathbf{N}^T \mathbf{b} \, dV + \int_{\Gamma} \mathbf{N}^T \mathbf{t} \, d\Gamma - \int_V \mathbf{B}^T \mathbf{C} \boldsymbol{\varepsilon}_0 \, dV + \int_V \mathbf{B}^T \boldsymbol{\sigma}_0 \, dV, \quad (2.19)$$

where \mathbf{b} are the body forces components, \mathbf{t} are the traction forces and V represents the element volume. The stiffness matrix can be finally computed using:

$$\mathbf{K}^{(e)} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} \, dV. \quad (2.20)$$

If the initial system is self equilibrating, as in the case of no residual stresses, then the forces given by Eq. (2.19) are zero. Thus the evaluation of this term is frequently omitted. The quantity $\mathbf{K}^{(e)}$ is then evaluated by numerical integration over the element volume (at the Gauss points).

The global stiffness matrix can be obtained from the assembly of the elementary stiffness matrices. Therefore, taking global values instead of local, we get the well known system of equations $\mathbf{KU} = \mathbf{F}$, where \mathbf{F} is the global load vector. This system is solved for the unknown displacement \mathbf{U} . Solving this system of equations numerically is investigated in details in Chapter 5.

For a deeper discussion on the finite element method, the reader is referred to [8, 48].

2.3

Density-based Methods

The most commonly used method for structural topology optimization can be classified as density-based. Others methods includes equation-driven methods such as level-set and phase-field, evolutionary methods, among others. Density-based methods operate on a fixed domain of finite elements, with the

intent of minimizing/maximizing the objective function by identifying whether each element consists of solid material or void.

In topology optimization, the discrete nature of the design variables (solid or void) poses as a challenging integer programming problem. The discrete design variables can be replaced by continuous variable, that assumes the physical interpretation of density. The formulation given in the prior section considered this aspect, and we have already introduced the SIMP model. A critical aspect in density based methods is the selection of an interpolation function and penalization technique to explicit the physical quantities in terms of relaxed design variables. As aforementioned, the distribution of design variables into the ρ quantity in density-based methods is physically interpreted as the material density in each element in the extended domain Ω .

In order to drive the solution into a solid/void design, implicit penalization techniques are used. The most common is the SIMP model, used here. However, alternatives interpolation functions were proposed, for example, the Rational Approximation of Material Properties (RAMP) [86]. Differently from SIMP, it presents nonzero sensitivity at zero density. Other approach, known as SINH [16], represents the density in terms of *sinh* functions. These interpolation schemes have also been tweaked to deal with topology optimization problems on alternative physics.

As mentioned, we observe that these methods are often used together with a continuation scheme, slowly increasing the value of the penalization parameter, effectively moving the solution towards a 0–1 design.

2.4

Other Methods

There are other methods for topology optimization. They will be briefly introduced and not discussed in detail here. This work uses density-based and does not have the objective of bringing the discussion about different methods. For a review on different methods and the state-of-the-art in each one, the reader is referred to the recent comprehensive review article by Deaton and Grandhi [27].

Boundary Variation Methods

As opposed to density-based methods, which parametrize each element in the domain, boundary variation methods uses differential equations to implicitly define the boundaries of the design domain. Two of the most

popular methods in this category are the level-set method and the phase-field method [37].

The group of solution methods using the level-set approach [68, 80] often involves solving the Hamilton-Jacobi equation for the evolution of shapes, which is time-dependent. In the solution process, $d\mathbf{x}/dt$ represents the movement of a point driven by the objective function such that it can be expressed in terms of the position of \mathbf{x} and geometry of the surface at that point. The optimal topology by the level-set is then determined by finding the structural boundary, which in turn is the solution of a PDE on Φ given by

$$\frac{\partial \Phi(\mathbf{x})}{\partial t} = -\nabla \Phi(\mathbf{x}) \frac{d\mathbf{x}}{dt} \equiv -\nabla \Phi(\mathbf{x}) \Gamma(\mathbf{x}, \Phi) \Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}), \quad (2.21)$$

where $\Gamma(\mathbf{x}, \Phi)$ is the “speed vector” of the level-set and depends on the objective of optimization. This vector is obtained as a descent direction of the objective via sensitivity analysis.

The level-set method uses shape derivative, sometimes in combination with topological derivative, to drive towards the optimal topology (see, for example, [17]), allowing the introduction and elimination of holes, which, in turn, defines a true topological design. Explicit solution to the PDE is usually performed on conventional level-set methods, leading to a few issues such as time step size restrictions for convergence and required reinitialization necessary for numerical convergence. However, a number of alternatives have been proposed to circumvent these problems. For a deeper discussion, the reader is referred to [102] where a detailed level-set review is presented.

The phase-field method [36, 96] is another popular technique that is also based on PDEs for the definition of the topology. It solves the Allen-Cahn equation through a FE mesh, a time-dependent reaction diffusion equation. The phase field ϕ takes values close to 1 in voids and values close to -1 if material is present. The interface between material and void is described by a diffuse interfacial layer proportional to a small length scale parameter. Converged topologies by the phase field method as well as for level-set have shown that they are dependent on the initial guess.

Hard-kill Methods

Another class of methods for topology optimization include the so-called hard-kill methods. They work by gradually removing (or adding) a finite amount of material from the design domain. The removal of material is based on heuristic criteria, which, in turn, may be based on sensitivity information. Differently from density-based, in hard-kill methods the domain boundaries

are sharply defined, as the formulation only allows the design variables to be taken as existence (material) or absence (void) of finite elements.

The most popular hard kill method is the Evolutionary Structural Optimization (ESO) [106]. In ESO, the design is updated based on stress information. That is, in a FE discretization, elements whose stress is below a threshold are removed. Additional formulations change the criterion function from stress to a sensitivity number that is essentially compliance. Further, modified formulation allows also the addition of material (Bi-directional ESO or, BESO), and a modified sensitivity based on filters exhibits stable convergence to mesh independent and checkerboard-free solutions.

ESO and BESO formulations can be easily coupled with commercial finite element codes due to its simplicity and use of readily available information from FEA. Thus, optimization with non-linear responses becomes relatively easy. Furthermore, ESO/BESO have been expanded to include multicriteria optimization, although it consists in a simple single weighted criteria. Criticism to ESO method can be found in [76, 109].

2.5 Optimizer

The optimizer is the algorithm responsible for the direction such that the design variables should be updated. The algorithm used to update the design variable and thus advance on the optimization process can differ from density-based methods and phase-field, for example. Several mathematical programming algorithms can be applied on density-based methods, such as, Optimality Criteria (OC) [11], Method of Moving Asymptotes (MMA) [94], Sequential Linear Programming (SLP), Sequential Quadratic Programming (SQP), among others.

In this work, the compliance minimization problems are solved using the OC algorithm. The main advantage of OC is its simplicity, and it is most effective when a single constraint is present⁴. Here, one can simplify the constraints down to one: volume fraction.

In order to advance in a gradient-based optimization process, we often need the sensitivity of the objective function with respect to the design variable. In topology design, the most effective method for calculating derivatives is by using the adjoint method, where the derivatives of the displacement are not calculated explicitly. In this case, we can rewrite the objective function adding

⁴When more than one constraint is present, OC methods are usually considered to become impractical, and, in turn, sequential approximate optimization (SAO) methods become the preferred choice.

a zero function:

$$c(\boldsymbol{\rho}) = \mathbf{F}^T \mathbf{U} - \tilde{\mathbf{U}}^T (\mathbf{K} \mathbf{U} - \mathbf{F}),$$

where $\tilde{\mathbf{U}}$ is any arbitrary, fixed real vector. Differentiating and rearranging, we obtain

$$\frac{\partial c}{\partial \rho_m} = \left(\mathbf{F}^T - \tilde{\mathbf{U}}^T \mathbf{K} \right) \frac{\partial \mathbf{U}}{\partial \rho_m} - \mathbf{U}^T \frac{\partial \mathbf{K}}{\partial \rho_m} \mathbf{U}, \quad (2.22)$$

which can be in turn written as

$$\frac{\partial c}{\partial \rho_m} = -\tilde{\mathbf{U}}^T \frac{\partial \mathbf{K}}{\partial \rho_m} \mathbf{U},$$

where $\tilde{\mathbf{U}}$ satisfy the adjoint equation:

$$\mathbf{F}^T - \tilde{\mathbf{U}}^T \mathbf{K} = \mathbf{0}. \quad (2.23)$$

We see that to satisfy the equality from Eq. (2.23), we obtain $\tilde{\mathbf{U}} = \mathbf{U}$. Therefore, Eq. (2.22) is reduced to simply

$$\frac{\partial c}{\partial \rho_m} = -\mathbf{U}^T \frac{\partial \mathbf{K}}{\partial \rho_m} \mathbf{U},$$

which can be easily derived in the form

$$\begin{aligned} \frac{\partial c}{\partial \rho_m} &= -\mathbf{U}^T \frac{\partial \mathbf{K}}{\partial \rho_m} \mathbf{U} \\ &= -\mathbf{U}^T \frac{\partial}{\partial \rho_m} \left(\int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV \right) \mathbf{U} \\ &= -\mathbf{U}^T \frac{\partial}{\partial \rho_m} \left(\int_V \mathbf{B}^T (\rho_m^p) \mathbf{C}^0 \mathbf{B} dV \right) \mathbf{U} \\ &= -\mathbf{U}^T \frac{\partial}{\partial \rho_m} (\rho_m^p) \mathbf{K}_e^0 \mathbf{U} \\ &= -p (\rho_m)^{p-1} \mathbf{U}^T \mathbf{K}^0 \mathbf{U}, \end{aligned} \quad (2.24)$$

where the subscript m used during this derivation indicates *density element-wise*, the superscript 0 refers to the solid phase ($\rho = 1$) and the Ersatz parameter was omitted for convenience. We also note that the derivative only involves information at the element level, thus the use of the subscript e in the last equation.

The volume constraint sensitivity with respect to the design variables can be calculated as follows:

$$\frac{\partial V}{\partial \rho_m} = \int_{\Omega_m} \rho dV \equiv V_s. \quad (2.25)$$

These quantities are required for gradient-based optimizer such as OC.

Sequential Approximate Optimization (SAO) algorithms in structural op-

timization involve replacing the objective and constraints functions with their approximation at the current design point. Essentially, it solves an approximate problem. In the OC method, the objective function is approximated using exponential intermediate variables and then linearized using first order Taylor expansion.

Now we show the necessary conditions of optimality for the density ρ of a minimum compliance problem using the SIMP.

In the OC update scheme, an explicit expression for an optimal density candidate for the next iteration, denoted by ρ_{min}^* , can be determined as:

$$\rho_{min}^* = \rho_m B_m^\eta, \quad (2.26)$$

where η is a numerical damping to add stability to the method (for compliance minimization problems, $\eta = 1/2$ is typically used), and B_m is found from the optimality condition as:

$$B_m = \frac{\frac{\partial c}{\partial \rho_m}}{\lambda \frac{\partial V}{\partial \rho_m}},$$

where λ is the Lagrange multiplier that can be determined using, for example, a bisection method and V represents the volume constraint. Plugging in Eqs. (2.24) and (2.25),

$$B_m = \frac{-p \rho_m^{p-1} c}{\lambda V_s}. \quad (2.27)$$

Still, Eq. (2.27) constitutes only a candidate, as it has to satisfy the problem's density bounds. Hence,

$$\rho_{min}^{new} = \begin{cases} \rho^+, & \text{if } \rho_{min}^* \geq \rho^+ \\ \rho^-, & \text{if } \rho_{min}^* \leq \rho^- \\ \rho_{min}^*, & \text{otherwise} \end{cases} \quad (2.28)$$

where ρ^+ and ρ^- are the bounds for the search region given by:

$$\begin{aligned} \rho^+ &= \min(1, \rho_m + m) \\ \rho^- &= \max(\rho_{min}, \rho_m - m), \end{aligned} \quad (2.29)$$

where m is a positive prescribed move-limit of the density and usually $m = 0.2$.

For a deeper discussion on the Optimality Criteria method, including derivation of the equations presented here, the reader is referred to [11, 12].

In this work, we will use the OC update scheme, since it is suitable for our single-constraint problem. However, an improvement to the OC method had been proposed by Groenwold [44]. The OC method, which is based on reciprocal intervening variables, may be improved with regard to efficiency and

accuracy at little increased complexity by the use of exponential intervening variables, resulting in a dual formulation.

It was proposed convex subproblems for use in SAO algorithms for the classical topology optimization, encompassing *any* number of linear constraints (as opposed to *one* in the OC). The subproblems are based on reciprocal and exponential approximations of the penalized compliance objective function. It was noted that when a single linear constraint on the volume was present, the application of a rudimentary dual SAO algorithm based on *exponential* intervening variables yielded update schemes identical to those achieved by the OC for any arbitrary value of the heuristic numerical dumping η . Further, the use of SAO with *reciprocal* objective function is identical to OC with $\eta = 0.5$. Further discussion of the equivalence of the OC and SAO methods, we refer the reader to Groenwold's paper [44].

2.6

Regularization Techniques

In topology optimization, regularization refers to the process of controlling the density and/or sensitivities values in order to prevent numerical problems and control the quality of the final topology obtained. Manufacturing constraint, perimeter constraint and filtering are some regularization and restriction techniques available. If solving ill-posed problem, solution may converge to finer and finer members, with micro-structural holes. Regularization is often necessary to provide well-posedness to usual discretizations. Here we will discuss filters, which consists in a popular regularization method in density-based topology optimization. We also note that perimeter constraints may also be used as a restriction for the topology, alleviating the problem of oscillating solid/void [4].

Filters addresses the aforementioned numerical problems of checkerboarding and mesh dependency issues on the final topology. We remark that checkerboard, for example, can be circumvented in usual discretization by using higher-order elements, and in unstructured polyhedral meshes, the checkerboarding is naturally eliminated. Further, we are interested in obtaining high fidelity topologies, that approximates analytical solutions, without the intent of obtaining mesh independent results. Our scheme does not need additional application of regularization schemes nor restriction methods. With the application of a filtering scheme we could employ a length scale constraint, for example, if we wish to obtain mesh-independent results.

Restriction methods imposes bounds to the maximum allowed oscillation of ρ , circumventing the problem of rapid alternation of material and void, a

characteristic of the checkerboard pattern. An example of a restriction methods is perimeter control, which was proved to lead to existence of solution of previously ill-posed problem [4].

Filters prevent the creation of checkerboards and alleviates mesh-dependency issues. They are categorized in two major groups: density filters [15] and sensitivity filter [83]. Sensitivity filters are applied to the design sensitivities rather than the design variables themselves. In sensitivity filters, the design sensitivities are modified at each iteration of the optimization, making the design sensitivity of a given element depend on a weighted average over the element and some neighbors.

The basic concept of density filter is to modify the density value of an element based on the densities of elements in a localized neighborhood [15]. The filter relies on the following principle: it gets the centroid of every element and apply a sphere of radius set by the user on element i . Then, the density of element i is determined by a combination of its own density and density of the collection of elements that lie in the sphere of influence of element i . Usually, the distance between each of these elements and element i is used as a weighting factor on how the density of element i is calculated.

The sensitivity is one of the quantities that usually goes through filtering, as a way to ensure mesh-independent designs. The sensitivity filter follows the same idea of a weighted value as density filters. The filter may be defined as

$$\widehat{\frac{\partial c}{\partial \rho_k}} = \frac{1}{\rho_k \sum_{i=1}^N \hat{H}_i} \sum_{i=1}^N \hat{H}_i \rho_i \frac{\partial c}{\partial \rho_i}, \quad (2.30)$$

where a weighting factor \hat{H}_i is written as:

$$\hat{H}_i = r_{\min} - \text{dist}(k, i), \quad (2.31)$$

where r_{\min} refers to radius of the filter sphere and $\text{dist}(k, i)$ is the distance between the point k and center of filter sphere i .

The filter radius is usually independent of element size, thus solving the issue related to mesh-dependency. Additionally, as checkerboard is actually a series of “thin members” they are also circumvented with the application of a filter.

Well posed topology optimization formulation can be achieved by several ways. Checkerboards may be circumvented with the use of higher-order finite elements, filters or alternative discretizations. Mesh dependency issues are addressed with restrictions methods, such as perimeter control, local gradients constraints and also filters. Further, convergence to local minima is alleviated with continuation on the penalty parameter p of the SIMP model. The reader

is referred to [84] for a review on regularization and restriction methods.

2.7

Discretization Choice

The discretization of a given domain can be achieved in several ways, using a handful of different types of finite elements. An inappropriate choice of finite element discretization can lead to non-optimal results. As mentioned before, the appearance of numerical instabilities is attributed to poor modeling of the response field.

The appearance of the so-called checkerboard patterns is due to the artificially high stiffness of first order quad/brick and/or triangular/tetrahedral elements [28]. The alternating solid-void patterns is favored in compliance minimization and compliant mechanism problems. As discussed, the most common approach to avoid such pattern is to apply a filtering technique [15, 45, 95]. Although the application of regularization schemes, such as filtering, is successful in suppressing these instabilities, they often involve heuristic parameters that can augment the optimization problem. It has also been demonstrated that using nonconforming two-dimensional [51] or three-dimensional [52] elements alleviates the appearance of such undesired patterns.

The use of uniform hexagonal mesh (honeycomb-like) with Wachspress shape functions have been investigated [97]. The shape of the mesh resulted in non-biased topologies. In fact, several works point in the direction of arbitrary polygonal elements for topology optimization [53, 90, 98]. Further, polygonal/polyhedral discretizations impose less restrictions on the orientation of structural members, leading to topologies less mesh-biased when compared to usual structured discretizations (quads/bricks, for example).

In this work, three-dimensional arbitrary polyhedrons are used to discretize the design domain.

2.7.1

A Discussion on Interpolants

In two-dimensional arbitrary polygonal elements, the interpolation space can be constructed using several types of interpolants, such as Wachspress basis functions [97], Laplace (natural neighbor) shape functions [90, 91], mean value coordinates, among others. Some of these interpolants present extension to three-dimensional space.

Recently, mimetic finite-difference schemes were developed within a variational framework. This approach, called Virtual Element Method (VEM) does not require explicit construction and evaluation of basis functions, representing

an advance and greater flexibility over nodal-based finite elements in polygonal-based Galerkin methods.

The VEM ultimately presents an efficient generalized barycentric coordinate-based Galerkin method on two- and three-dimensional polytope-based grids. The stiffness matrix is decomposed in two terms by means of a projection operator: a consistent matrix (known) and a stability matrix (computed by numerical quadrature with generalized barycentric coordinates). The use of three-dimensional polytopes elements for topology optimization has been investigated by Gain [35] in the context of virtual element method and mimetic finite difference methods. The author proposed a numerical approach in which the polytopes can be treated as the displacement elements as well. VEM formulation does not make use of interpolants functions, and for that reason there's no explicit shape function evaluation. The author was able to achieve mesh-independent solutions on arbitrary polyhedral meshes.

Let us give a brief introduction to generalized barycentric coordinates on polytopes. Let $\Omega_e \in \mathbb{R}^d$ be an arbitrary polytope (polyhedron in \mathbb{R}^3), with vertices $\mathbf{x}_1, \dots, \mathbf{x}_n$. Given weight functions $\omega_a(\mathbf{x}): \Omega_e \rightarrow \mathbb{R}$, the functions $\phi_a(\mathbf{x}): \Omega_e \rightarrow \mathbb{R}$, $a = 1, \dots, n$, are called generalized barycentric coordinates with respect to Ω_e if they form a partition of unity:

$$\phi_a(\mathbf{x}) = \frac{\omega_a(\mathbf{x})}{\sum_{b=1}^n \omega_b(\mathbf{x})}, \quad \sum_{a=1}^n \phi_a(\mathbf{x}) = 1, \quad (2.32)$$

and satisfy the Kronecker delta property:

$$\phi_a(\mathbf{x}_b) = \delta_{ab}, \quad (2.33)$$

where any point \mathbf{x} that lies inside Ω_e is an affine combination of vertices:

$$\sum_{a=1}^n \phi_a(\mathbf{x}) \mathbf{x}_a = \mathbf{x}. \quad (2.34)$$

The barycentric coordinates are also non-negative, $\phi_a(\mathbf{x}) \geq 0$.

The generalized barycentric coordinates is a synonym of shape functions in finite element methods. Due to properties shown above, these coordinates can be used to construct trial and test approximations in Galerkin methods. For example, for a scalar function $u(\mathbf{x})$, the interpolant $u_e^h: \Omega_e \rightarrow \mathbb{R}$ is written as

$$u_e^h(\mathbf{x}) = \sum_{a=1}^n \phi_a(\mathbf{x}) u_a, \quad (2.35)$$

where due to the Kronecker delta property, u_a can be interpreted as the nodal value of the interpolant.

Although construction of rational shape functions started in 1975 with

Wachspress [103], applications of finite elements on polygons and polyhedrons are of more recent origin. There has been great effort to further the development of polyhedral finite elements methods over the past decade using generalized barycentric coordinates. However, accuracy and efficient numerical integration is a relevant issue in these methods. Non-polynomial shape functions are used in polyhedral FEM, affecting its numerical accuracy.

Several different barycentric coordinates have been developed for polytopes. For polygons (\mathbb{R}^2), Wachspress coordinates [103], mean value coordinates [32], harmonic and maximum-entropy coordinates [62], generalized three-point coordinates [33], among others are a few examples. In two dimensions, barycentric coordinates are well established and have demonstrated to perform very well for a wide range of applications. For a polyhedron (\mathbb{R}^3), though, only a few construction of generalized barycentric coordinates exist [63]. Wachspress generalization to \mathbb{R}^3 [30, 104], extensions to mean value coordinates [34] and metric coordinates [58] had also been proposed. Wicke et al [105] used three-dimensional mean value coordinates as shape functions on convex polyhedra. However, mean value coordinates (as well as Wachspress) are only defined on simplicial polyhedra (i.e., *convex* elements with *triangulated* faces), limiting severely its application in *arbitrary* polyhedra. In a related work [64], harmonic basis functions were investigated, however, closed form expressions for harmonic basis functions for general elements have to be computed numerically, resulting in subproblems. Recently, harmonic basis function was also investigated by [14].

A combination of polygonal/polyhedral FEM and VEM have been recently employed in the solution of second-order elliptic problems on polytope discretizations [63], presenting superior performance.

Moreover, currently available polyhedral interpolants are subject to restriction on the topology of admissible elements (such as convexity) and can be sensitive to geometric degeneracies [35]. As a result, in this work, the displacement numerical solution of the elasticity problem, performed in each topology optimization loop, is not carried out using the polyhedral FEM, nor it is the intent to dive into the virtual element method. We developed a mapping-based algorithm in which, although the topology is updated only on polyhedrons, it uses tetrahedral elements to carry out displacement computations.

2.8

Concluding Remarks

In this Chapter we have presented the classical topology optimization formulation for compliance minimization problems. Quick reviews of density-

based and equation-driven methods were discussed, as well as discussions on regularizations and restriction schemes.

Our approach focus on the use of unstructured polyhedral meshes for the design variables (and density). Polyhedron offers several advantages for topology optimization, such as provide topologies free of numerical instabilities, without the need of imposing restriction constraints, for example. This amounts to the elimination of rapid alternating solid-void elements and such numerical instabilities. The sensitivity does not need to be modified before use in the optimizer.

Further, we reviewed uses of interpolation functions in arbitrary polyhedra and issues associated with their use. In this work we will perform FEA by means of tetrahedral elements only, derived from the decomposition of the polyhedra. These aspects are covered in the next Chapter.

3

Embedding of Elements

Usually, in topology optimization problems, every single finite element in the domain is a potential void or structural member. The same domain discretization used in finite element analysis is used in the material distribution (i.e., topology) as well. This approach is usually called “element-based”.

Both displacement and the design variable fields are closely related, as they share the same discretization, i.e., a given element is linked to both fields. This implies that one field cannot be further refined without the other field being refined as well.

This work proposes the use of a two-level mesh representation, involving elements associated with displacement analysis and topology optimization elements associated with the design variables. Here, we call this approach *embedding of elements*, and the modified element containing optimization and displacement informations, a *super-element*. From here onwards, we shall call *displacement elements* the elements associated with the displacement mesh, used to perform FEA, *design elements* the elements associated with design variables and *density elements* the elements associated with the density mesh.

This chapter aims in detailing the embedding concept, its motivation, capabilities and how one discretization is linked to the other.

3.1

Motivation

Achieving an optimal topology through an element-based or nodal-based approach often requires further application of regularization methods such as a filtering scheme [11]. In existing element-based methods, each displacement element with uniform density is represented by a single design variable. In comparison, the design variables of nodal-based approaches are the nodal densities [45]. The element densities are then obtained from the nodal values using projection schemes.

We naturally think in expanding this concept by imposing multiples design variables into a single finite element or vice-versa. This amounts to the definition of two or more different discretizations. However, to avoid the

creation of several new nodal points, one can think as one of the discretization being a sub-discretization of a coarser mesh, containing a few common nodal points.

Constructing a finite element with multiple design variables has been studied in the topology optimization literature. Nguyen et al. [67] proposed a multi-resolution topology optimization scheme (MTOP), in which three distinct discretizations are employed: a coarse displacement mesh to perform finite element analysis, a finer design variable mesh to perform the optimization and a density mesh to represent material distribution. In usual MTOP application, the design variable and density mesh shares the same discretization and they are a sub-discretization of the coarse displacement mesh (see Fig. 3.1). In contrast, the MTOP framework can also deal with super-elements, in which a density mesh is nested inside a design variable mesh.

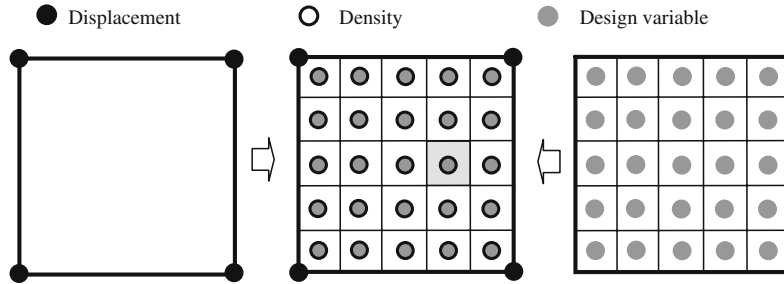


Figure 3.1: Multiresolution topology optimization scheme [67].

This work investigates the concept of super-element: several displacement elements embedded into a single design element. In our approach, differently from a simple MTOP expansion, we study the case where the element that carries the design variable is a polyhedron. The tetrahedrons are obtained by the *embedding* of elements approach, which consists in triangulating the polyhedron and generating a tetrahedral sub-discretization.

Early work on embedding displacement elements yielded good results [70], with the checkerboard phenomena being alleviated and higher resolution topologies being obtained in 2D as well as 3D (see Fig. 3.2). Other important motivation to embed tetrahedra into polyhedra comes from the issues associated with the direct use of polyhedra elements in FEA. As previously discussed in section 2.7.1, the early stage of development of three-dimensional shape functions for arbitrary polyhedron is still an issue for reliable and accurate use in finite element analysis.

The use of polyhedra meshes for structural topology optimization is promising in achieving pathology-free topologies [36, 98, 100]. This work aims in expanding the concept of embedding of elements to arbitrary polyhedra

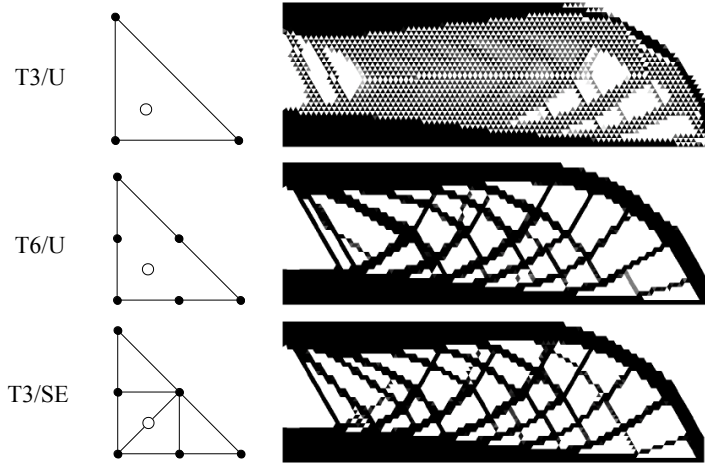


Figure 3.2: Higher resolution topologies obtained using embedding technique on structured meshes [70]. Triangular elements are shown.

meshes. We focus on a complete elimination of one-node or edge connections (“hinges”), inherently common in discretizations using brick elements. Further, finite element analysis can be easily carried out on linear tetrahedron elements, derived from an embedding process, and the separation of analysis and optimization variables is promising for achieving high fidelity solutions.

3.2

The Embedding Approach

Let’s first begin to describe the embedding procedure by defining three different meshes. Here, the concepts of finite element mesh, design variable mesh and density mesh are distinct. We use the finite element mesh to perform the displacement analysis, through FEA, the design variable mesh to perform optimization and the final topology is updated in the density mesh.

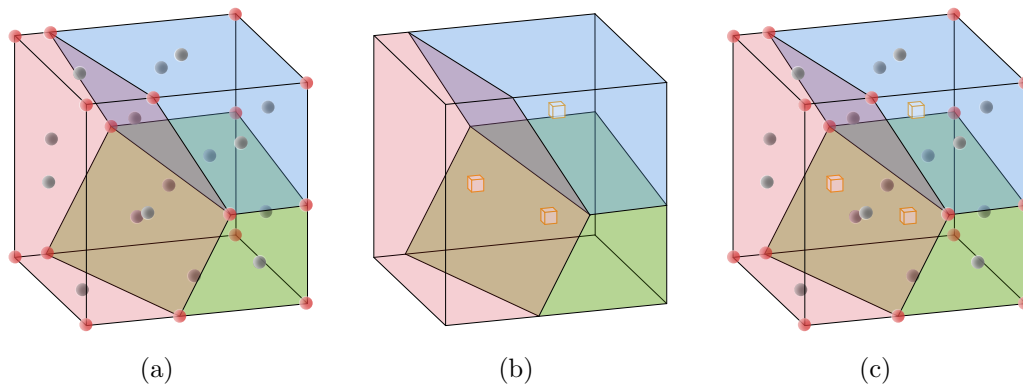


Figure 3.3: Two-level mesh representation. (a) Displacement variables; (b) design variables; (c) both variables superimposed: super-element mesh. Displacement nodes represented by spheres (red ones are from the original mesh; grey ones are the created nodes for triangulation), and design variables represented by small orange-outlined cubes.

The design variable mesh is composed of arbitrary polyhedron elements, originated from Voronoi diagrams. In order to carry out finite element computation, a tetrahedral mesh is nested inside the polyhedral mesh. A design domain is then represented by two distinct, but coupled meshes (Fig. 3.3 and 3.4).

The embedding approach consist in decomposing a polyhedron into tetrahedrons. By using the element and faces centroids as new nodes, new tetrahedrons are defined by triangulation. Algorithm 3.1 presents the embedding technique in pseudo-code format.

Algorithm 3.1 The embedding of elements algorithm

```

1: procedure EMBEDDING(NElem, nodes, connectivity)
2:   for all elements in domain do
3:     determine element centroid
4:     create new node, ec
5:     for all element's faces do
6:       count number of nodes
7:       if number of nodes > 3 then
8:         determine face centroid
9:         create new node, fc
10:        for all face's nodes do
11:          get node and its neighbor:  $n_i$  and  $n_{i+1}$ 
12:          new tetrahedron( ec, fc,  $n_i$ ,  $n_{i+1}$  )
13:          move to next node
14:        end for
15:      else
16:        new tetrahedron( ec, 3 face's node )
17:      end if
18:    end for
19:  end for
20: end procedure

```

Following the algorithm, we note that the polytope's centroid node is common to every tetrahedra derived from that polytope. The location of the centroid is also the location of the design variable on the polyhedral mesh. Figure 3.4 also illustrates the embedding of elements approach.

In the following sections, each field used in the embedding approach is discussed in detail.

3.3

Displacement Field

As previously discussed, the displacement mesh is used to carry out finite element analysis. The resulting displacement field is used to compute gradients and sensitivity information, which are then used to update the design variables.

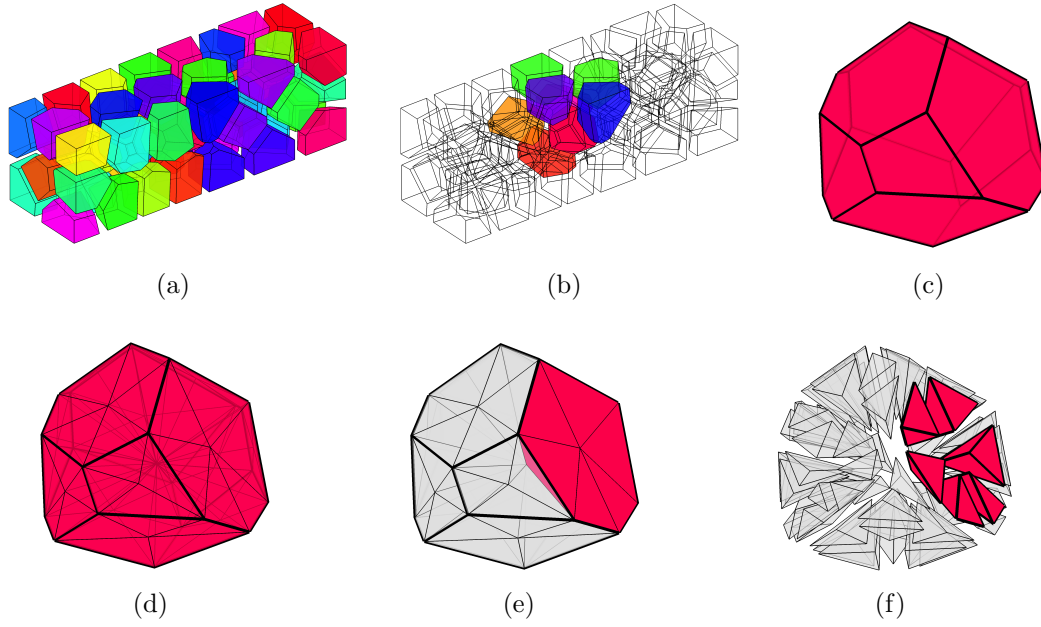


Figure 3.4: The embedding process on a simple mesh: (a) Exploded view of a beam; (b) Highlighted polyhedrons; (c) An isolated polyhedron; (d) New nodes are created in every polyhedron faces and tetrahedrons are defined by the previously described algorithm; (e) A n -vertex face generates n tetrahedrons; (f) Exploded view of a single polyhedral divided into tetrahedrons. This element in particular yielded 66 tetrahedrons.

The displacement mesh is obtained through the embedding algorithm and is composed of tetrahedrons only, with four nodes each (i.e., a linear tetrahedron, with 3 DOFs per node).

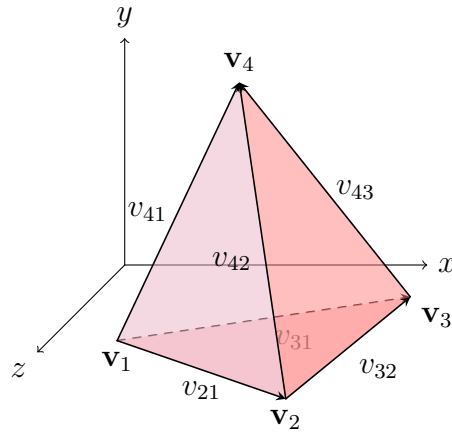


Figure 3.5: Tetrahedron nomenclature.

Figure 3.5 shows a typical four-node tetrahedron. Its geometry is fully defined by the position of the nodes $\mathbf{v}_i = (x_i, y_i, z_i)$ and faces' incidence. Then,

we can readily obtained its volume by

$$V = \int dx dy dz = \frac{1}{6} \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}.$$

Defining

$$a_1 = \begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}; \quad b_1 = - \begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix};$$

$$c_1 = - \begin{vmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{vmatrix}; \quad d_1 = - \begin{vmatrix} 1 & y_2 & 1 \\ 1 & y_3 & 1 \\ 1 & y_4 & 1 \end{vmatrix},$$

with the other constants defined by cyclic interchanging the subscripts in the order 1, 2, 3 e 4, it is possible to determine the shape function (see Eq. (2.13)) of the tetrahedron element:

$$N_i = \frac{a_i + b_i x + c_i y + d_i z}{6V}, \quad (3.1)$$

where the subscript i refers to the element number.

It is of interest that the tetrahedron elements are not greatly deformed, as they directly impacts the stability of the solution of the linear elasticity problem through iterative solvers. Slivers elements presents a few issues. For instance, the shape function has the Kronecker delta property in each node, meaning that its value goes from 1 to 0 in each adjacent node. We may observe that if the nodal distances are not homogeneously spread, the quantity $\nabla \mathbf{N}$ may become arbitrarily large¹, reflecting directly in the stiffness matrix through the strain-displacement matrix \mathbf{B} . Furthermore, slivers or other highly distorted elements may present near-zero volume, affecting the shape function evaluation through the quantity $6V$ (this quantity is called Jacobian).

These are some of the issues with badly shaped elements that can lead to numerical problems. A tetrahedron can be considered sufficiently bad if a single small edge is present. However, a tetrahedron product of the embedding process will contain small edges if and only if the polyhedron contains a small

¹That is, N_i decreases from 1 at \mathbf{v}_i to 0 at \mathbf{v}_{i+1} , over a very short distance, thus leading to a large $\nabla \mathbf{N}$.

edge. This issue is directly linked to mesh generation and will be discussed in details in Chapter 4.

3.4

Density Field

The density field is usually determined from the design variable field. Although the design variable does not have a physical meaning on its own, it can be thought of having a density physical meaning. The difference from design variable to density is that the density is a function of the design variable field *and* a regularization scheme.

As previously discussed, regularization methods are used in order to create a density distribution that alleviates problems associated with numerical issues that would exist if the density was obtained from a 1:1 mapping from the design variable². As a result, it is common in the topology optimization community to use *separate* fields to deal with design variables and density.

With that in mind, we define our density field over the polyhedral elements. Within each density element, the material density is constant.

This amounts to the final topology, i.e., $\rho = 1$, being a subset of the initial polyhedral mesh.

Our embedding algorithm does not make use of an explicit filter mechanism. It uses the embedding to create polytope-based final topologies³. Therefore, our density field is numerically identical to the design variable field. We have preferred to keep these as separate fields to make possible further extension of the framework.

As mentioned throughout this work, the density field composed of polytopes elements for structural topology optimization is advantageous in several ways.

3.5

“Super-elements”

As previously mentioned, the finite element mesh is nested inside the design variable mesh. In a geometrical sense, the displacement mesh can be thought as a subset of the design variable mesh.

The super-element concept accumulates information from both meshes. A super-element can be thought of a single polyhedron, however, containing in-

²The 1:1 mapping refers to a direct copy of the design field to the density field. That is, an element density would not be a function of nearby elements and would carry a density value that is equal to its design value.

³Although the embedding can be interpreted, in some sense, as a filter. We will get back to this aspect later on.

formation from both discretization, or, in other words, a group of displacement elements that corresponds to a design element.

For the sake of illustration, consider the polyhedral mesh from Fig. 3.4(a). The superposition of polyhedra and tetrahedra mesh constructs the super-element mesh (see Fig. 3.6).

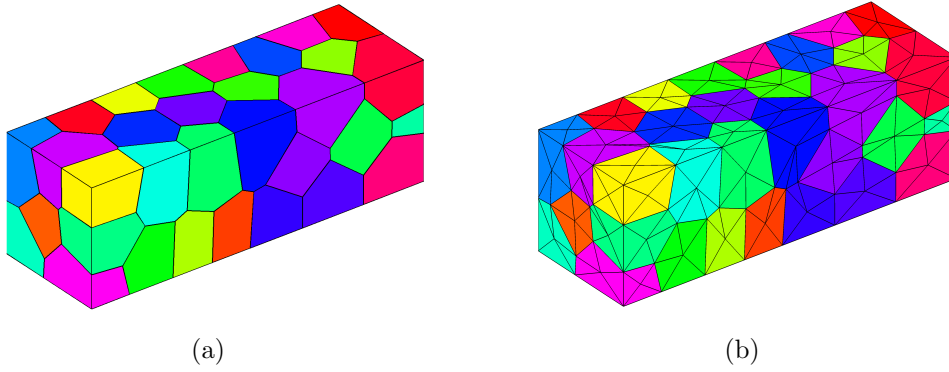


Figure 3.6: Different meshes. (a) The design variable mesh; (b) the finite element mesh, associated with displacement elements. With the superposition of these meshes, a super-element mesh is obtained. Polyhedrons individually colored for the sake of illustration.

We note that in a computational scheme, a super-element is never explicitly created or defined. A *mapping* between optimization variables/routines to and from FEA variables is responsible for the exchange of informations.

3.6

Mapping Between Fields

In structural optimization, the displacement field is used in the optimization process. In compliance minimization problems, the displacement field is necessary in order to evaluate the objective function and compute the gradients. Based on these informations, the design is updated.

Therefore, the resulting topology optimization formulation involves two separate fields: displacement and design. The proposed approach naturally decouples the finite element field and the optimization field (which shares the same discretization with density). Optimization computations are carried out on polyhedrons, whereas FEA is carried out on tetrahedrons.

Now that both fields have been defined, we introduce the idea of a *mapping* framework. Information from one mesh is mapped into the other (and vice-versa). We call the embedding of elements a *mapping-based* algorithm, as one discretization is consistently *mapped* into the other.

Let D be a displacement mesh, f be a displacement element and d be a design element. If f is a subset of d , then:

$$\mathbf{K}_f = (\rho_d)^p \mathbf{K}_f^0, \quad (3.2)$$

where $\mathbf{K}_e^0 = \int_{\Omega_e} \mathbf{B}^T \mathbf{C}^0 \mathbf{B} dV$ is the stiffness matrix for the reference element and p is the penalty parameter from the SIMP model. Expanding Eq. (3.2), the stiffness matrix for an element in the finite element mesh, considering the current value of density (or, in this case, design variable) can now be written as:

$$\mathbf{K}_e^{(e)} = \int_{\Omega_e} \mathbf{B}^T \mathbf{C} \mathbf{B} dV = \int_{\Omega_e} [\rho(\mathbf{x})]^p \mathbf{B}^T \mathbf{C}^0 \mathbf{B} dV. \quad (3.3)$$

Note that from the equation above, the polyhedron element passes the design information to its derived tetrahedrons. The group of finite elements that lie in the support of density element d can be defined as S_d (see Fig. 3.7):

$$S_d = \{f : \Omega_f \subseteq \Omega_d\}. \quad (3.4)$$

This definition is illustrated in Fig. 3.7.

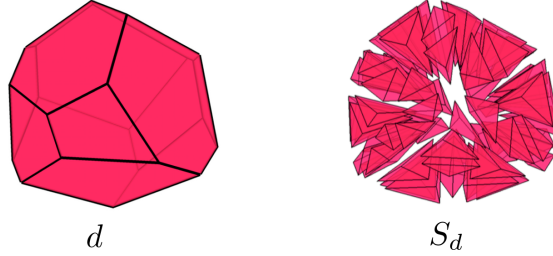


Figure 3.7: Element association between meshes: d on the left and S_d on the right.

In order to detail how computations are carried out using two meshes, consider, for instance, the sensitivity of the compliance (Eq. 2.24). It can be evaluated for every element in S_d as follows:

$$\frac{\partial c}{\partial \rho_d} = - \sum_{f \in S_d} \left[\mathbf{U}_f^T \frac{\partial \mathbf{K}_f}{\partial \rho_d} \mathbf{U}_f \right] = - \sum_{f \in S_d} [p(\rho_d)^{p-1} \mathbf{U}_f^T \mathbf{K}_f^0 \mathbf{U}_f]. \quad (3.5)$$

We point out that the task of determining set of elements $\{S_d\}_{m=1}^N$ for a N -polytope mesh is purely geometric. In fact, this association is performed during the mesh generation phase, where each polyhedron is decomposed into several tetrahedrons.

Let \mathbf{y} and \mathbf{z} be column vectors of tetrahedrons and polyhedron elements, respectively. A mapping between fields is given by the following relationship:

$$\mathbf{y} = \mathbf{P} \mathbf{z}, \quad (3.6)$$

where \mathbf{P} is a mapping matrix, consisting of 0's and 1's. The binary \mathbf{P} matrix indicates which tetrahedrons are part of a given polyhedron, that is, the collection of elements S_i is given by column i .

Therefore, recalling Eq. (3.5), we can express $\partial c / \partial \rho_f$ by mapping the polytopes densities into the tetrahedrons by:

$$\rho_f = \mathbf{P} \rho_d, \quad (3.7)$$

where ρ represent the density vector for each f and d elements.

For the sake of illustration, consider a two-dimensional case where each design element contains 8 displacement elements (Fig. 3.8). Let d denote the design element 9.

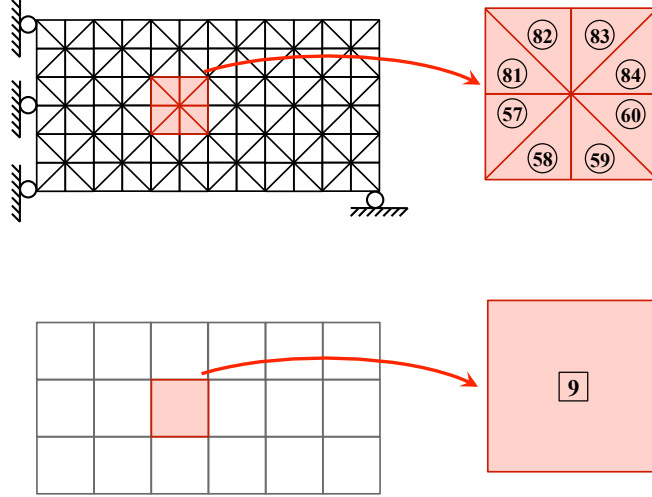


Figure 3.8: Mapping between elements [71].

The collection of finite elements that lies in its support is given by $S_9 = \{57, 58, 59, 60, 81, 82, 83, 84\}$. Therefore, its mapping matrix can be partially expressed as:

$$\mathbf{P} = \begin{array}{c} j = 9 \\ \downarrow \\ \left[\begin{array}{cccccc} \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ \vdots & \vdots & \vdots & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right] \end{array} \begin{array}{l} \leftarrow i = 57 \\ \leftarrow i = 58 \\ \leftarrow i = 59 \\ \leftarrow i = 60 \\ \\ \leftarrow i = 81 \\ \leftarrow i = 82 \\ \leftarrow i = 83 \\ \leftarrow i = 84 \end{array}.$$

The mapping \mathbf{P} may also be used to retrieve information from tetrahedron back to its polyhedron. This is achieved simply by means of \mathbf{P}^T (Fig. 3.9).

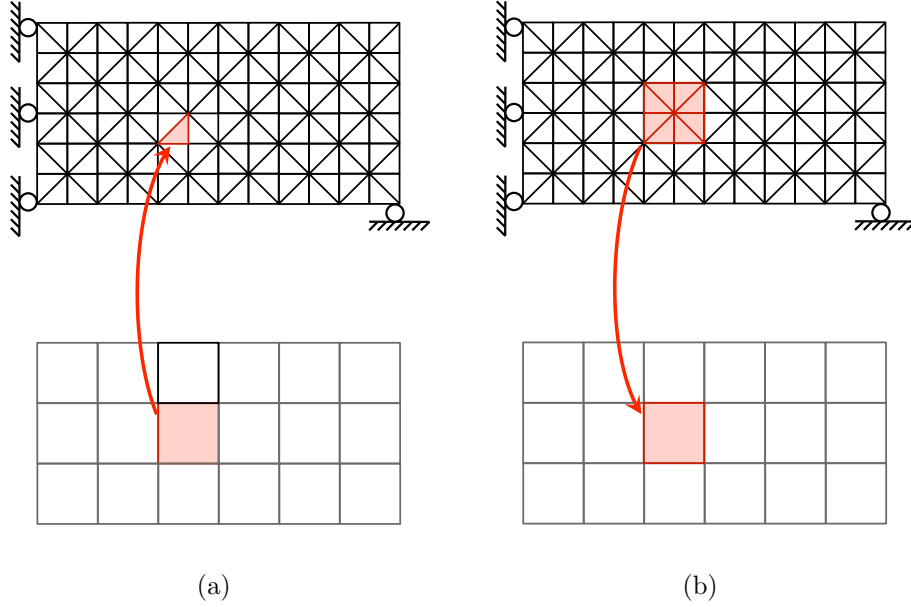


Figure 3.9: Map between the two discretizations done by means of (a) \mathbf{P} and (b) \mathbf{P}^T .

Furthermore, we point out that features such as symmetry may also be imposed by means of a linear mapping. Let's denote the aforementioned embedding mapping matrix \mathbf{P} as \mathbf{P}_E , denoting *embedding*. A new mapping matrix \mathbf{P}_S may be defined in which elements are parametrized with respect to a symmetry line. Density filters may also be used the same way. Since these are linear filters, to apply them simultaneously, we may simply set \mathbf{P} as

$$\mathbf{P} = \mathbf{P}_1 \mathbf{P}_2 \dots \mathbf{P}_{N_f}, \quad (3.8)$$

where N_f is the number of filters. For instance, to apply the embedding scheme, symmetry and density filters, we may set $\mathbf{P} = \mathbf{P}_E \mathbf{P}_S \mathbf{P}_D$, where E , S and D refers to embedding, symmetry and density, respectively.

The embedding itself can be thought of being essentially a filtering scheme. It enforces that solid and void elements stay grouped as polyhedrons. From the design variable mesh, the mapping of elements acts as a filtering, ensuring that every finite element sharing a node with the same physical position as the design node will have the same design value. We recall that the embedding, acting like a filter, is applied on the design variable mesh, and the subsequent lack of filtering amounts to a numerically identical density mesh.

3.7

Concluding Remarks

Concluding the chapter on the embedding technique, we outline its advantage over commonly used elements and/or topology optimization formulation:

- The use of polyhedron elements naturally alleviates the problem of mesh-biased sub-optimal solutions;
- The embedding framework connects with polygonal and polyhedral discretizations of mimetic-inspired methods [10,35];
- The mapping used to assign density values to tetrahedrons can be thought as a filtering scheme. As a linear filter, it may be combined with regularization filters or a symmetry filter. The framework does allow easy extensions;
- The modularity of the code and separation of computational domains allow the framework to be used in other areas of topology optimization. The proposed embedding scheme is general and expandable.

4

Mesh Generation

Meshing a given geometry means determining a discrete representation of some spatial *domain* Ω , characterized by node points and element connectivity. Meshing is needed in several areas of computational mechanics. Some typical applications includes numerical solution of partial differential equations (FEM, FVM, BEM), computer graphics and visualization, among others.

Discretization can be in Cartesian domain, using quadtree/octree or fully unstructured grids. In this work we focus our attention on fully unstructured polyhedral grids.

On a tetrahedral mesh, interpolation accuracy, discretization errors¹ and matrix conditioning for finite element applications are critically dependent on a high quality mesh consisting only of well-shaped, non-degenerate elements [81]. The performance and numerical accuracy of solving a PDE depend on the condition number of the resulting linear system of equations, which is heavily influenced by the shape of the finite elements.

The objective of this chapter is to describe the meshing algorithm, issues regarding elements shape and ways to circumvent them. Moreover, the meshes quality is investigated by determining the condition number of the stiffness matrix associated to two different types of problem.

The foundation of the developed algorithm is the PolyMesher code, proposed by Talischi et al. [99,101]. Its principles relies on an implicit description of the domain and centroidal Voronoi diagram are used for the discretization. Here we have further extended the code and deal with mesh optimization procedures that effectively penalizes small edges, leading to better-shaped tetrahedrons.

First, let us define some commonly used terms in three-dimensional geometry:

- **Vertices:** x, y, z location (also called **Node**);

¹*Interpolation error* is divided into two types (for most applications, including FEA): the difference between the interpolated function and the true function, and the difference between the gradient of the interpolated function and the gradient of the true function.

Discretization error is the difference between the approximation computed by the FEM and the true solution.

- **Edges:** Bounded by two vertices;
- **Surfaces:** closed set of edges;
- **Volume:** closed set of surfaces (also called **Element**).

For an accurate capture of the design response of several problems in topology optimization, the use of unstructured meshes may be required [98]. Based on Voronoi diagrams, three-dimensional polyhedral meshing turns out to be naturally unstructured.

A mesh is defined as a discrete representation Ω_h of some spatial *domain* Ω . It can be completely defined in terms of (unique) vertices and some sort of element incidence information (triangulation).

4.1

Implicit Representation

The domain is represented implicitly by the *signed distance function* [73], $d(x)$, that can take the following values

$$d(\mathbf{x}) \begin{cases} < 0, & \text{if } \mathbf{x} \in \Omega \\ = 0, & \text{if } \mathbf{x} \in \partial\Omega \\ > 0, & \text{if } \mathbf{x} \notin \Omega, \end{cases} \quad (4.1)$$

where $\partial\Omega$ denotes the boundary of Ω .

The signed distance function gives norm-2 of the shortest distance from a point \mathbf{x} in the space to the boundary, and its sign is determined by the location of \mathbf{x} with respect to Ω . That is, if \mathbf{x} lies inside the domain Ω , then the signed distance function would be minus the distance of \mathbf{x} to the boundary.

Using this representation, we can easily define the domain Ω and its boundary $\partial\Omega$ as:

$$\begin{aligned} \Omega &= \{\mathbf{x} \in \mathbb{R}^3 \mid d(\mathbf{x}) \leq 0\} \\ \partial\Omega &= \{\mathbf{x} \in \mathbb{R}^3 \mid d(\mathbf{x}) = 0\}. \end{aligned} \quad (4.2)$$

The boundary is represented by the zero level set of the signed distance function on an implicit representation (see Fig. 4.1).

This representation allows us to easily construct the distance function of simple geometries. For example, for a sphere $\Omega \in \mathbb{R}^3$, with center \mathbf{P} with radius r , its distance function can be given as:

$$d(\mathbf{x}) = \|\mathbf{x} - \mathbf{P}\|_2 - r. \quad (4.3)$$

Complex domain shapes can be produced by combination of these simple forms. For instance, consider the regions Ω_A and Ω_B , with corresponding signed distance functions d_A and d_B respectively. They can be combined as follows:

Union	$d_{\Omega_A \cup \Omega_B}(\mathbf{x}) = \min(d_{\Omega_A}(\mathbf{x}), d_{\Omega_B}(\mathbf{x}));$	(4.4)
Intersection	$d_{\Omega_A \cap \Omega_B}(\mathbf{x}) = \max(d_{\Omega_A}(\mathbf{x}), d_{\Omega_B}(\mathbf{x}));$	
Difference	$d_{\Omega_A \setminus \Omega_B}(\mathbf{x}) = \max(d_{\Omega_A}(\mathbf{x}), -d_{\Omega_B}(\mathbf{x}));$	
Complementation	$d_{\mathbb{R}^3 \setminus \Omega_A}(\mathbf{x}) = -d_{\Omega_A}(\mathbf{x}).$	

Some of the advantages of using implicit representation can be listed [73], such as:

- Easy to implement: set operators, blending, etc;
- Natural extension to higher dimensions;
- Level sets methods for evolving interfaces.

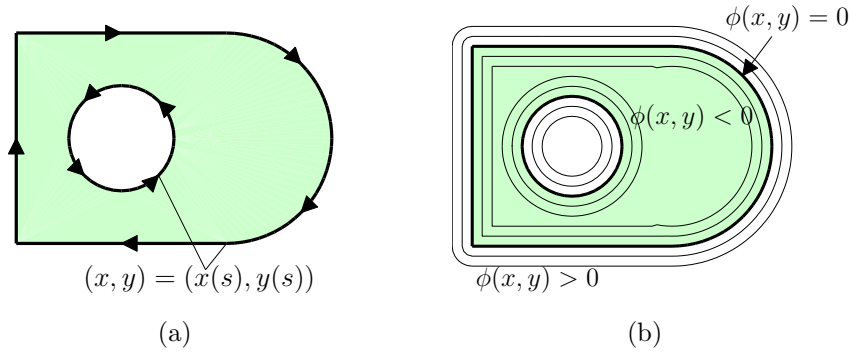


Figure 4.1: Geometric representation of a domain. (a) Explicit representation, with parametrized boundaries. (b) Implicit representation, with level sets of the distance function shown. The boundary corresponds to the zero level set. Figure from [72].

4.2

Delaunay Triangulation and Voronoi Tessellations

The unstructured polytope grid is achieved by the initial random placement of points, called *seeds* in the desired design domain. Then, Delaunay triangulation is performed and is followed by the construction of Voronoi cells around each seed.

Definition The Delaunay tessellation $DT(S)$ is obtained by connecting with a line segment any two points p, q of S for which a circle C exists that passes through p and q and does not contain any other seed of S in its interior or boundary. The edges of $DT(S)$ are called *Delaunay edges* [46].

The Delaunay triangulation follows the *empty circle condition* (sphere in \mathbb{R}^3): a given tetrahedron $pqrs$, for example, appears in the Delaunay triangulation $DT(S)$ if and only if its circumsphere encloses no other points of S . The Delaunay triangulation is the dual of the Voronoi tessellation.

Let $\Omega \subseteq \mathbb{R}^3$ be an open set. The set $\{V_i\}_{i=1}^k$ is called a *tessellation* of Ω if $V_i \cap V_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^k \bar{V}_i = \bar{\Omega}$. Given a set of k points $\{\mathbf{z}_i\}_{i=1}^k$ belonging to $\bar{\Omega}$, the Voronoi region (also called cell) \hat{V}_i corresponding to the point \mathbf{z}_i is defined by:

$$\hat{V}_i = \{\mathbf{x} \in \Omega \mid |\mathbf{x} - \mathbf{z}_i|_2 < |\mathbf{x} - \mathbf{z}_j|_2, \quad j = 1, \dots, k, j \neq i\}, \quad (4.5)$$

where $\mathbf{x} = (x, y, z)$ is any point in the domain and $|\cdot|_2$ denotes the Euclidean distance in \mathbb{R}^3 . The set $\{\hat{V}_i\}_{i=1}^k$ is a Voronoi tessellation (or diagram) of Ω , and each \hat{V}_i is a Voronoi cell. The k points \mathbf{z} are called generators or seeds. These cells are inherently polytopes.

Figure 4.2 shows a set of seeds, its Delaunay triangulation and the corresponding Voronoi diagram (2D for simplicity; trivial extension to 3D).

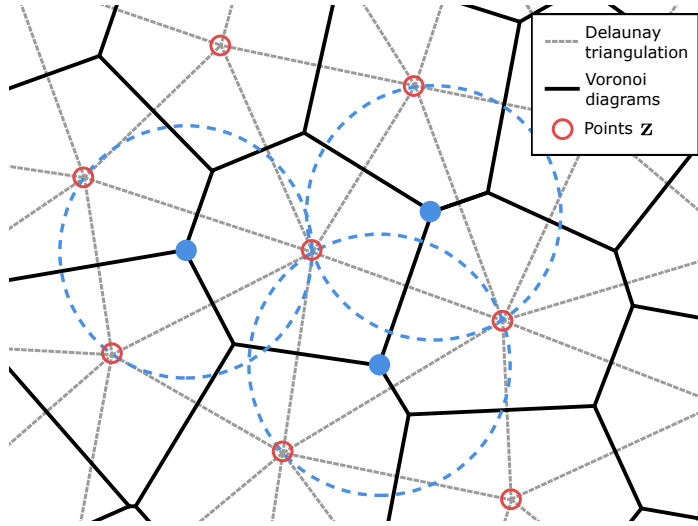


Figure 4.2: Delaunay triangulation and corresponding Voronoi diagram. Note the empty circle property: a circumcircle on Delaunay triangulation does not englobe any other point \mathbf{z} .

4.2.1

Centroidal Voronoi Tessellations and Lloyd's Algorithm

The Voronoi tessellation (and Delaunay triangulation) that arises from initial random or quasi-random placement of seeds may not be suitable for further embedding and finite element analysis. We may use the concept of Voronoi diagrams mass centroid to create higher quality regions (see Fig. 4.3).

Taking a density function ρ defined over a given region $W \subseteq \mathbb{R}^3$, the centroid of W is defined as follows:

$$\mathbf{z}^* = \frac{\int_W \mathbf{y} \rho(\mathbf{y}) \, d\mathbf{y}}{\int_W \rho(\mathbf{y}) \, d\mathbf{y}}. \quad (4.6)$$

For a region with constant density ρ , the centroid is also the center of mass.

Given a set of k seeds \mathbf{z}_i , their associated Voronoi regions are \hat{V}_i , $i = 1, \dots, k$. Further, given the regions \hat{V}_i , we can compute their centroids \mathbf{z}_i^* . A Centroidal Voronoi Tessellation (CVT) is one that satisfy

$$\mathbf{z}_i = \mathbf{z}_i^*, \quad i = 1, \dots, k. \quad (4.7)$$

In other words, the generators are the mass centroids of the corresponding Voronoi cell themselves. Finding the \mathbf{z}_i location that satisfies Eq. (4.7) means finding the generators of a CVT.

There are several ways to achieve a CVT available in the literature. Here, we investigate a CVT achieved by a popular technique called Lloyd's method [61]. It is an iterative process that places the next-iteration generators on the mass centroid of the current-iteration cell's mass centroid. The algorithm iterates until a certain tolerance between the generators and centroids distance is met. Fig. 4.3(b) and (d) were created using Lloyd's algorithm.

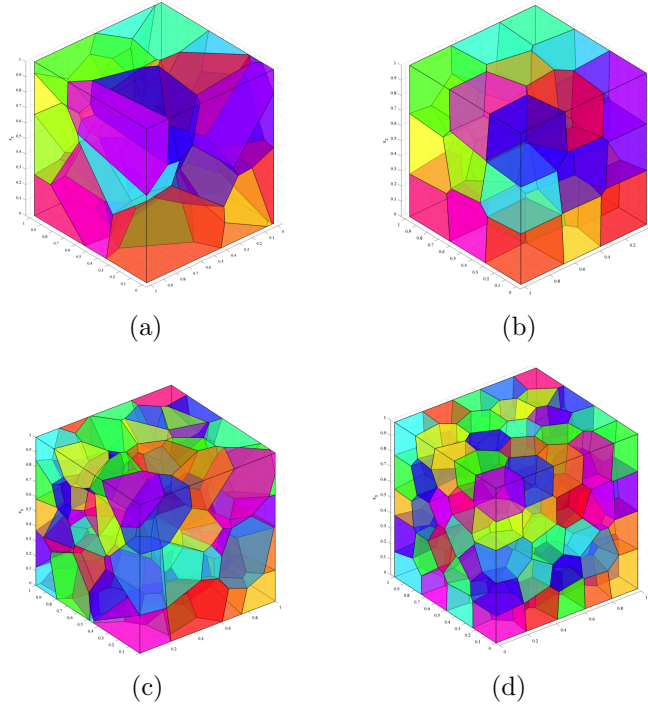


Figure 4.3: Initial seeds distributions and their CVTs. (a) 30 seeds, initial random distribution, (b) 30 seeds, CVT (after 40 Lloyd's iterations); (c) 120 seeds, initial random distribution, (d) 120 seeds, CVT (after 40 Lloyd's iterations).

Much of the CVTs obtained guarantee convex elements and good angles (that is, no near-zero or near-180° angles) but some undesired features are not explicitly penalized, such as small faces and small edges (see Fig. 4.4). Here we define a *small* quantity as less than 10% of the average value of that quantity. In other words, even a *good-looking* CVT mesh may contain a number of short

Voronoi edges in 2D or small facets and short edges in 3D [82]. As a result, we cannot rely solely on Lloyd's algorithm for obtaining high quality meshes.

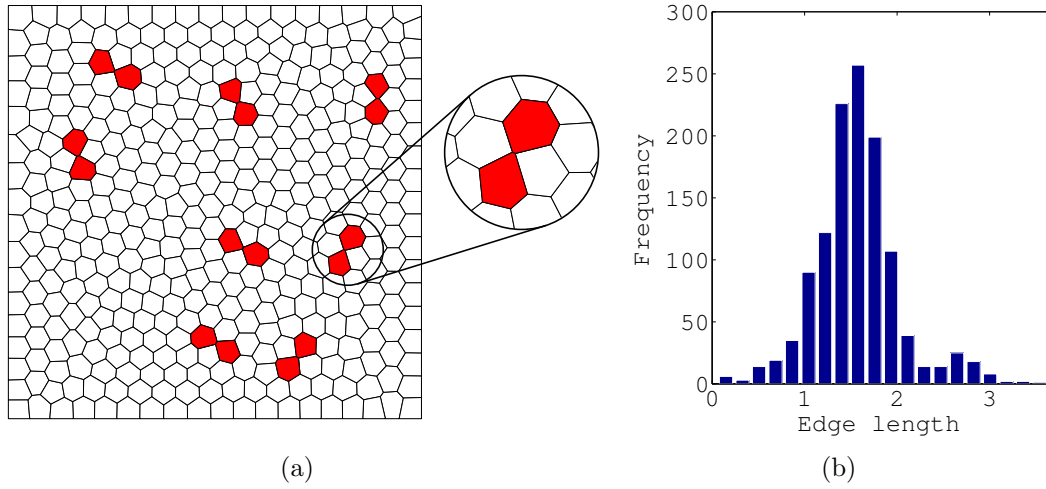


Figure 4.4: Two-dimensional CVT mesh, 50×50 domain. (a) Emphasized elements contains at least one edge that is smaller than 25% of the average edge length; (b) Edge length statistics: $\mu_{e.l.} = 1.66$, $\sigma = 0.50$.

4.3 Mesh Optimization Procedure

Let us separate the issues associated with boundary elements and with interior elements. Initially, let's review boundary-related issues.

PolyMesher is a 2D polygonal meshing algorithm based on the concept of Voronoi diagrams and CVT. Its code is publicly available for educational purposes [99]. It had been recently extended to 3D [101]. In this section we will investigate issues associated with CVT meshes and ways to prevent them.

In the PolyMesher code, the domain bounds are achieved by reflecting seeds outside the region of interest, thus creating the desired bounding box. However, there are some numerical issues in which perfectly straight domain boundaries are not achieved.

Moreover, in \mathbb{R}^2 , two close Delaunay nodes create small Voronoi edges (issue shown in Fig. 4.4) that can be trivially collapsed into only one node with subsequent appropriate nodes reindexing. Domain boundary imperfections are addressed collapsing nodes. Figure 4.5 illustrates the approach from 2D PolyMesher code. In 3D we cannot simply collapse nodes since non-planar faces are likely to appear.

The boundary elements also present issues. Reflections of the seeds near the boundary are oriented by a gradient that is computed by means of finite differences. Errors from the finite differences approximation do not create a perfect reflection, thus yielding slightly irregular boundary and therefore small

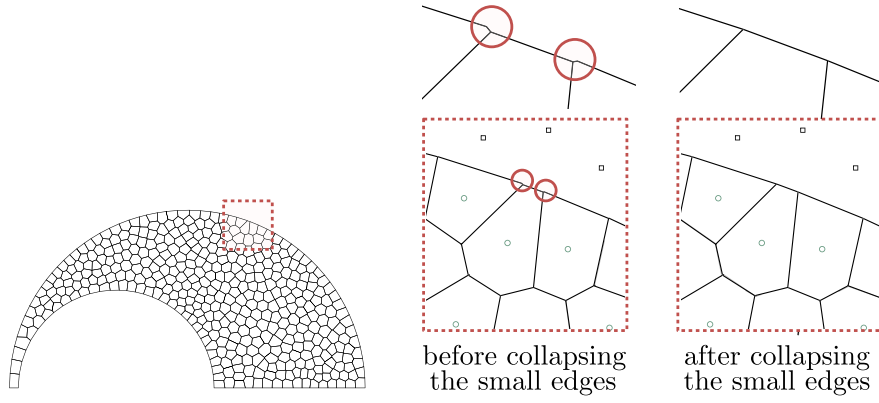


Figure 4.5: Two-dimensional PolyMesher approach dealing with small edges [99]. Nodal collapsing does not extend to three-dimensions without the introduction of non-planar faces' nodes.

edges (2D) and faces (3D). We can address these issues in two ways: first, for Cartesian domains, the numerical gradient can be corrected to the real value, since we know beforehand the faces normal vectors; and second, for non-Cartesian domains, we may rely on the Voro++ library [77]. It is an open source software library for the computation of Voronoi diagrams. Voro++ provides an easy-to-use bounding box feature that limits the design domain and consequent Voronoi diagrams to a specified domain geometry.

Before discussing the interior of the domain, let us investigate what are the origins of a short Voronoi edge. Consider the Delaunay vertices \mathbf{z}_i and a short edge $(\mathbf{x}_i, \mathbf{x}_j)$ in a Voronoi cell². The Voronoi vertices are the circumcenters of the Delaunay triangulation. A short Voronoi edge $(\mathbf{x}_i, \mathbf{x}_j)$ corresponds to two spatially close circumcenters \mathbf{x}_i and \mathbf{x}_j . As can be observed in Fig. 4.6, four almost co-circular Delaunay vertices yields two almost coincident circumcenters, i.e., a short Voronoi edge.

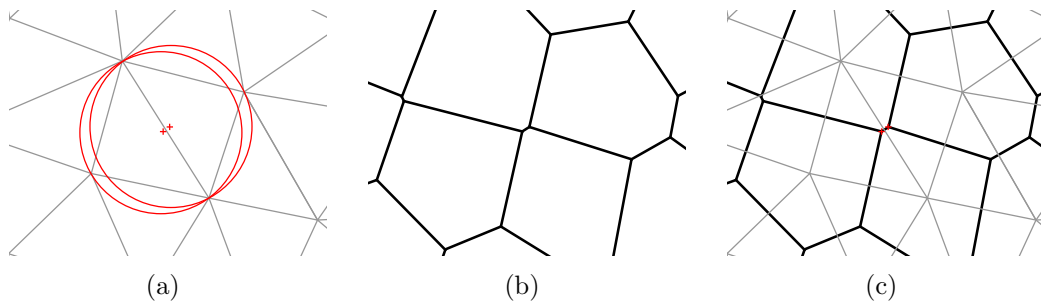


Figure 4.6: Origin of a small Voronoi edge. (a) Delaunay triangulation with two spatially close circumcenters; (b) Corresponding Voronoi diagram. Voronoi vertices are the circumcenters of Delaunay triangles; (c) Overlap of (a) and (b).

²Note that the Delaunay vertices are essentially the Voronoi cells generators, and thus, we're calling it \mathbf{z}_i .

Next, let's focus our attention on the interior of the domain. Here, for the mesh optimization, we have used an energy minimization formulation. The energy is a direct measure of the squared distance of the circumcenters and incenters of Delaunay triangles (tetrahedra in \mathbb{R}^3). It is based on the two-dimensional algorithm proposed by Sieger et al. [82] that consists in shifting the circumcenter as much as possible into the interior of the triangle, that is, the closest as possible to the incenter. This amounts to minimizing the squared distance of the incenter and circumcenter.

As the Voronoi diagrams are fully defined by their dual Delaunay triangulation, the degrees of freedom of the optimization algorithm is the set $\{\mathbf{z}_i\}_{i=1}^k$ of Delaunay vertices (i.e. Voronoi generators). Note that these are the same degrees of freedom of the computation of a CVT through Lloyd's method, thus leading to an elegant implementation of a mixed algorithm.

The energy function is represented by the squared Euclidean distance d^2 from a triangle's circumcenter to its incenter. This can be easily computed by using Euler's *triangle formula* $d^2 = R(R - 2r)$, where $R, r \in \mathbb{R}^3$ are the radii of the circumsphere and insphere, respectively (also called circumradius and inradius). Therefore, for every tetrahedron $t \in \hat{V}$:

$$E(\mathbf{z}_i, \dots, \mathbf{z}_k) = \sum_{t \in \hat{V}} R_t (R_t - 2r_t). \quad (4.8)$$

For a tetrahedron, its circumradius R and inradius r can be computed using the following equations:

$$R = \frac{\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}{2|a|} \quad \text{and} \quad r = \frac{3V}{A_1 + A_2 + A_3 + A_4}. \quad (4.9)$$

The quantities presented in Eqs. 4.9 can be defined by some geometrical interpretation. Consider the same tetrahedron from the previous chapter (Fig. 3.5) with vertices $\mathbf{v}_i = (x_i, y_i, z_i)$, $i = 1, \dots, 4$. Let A_i denote the area of the surface opposed to vertex \mathbf{v}_i and v_{ij} the vector going from \mathbf{v}_j to \mathbf{v}_i . The areas and volume can be computed as:

$$A_1 = \frac{1}{2} |v_{32} \times v_{42}|; \quad A_2 = \frac{1}{2} |v_{31} \times v_{41}|; \quad (4.10)$$

$$A_3 = \frac{1}{2} |v_{21} \times v_{41}|; \quad A_4 = \frac{1}{2} |v_{21} \times v_{31}|; \quad (4.11)$$

$$V = \frac{1}{6} \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{vmatrix}. \quad (4.12)$$

Further,

$$Dx = \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 & y_1 & z_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 & y_2 & z_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 & y_3 & z_3 & 1 \\ x_4^2 + y_4^2 + z_4^2 & y_4 & z_4 & 1 \end{vmatrix}; \quad Dy = \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 & x_1 & z_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 & x_2 & z_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 & x_3 & z_3 & 1 \\ x_4^2 + y_4^2 + z_4^2 & x_4 & z_4 & 1 \end{vmatrix}; \quad (4.13)$$

$$Dz = \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 & x_3 & y_3 & 1 \\ x_4^2 + y_4^2 + z_4^2 & x_4 & y_4 & 1 \end{vmatrix}; \quad a = \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}; \quad (4.14)$$

$$c = \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 & x_1 & y_1 & z_1 \\ x_2^2 + y_2^2 + z_2^2 & x_2 & y_2 & z_2 \\ x_3^2 + y_3^2 + z_3^2 & x_3 & y_3 & z_3 \\ x_4^2 + y_4^2 + z_4^2 & x_4 & y_4 & z_4 \end{vmatrix}. \quad (4.15)$$

The simple form of E presented in Eq. (4.8) allows us to evaluate and minimize the energy in an efficient way. The gradient is computed taking the partial derivatives of E with respect to all vertices $\{\mathbf{v}_i\}_{i=1}^4$, i.e.:

$$\frac{\partial E}{\partial \mathbf{v}_i} = 2 \sum_{t \in \hat{V}} \left[(R_t - r_t) \frac{\partial R_t}{\partial \mathbf{v}_i} - R_t \frac{\partial r_t}{\partial \mathbf{v}_i} \right]. \quad (4.16)$$

Let $\mathbf{V}^{(0)}$ be a stacked vector of the Delaunay triangulation vertices $\{\mathbf{v}_i\}_{i=1}^k$. In each iteration e , all vertices \mathbf{z} are shifted in the direction of the negative gradient $-\nabla E(\mathbf{V}^{(e)})$ by a step size h :

$$\mathbf{V}^{(e+1)} \leftarrow \mathbf{V}^{(e)} - h \nabla E(\mathbf{V}^{(e)}). \quad (4.17)$$

However, before the nodes are indeed shifted, their positions are verified whether they still lie inside the design domain. Sometimes, when the energy gradient is too steep, some nodes can get reallocated outside Ω . Line 8 of Algorithm 4.1 guarantees valid new vertex positions, which turns out to be an easy task due to the implicit representation.

For the gradient computation carried out in Eq. 4.16, the circumradius and inradius derivatives can be obtained applying the chain rule on Eq. (4.9):

$$\frac{\partial R}{\partial \mathbf{v}_i} = \frac{\partial R}{\partial Dx} \frac{\partial Dx}{\partial \mathbf{v}_i} + \frac{\partial R}{\partial Dy} \frac{\partial Dy}{\partial \mathbf{v}_i} + \frac{\partial R}{\partial Dz} \frac{\partial Dz}{\partial \mathbf{v}_i} + \frac{\partial R}{\partial a} \frac{\partial a}{\partial \mathbf{v}_i} + \frac{\partial R}{\partial c} \frac{\partial c}{\partial \mathbf{v}_i} \quad (4.18)$$

$$\frac{\partial r}{\partial \mathbf{v}_i} = \frac{\partial r}{\partial V} \frac{\partial V}{\partial \mathbf{v}_i} + \frac{\partial r}{\partial A_1} \frac{\partial A_1}{\partial \mathbf{v}_i} + \frac{\partial r}{\partial A_2} \frac{\partial A_2}{\partial \mathbf{v}_i} + \frac{\partial r}{\partial A_3} \frac{\partial A_3}{\partial \mathbf{v}_i} + \frac{\partial r}{\partial A_4} \frac{\partial A_4}{\partial \mathbf{v}_i}, \quad (4.19)$$

where each derivative can be computed as follows:

$$\begin{aligned}\frac{\partial R}{\partial Dx} &= \frac{1}{2|a|} \frac{Dx}{\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}; \\ \frac{\partial R}{\partial Dy} &= \frac{1}{2|a|} \frac{Dy}{\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}; \\ \frac{\partial R}{\partial Dz} &= \frac{1}{2|a|} \frac{Dz}{\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}; \\ \frac{\partial R}{\partial a} &= \frac{-\text{sign}(a)\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}{2|a|^2} - \frac{c}{|a|\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}; \\ \frac{\partial R}{\partial c} &= -\frac{a}{|a|} \frac{1}{\sqrt{Dx^2 + Dy^2 + Dz^2 - 4ac}}; \\ \frac{\partial r}{\partial V} &= \frac{V}{A_1 + A_2 + A_3 + A_4};\end{aligned}$$

And

$$\frac{\partial r}{\partial A_1} = \frac{\partial r}{\partial A_2} = \frac{\partial r}{\partial A_3} = \frac{\partial r}{\partial A_4} = \frac{-3V}{(A_1 + A_2 + A_3 + A_4)^2}.$$

The optimization algorithm described is presented in Algorithm 4.1.

Algorithm 4.1 Mesh Optimization

```

1: procedure MESHOPTIMIZATION(CVT Delaunay vertices, minimum step
    size  $\varepsilon$ , maximum iterations MaxIter)
2:   Initialize Delaunay vertices vector,  $\mathbf{V}^{(0)}$ 
3:   for  $k = 1, 2, \dots, \text{MaxIter}$  do
4:     Compute energy gradient,  $\mathbf{G} = \nabla E(\mathbf{V}^{(k)})$ 
5:     for  $h = 1, 1/2, 1/4, \dots, \varepsilon$  do
6:       if  $E(\mathbf{V}^{(k)} - h\mathbf{G}) < E(\mathbf{V}^{(k)})$  then
7:         if new vertices does not go outside domain then
8:           Update vertex positions,  $\mathbf{V}^{(k+1)} = \mathbf{V}^{(k)} - h\mathbf{G}$ 
9:           break
10:        end if
11:      end if
12:    end for
13:    if  $h == \varepsilon$  then
14:      break //Convergence achieved
15:    end if
16:  end for
17:  return Delaunay triangulation  $\mathbf{V}^{(k)}$ 
18: end procedure

```

4.3.1 Optimization Steps

In this section we quickly illustrate the steps of the mesh optimization algorithm presented. To show how the algorithm works, we present a very simple 2D example, where a single Voronoi cell is drawn and only one degree of freedom is present. The concepts illustrated here are trivially extended to 3D.

First, consider a starting Voronoi cell as result of a single optimization iteration, i.e., the single Voronoi generator is moved in the direction of the opposite gradient only once (Figure 4.7).

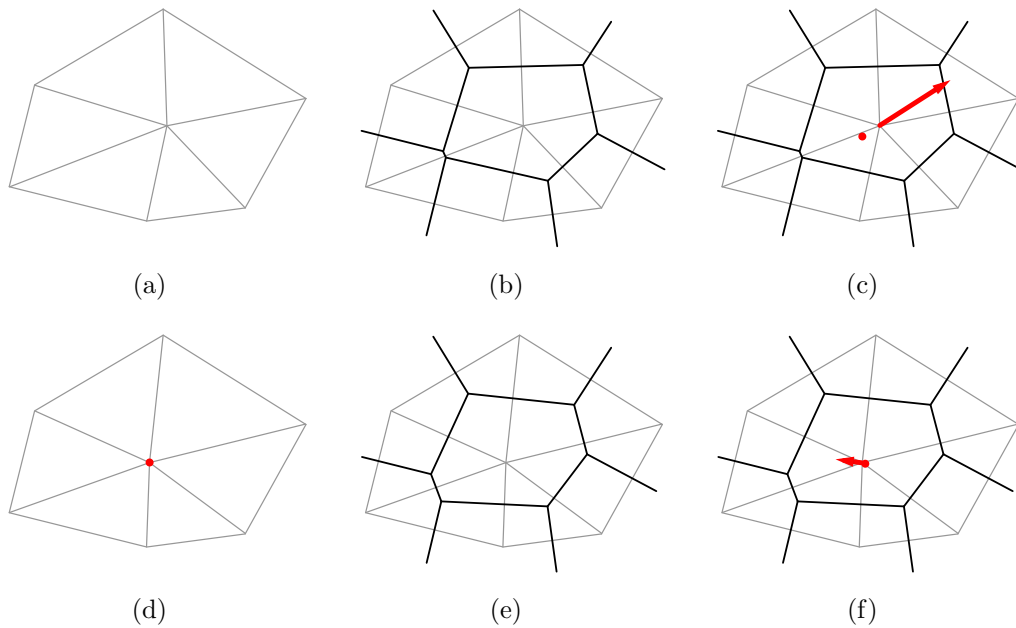


Figure 4.7: Meshing optimization procedure (a) Initial Delaunay triangulation in a given grid distribution; (b) The corresponding Voronoi cell associated with the Delaunay triangulation. Note the small edge; (c) The energy gradient vector and the location of the Voronoi seed of the next iteration; (d) The Voronoi seed is moved in the direction of the negative energy gradient; shown is the resulting Delaunay triangulation; (e) Corresponding Voronoi cell, with more uniform edges lengths; (f) Energy gradient and seed location of the next iteration. The proximity of two consecutive seed location indicates convergence. Arrow length represents the gradient vectors norm.

Now, consider the same step performed on the single-cell mesh from Fig. 4.7. Figure 4.8 shows the incircles (dashed) and circumcircles (continuous) of two Delaunay triangles that represents a short Voronoi edge. It can be noted that the resulting L2-norm distance between the circumcenter and incenter of the Delaunay triangles was minimized. The optimization process consists of shifting the circumcenter towards the incenter of the triangle/tetrahedra.

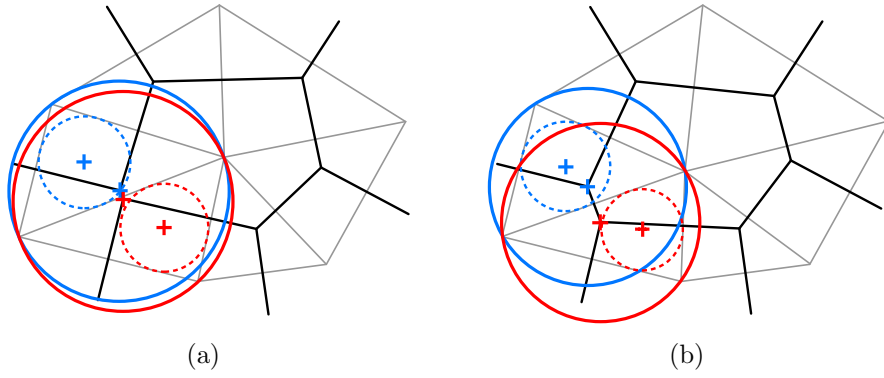


Figure 4.8: Circumcenters are shifted towards the incenters. Incircles are shown in dashed line and circumcircles in continuous line. (a) A Voronoi cell containing a small edge. (b) Distance between incenter and circumcenter minimized, resulting in a more uniform edge length distribution.

It should be noted that the single-step optimization convergence in this example was due to the nature of 1-DOF problem. Here, Delaunay boundary was constrained to a fixed (x, y) location, whereas in real meshes this constraint is not imposed.

4.4

Results and Concluding Remarks

The centroidal Voronoi tessellation through Lloyd's algorithms is normally used to obtain mesh regularity by removing excessive element distortion. However, using this approach, 2D elements can contain short edges and 3D elements small facets in the resulting Voronoi diagram as shown. Deformed elements can have a significant impact on the condition number of the stiffness matrix. It has been shown that a single bad element in the domain can be sufficient to ruin the numerical condition [81].

We assess the mesh improvement by analyzing the condition number of stiffness matrix related to a topology optimization problem with embedding (i.e., a linear elasticity problem solved through finite elements). The condition number for stiffness matrices related to a CVT and an optimized mesh will be computed and compared.

The condition number is a measure of how output error is influenced by input error. A problem with high condition number is called ill-conditioned, whereas a problem with a low condition number is said to be well-conditioned. The difficulties in solving $\mathbf{KU} = \mathbf{F}$ numerically are associated with the condition number of \mathbf{K} , specially for iterative solvers.

The condition number is defined as the maximum ratio of the relative error in the output over the relative error in the input. The relative error computation depends on the norm used. In Euclidean norm, we can express

the condition number in terms of its maximum and minimum eigenvalues [42]:

$$\kappa(A) = \left| \frac{\lambda_{\max}(\mathbf{K})}{\lambda_{\min}(\mathbf{K})} \right|. \quad (4.20)$$

The smallest eigenvalue λ_{\min} mainly depends on the smallest element, the largest eigenvalue λ_{\max} may become arbitrarily large even when a single badly shaped element is present [81].

With the proposed technique, the element size spread is limited and the elements are quite similar in size. Moreover, small edges are penalized and the smallest edge in the domain is increased by 9 orders of magnitude.

4.4.1

General Results

First, we present edge statistics of meshes of different sizes (Table 4.1) and histograms of the corresponding meshes (Fig. 4.9).

Table 4.1: Mesh statistics for the unitary cube domain, $\Omega = (0, 1)^3$.

Elements	Edges							
	CVT				Optimized			
	μ	σ	min	max	μ	σ	min	max
2k	3.8E−2	1.7E−2	4.5E−13	1.0E−1	4.1E−2	1.6E−2	4.6E−2	1.1E−1
4k	3.0E−2	1.3E−2	5.2E−13	7.4E−2	3.1E−2	1.1E−2	7.9E−4	8.8E−2
6k	2.6E−2	1.1E−2	1.3E−13	6.9E−2	2.7E−2	9.8E−3	6.7E−4	8.1E−2
8k	2.4E−2	9.8E−3	4.4E−13	6.7E−2	2.5E−2	9.0E−3	7.1E−4	7.3E−2
10k	2.2E−2	9.1E−3	6.6E−13	5.5E−2	2.3E−2	7.9E−3	2.8E−5	6.9E−2
12k	2.0E−2	8.5E−3	8.6E−13	5.1E−2	2.1E−2	7.3E−3	9.1E−5	6.4E−2
14k	1.9E−2	8.0E−3	9.1E−13	5.1E−2	2.0E−2	7.0E−3	6.0E−5	6.3E−2
16k	1.9E−2	7.8E−3	8.1E−13	5.4E−2	1.9E−2	6.1E−3	5.4E−4	5.6E−2

The meshes and examples provided here only serves for the purpose of showing the capabilities of the developed meshing optimization algorithm. All topology optimization problems presented throughout this work were executed by making use of the optimized meshes, unless otherwise noted.

By analyzing Tab. 4.1 and Fig. 4.9 and from further investigation within the interior of the domain, some conclusions can be drawn:

- Centroidal Voronoi cells' contain high number of small edges, with the smallest being in the order of 10^{-13} for a unitary cube domain;
- Edges smaller than 10^{-8} belongs to the boundary. These small edges are a product of the discussed finite differences-based reflection method. These edges are not a product of the Lloyds' algorithm nor it is a

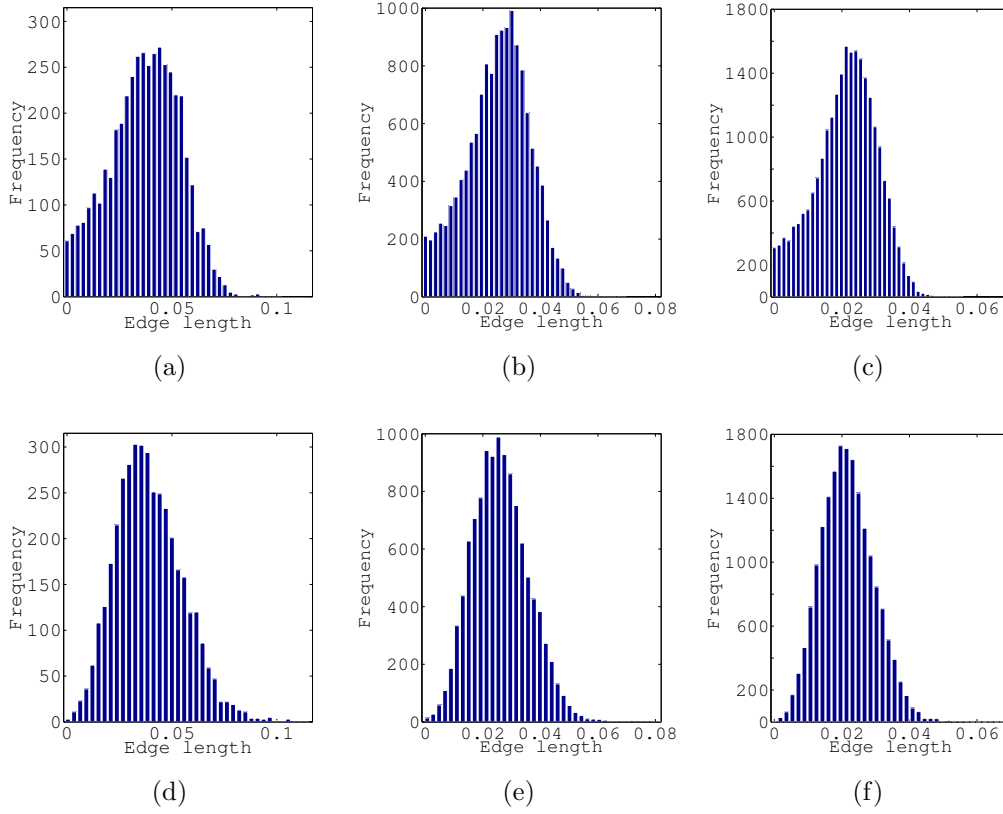


Figure 4.9: Histograms of edges lengths for the unitary cube. Top: CVT; bottom: optimized. (a) and (d) 2000; (b) and (e) 6000; (c) and (f) 10000 elements.

common characteristic of a CVT mesh, they are a result from the 3D implementation approach. Since these issues have been circumvented, the smallest edge that is a natural product of a CVT is in the order of 10^{-8} . Now, considering the CVT with perfect boundaries, the algorithm effectively increases the edges lengths 3 to 5 orders of magnitude;

- Most of the small edges³ are penalized. The few that are left are actually about 9 orders of magnitude larger than the original CVT and 3 to 5 orders larger than the CVT with fixed boundary. This amounts to the effectiveness of the algorithm in penalizing and removing small edges;
- The number of small edges is reduced by 95–98% after the optimization (see Tab. 4.2). As the number of elements (therefore, degrees of freedom for the optimization) increases, this percentage is also incremented;
- The standard deviation difference from a CVT mesh to a final optimized mesh is not significant. Large edges are not explicitly penalized and in fact there is no restriction for the introduction of even larger edges than

³As previously defined, a *small edge* is an edge smaller than 10% of the average edge length. Note that this measure would not make sense if the initial CVT was not created from a constant density function. If this were the case, differently sized element would be expected and the edge length would no longer be a suitable measure of quality.

on CVT. Small edges are somewhat compensated by the introduction of large edges, leading to a minimal difference in standard deviation.

Table 4.2: Effectiveness in removing small edges. The “small edges removed” measures the percentage of small edges present in CVT that are effectively removed by the optimization.

Elements	2k	4k	6k	8k	10k	12k	14k	16k
Small edges removed	97.5%	95.7%	95.7%	98.9%	98.0%	98.9%	98.6%	98.5%

Moreover, we note that small edges are bad features for the embedding of tetrahedra, since each (really) small edge can represent two slivers in the mesh, thus hurting numerical conditioning and solver robustness. Next, we assess mesh quality by analyzing the condition number associated with the matrix \mathbf{K} .

4.4.2 Linear Elasticity Problem

Now a linear elasticity problem through finite element analysis is executed. This problem is solved in every topology optimization iteration. Recalling Section 2.2.1, the stiffness matrix is given by

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV,$$

where the matrix \mathbf{B} is function of $\nabla \mathbf{N}$. Therefore, small edges will also hurt linear elasticity stiffness matrix.

Table 4.3 presents results for a beam fixed at one end and subject to a transverse load on the other end.

Analysis of the results confirms the effectiveness of the optimization algorithm on the meshes. Smallest edges have been increased by 9 orders of magnitude (4, on average, considering only the interior) and the condition number has been decreased by 2 orders of magnitude on average.

Table 4.3: Condition number associated with the stiffness matrix of the linear elasticity problem. The number of elements represents the number of polyhedrons.

Elements	Condition Number	
	CVT	Opt
2k	1.9E9	4.7E7
4k	4.7E9	3.4E7
6k	2.4E10	1.9E8
8k	1.3E10	1.0E8
10k	1.4E10	1.5E9
12k	1.0E10	5.1E8

5 Solving the System of Equations

The ultimate equation of interest when solving computational mechanics problems by means of the finite element method is $\mathbf{KU} = \mathbf{F}$, where \mathbf{K} is the global stiffness matrix and \mathbf{F} is the load vector. This set of equations is solved for the unknown nodal displacements \mathbf{U} . In linear FEA the cost of solving this system of equations rapidly overwhelms other computational phases. Therefore, it is important efficient handling of the \mathbf{K} matrix and the solution of the system.

Thus, solving this system of equations represents the single highest computational cost of a topology optimization program and it is, in fact, its bottleneck. Figure 5.1 shows the runtime breakdown of a typical topology optimization problem. The reader can observe that, as the mesh size increases, the time spent on the solution of the system of equations is dominant in the overall problem solution time.

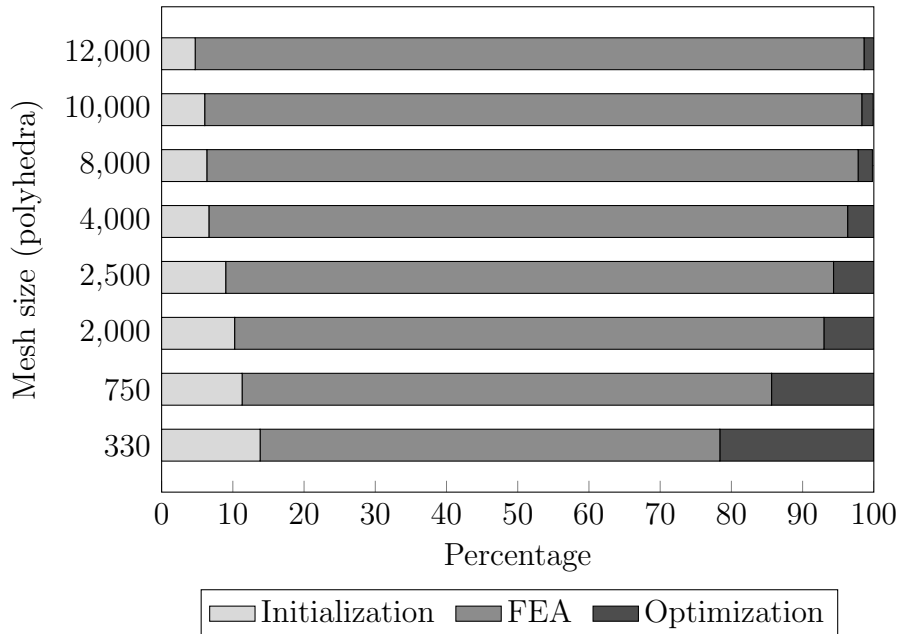


Figure 5.1: Runtime breakdown for different meshes size. Note how it is the most time consuming operation. Data obtained using the PARDISO [79] solver. Optimization was performed varying the p from 1 to 3.5, with increments of 0.5 and 100 iterations was allowed in each value of p .

In Fig. 5.1, the *Initialization* phase consists in read input file, create appropriate data structure, read the mapping, initialize the optimization variables, populate the finite element class (creating the local stiffness matrices), and apply boundary conditions. The *FEA* refers to the time spent assembling the global stiffness matrix and solving the sparse linear system, whereas *Optimization* includes the application of material model (SIMP), mapping of variables, objective and constraint function evaluation, computation of sensitivities and update of design variables. We note that the post processing was omitted¹ (it encompass stress computations and output file writing).

In this chapter are discussed ways to solve $\mathbf{KU} = \mathbf{F}$, represented generally by $\mathbf{Ax} = \mathbf{b}$. Issues regarding performance of some solvers are also reviewed.

For an in-depth discussion on efficient algorithms regarding different types of matrices and problems, the reader is referred to the book by Z. Bai et al. [6].

5.1 Introduction

Finite Element discretization for a typical topology optimization problem involves a large set of linear equations. Usually two types of solvers may be used to solve such system of equations: direct or iterative solvers.

Direct solvers are known for their robustness and generality. Direct methods solves the system of equations simultaneously, achieving the “exact” answer in one step. The downside of direct solvers is their memory requirements, which can limit the size of the problem that can be solved.

Iterative solvers, on the other hand, have significantly smaller memory requirements than a direct solver for the same sized problem. Iterative methods approach the solution gradually, rather than in a single, computationally-expensive, step. Depending on the tolerance used, an iterative solver will never get to the exact solution, but rather an *approximate* one. Iterative solvers’ performance, however, can be improved with the use of a suitable preconditioner.

In finite element applications, the matrix that arises from the system of linear equations is highly sparse. In addition, for linear elasticity, this matrix is symmetric and positive definite. Thus, using algorithms that take advantage of the sparsity of the matrix is essential to achieve high performance.

Before we start, let’s define some useful terms:

¹ It was neglected because its influence on the total solution time was minimal. In the most coarse case shown, it was 0.03% of total time. In the following mesh, less than 0.0001%. In the subsequent mesh, this time became even more insignificant.

- A *sparse* matrix is a matrix in which a very high percentage of its entries are zero;
- *Structured* matrices are the type of matrices that have enough structure that is worth taking advantage of it (e.g. Toeplitz, Vandermonde);

Both types of solvers, either direct or iterative, described in this chapter are investigated using their correspondent sparse algorithms, i.e., algorithms specifically designed to deal with sparse matrices.

5.2

Direct Sparse Solvers

Direct solvers is a category of numerical methods which solves the system of equation directly and exactly (up to the machine precision). They are very robust and they are always able to achieve a solution (as opposed to iterative solvers), if one exists. In this section, direct solvers are reviewed and their features discussed.

Direct solvers are often not efficient regarding memory usage. To circumvent these types of problems, *out-of-core* solver has been developed [43]. They perform I/O operations to/from the hard disk, decreasing the memory requirements. This is specially useful in large problems. However, it is important to note that using a machine with enough RAM, hence performing the factorization in-core, will greatly reduce the computational time. Traditional direct frontal algorithm [50] is an example of a direct solver. A frontal solver builds a LU or Cholesky decomposition of a sparse matrix given as a global matrix by assembling the matrix and eliminating equation only on a subset of elements. This subset is called the front and represents the region between the part already processed and the part yet to be processed. The advantage of a frontal solver is that the whole sparse matrix is never created explicitly.

Multifrontal solver represents an advance over the frontal solver. Its principle relies on the use of several independent fronts simultaneously. Multifrontal solvers efficient reordering techniques addresses issues related to memory and computational speed. A well known example of a multifrontal solver implementation is the UMFPACK package [22], which was investigated in this work.

Direct solvers for sparse matrices involve much more complicated algorithms than for dense matrices. A direct method for solving $\mathbf{Ax} = \mathbf{b}$ where \mathbf{A} is a dense matrix consists of the following steps:

1. Compute a factorization of \mathbf{A} ;
2. Use the factorizations to solve $\mathbf{Ax} = \mathbf{b}$.

However, given the structure of the stiffness matrix in finite element applications, the sparse matrix algorithms should be employed. A typical sparse solver consists of four main steps, instead of two for a dense matrix case [6]:

1. An **ordering step** that reorders the rows and columns such that the factors suffer little fill-in, or that the matrix has special structure, such as block-triangular form;
2. An analysis step or **symbolic factorization** that determines the nonzero structures of the factors and creates suitable data structures for the factors;
3. **Numerical factorization** that computes the **L** and **U** factors (or other appropriate factorization);
4. A **solve step** that performs forward and backward substitution using the factors.

Steps 1 and 2 involve only the graph associated to the matrices, i.e., connectivity of elements and how they are presented in the assembled global stiffness matrix, and therefore only integer operations. Step 3 includes floating-point operations and is the most time-consuming. Step 4 is roughly one order of magnitude faster than step 3 and also involves floating-point operations as well. The algorithm used in step 1 is independent of that used in step 3 or 4. However, algorithm used in step 2 is usually closely related to that of step 3. These steps are discussed in detail in the following sections.

5.2.1 Ordering

The form the matrix is presented is important regarding solver performance. In this step, the matrix is reordered to reduce fill-in during factorization. There are a number of methods to carry such task, and here are presented some of them.

During the factorization stage, the nonzero entries of the factorized matrices that were zeros in **A** are called fill-in. Thus, in the current stage, a permutation is created in order to reduce fill-in during factorization.

The stiffness matrix arrangement is a result of the unknown numbering (here, we're dealing with nodal numbering, as the nodal values corresponds to our unknowns). Therefore, the form the nodal numbers are assigned is important. It is always recommended that nodes connected to each other have numbers that are close to one another. Such practice leads to matrices in which the non-zeros terms are confined to a diagonal band, comprising the

main diagonal plus more “diagonals” on either side. Such matrices are called *band matrices* and can be conveniently stored.

The *bandwidth* of the matrix corresponds to the number of diagonals in which there is nonzeros terms. In the specific case of global stiffness matrix assembling, a matrix with large bandwidth corresponds to the fact that the nodes are linked over arbitrarily large distances. Sparse matrices, stored as a banded matrices, tend to be more efficient from a computational perspective and easier to store. On the other hand, the *profile* of a matrix is the sum, for each row, of the number of elements from the first non-zero until the diagonal. Note that global stiffness matrices in linear elasticity applications are inherently symmetrical.

In the conditions of the proposed embedding approach, Chapter 3, there is a special concern regarding nodal numbering, because new numbers are assigned as new nodes are being created. The reader should also note that in three-dimensional FEM application, the spread of the values in the matrix is inherently greater than in two-dimensional cases.

Recalling from the Chapters 3 and 4, upon the initial definition of the polyhedrons, the nodal numbering is rather satisfactory (see upper left part of the matrix in Fig. 5.3(a)). However, in order to define the tetrahedrons embedded into the polyhedrons, new nodes are created. Numbering of the newly created nodes continues after the last node of the original polyhedra mesh. This issue is illustrated in Fig. 5.2 by an arbitrary polyhedron.

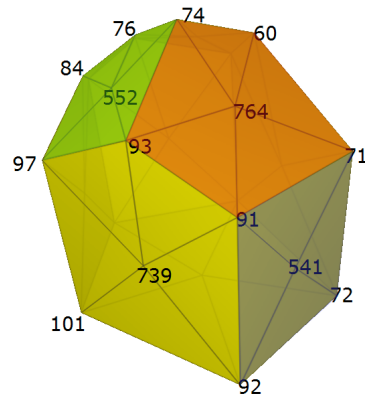


Figure 5.2: Example of post-embedding numbering. Note how the original numbers and face centroids numbers are distante from each other (similarly with polyhedron centroid node, which is 1089). The bigger the mesh, greater the difference will be.

The numbering illustrated by Fig. 5.2 leads to an odd sparse matrix: the square portion comprised by the first n original nodes is in a small bandwidth format (good); then the portion associated with face and element centroids are

more spread because lots of greatly spaced nodes are connected to them (bad) (see Figure 5.3(a)).

Fortunately, matrix processing techniques are available, decreasing storage and solution time, by rearranging the stiffness matrix in special structure formats, such as band matrices. Thus, the symmetrical matrix is converted into a band matrix form, with a small bandwidth, through the use of specific algorithms. The key idea of these algorithms is to rearrange the matrix by means of successive row and column permutation, moving all the nonzero elements as close as possible to the diagonal.

Figure 5.3(a) shows the stiffness matrix for a typical polyhedra with embedded tetrahedra mesh, before any reordering technique takes place. Fig. 5.3(b) and (c) shows the sparsity pattern after reordering techniques are applied.

There are many algorithms for profile and bandwidth reduction, such as Cuthill-McKee [20] and reverse Cuthill-McKee [39], Gibbs-Poole-Stockmeyer [40], Sloan [85] and Kaveh [56]. Some of these algorithms are treated in the following sections. The efficiency of the reordering algorithms have been assessed with different direct solvers.

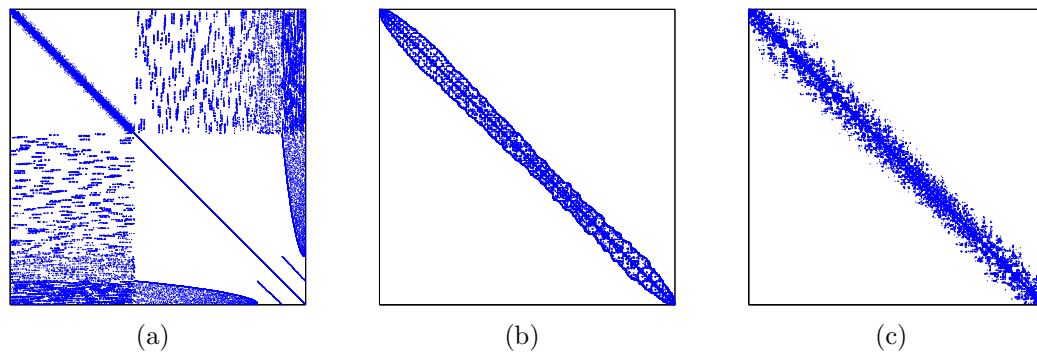


Figure 5.3: Matrix patterns after reordering. (a) Original matrix; (b) After RCM, 93.7% bandwidth reduction and 89.4% profile reduction; (c) After Sloan, 86.4% bandwidth reduction, 92.4% profile reduction.

Reverse Cuthill-McKee Algorithm (RCM)

The reverse Cuthill-McKee (RCM) algorithm, proposed by Alan George [39], is a heuristic method for bandwidth reduction in sparse matrices. It is based on the original Cuthill-McKee (CM) algorithm [20], differing in the resulting index numbers, which are reversed.

Both CM and RCM belongs to a class of reordering technique that proceeds by levels in a graph. In graph theory, it is to some extent related to breadth-first search. Its key idea is to visit the nodes by “levels”, in which the

level 0 is level of the starting node. In the starting node, the algorithm inspects all the neighboring nodes. Then, for each of these neighbors, it inspects their neighboring nodes which were unvisited, and so on.

The RCM algorithm also uses the idea of graph separator. A graph separator is a set of vertices that separates the original graph. Removing this set (or single) of vertices will result in two disjoint graphs. As a consequence, the matrix resulting from the CM/RCM ordering is a block-tridiagonal², with each block being the size of its level.

The CM algorithm consists in successively permuting rows and columns aiming to reduce its bandwidth. It is straightforward to see its advantage inspecting the resulting matrix bandwidth and profile.

Sloan Algorithm

While the RCM is a bandwidth reducing reordering, the Sloan algorithm aims in reducing its profile and the wavefront of a graph. Its principle relies on reordering of the indices assigned to each vertex.

The Sloan algorithm [85] uses an starting and end vertex, then a priority is assigned to each vertex, based on its distance from the start and end vertex. This priority is a weighted sum, taking into account a local criteria, i.e., its neighbors vertices, as well as global criteria.

The output of the algorithm is the new vertex ordering, which can be easily applied to a nodal numbering in a finite element mesh.

The obtained reordering with both RCM and Sloan has been applied in a polyhedra mesh with embedding. The resulting sparsity pattern of the global stiffness matrix is shown in Fig. 5.3.

Multilevel Nested Dissection Algorithm

The Nested Dissection Algorithm is internally used by the PARDISO solver. It is part of the METIS package, a package “*for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing ordering of sparse matrix*” [55].

The nested dissection algorithm is also based on the idea of graph partitioning and multilevel algorithms, similar to RCM, in which some vertices connects sets of vertices taken as *blocks*. The nodes in the separator are moved to the end of the matrix, and a similar process is applied recursively for each one of the other two parts. The multilevel nested dissection algorithm is quite effective in producing re-orderings that incur low fill-in [54].

²Block-tridiagonal matrix is similar to a tridiagonal matrix, but each component is a sub-matrix instead of a scalar.

The sparsity of the matrix that results from this algorithms does not show evident advantage. The advantage of the Nested Dissection is that the resulting factorization does not induce high fill-in.

Permutation vector

The algorithms described are used for sparse matrix reordering. Their output is a permutation vector, defined here as **perm**, of size *Number of Nodes*. This vector is used to reorder the sparse matrix.

Let \mathbf{K} be a matrix and $n\mathbf{K}$ be the reordered matrix. The permutation vector maps the i -th row (column) of \mathbf{K} into the **perm**[i] row (column) of $n\mathbf{K}$. We note that the permutation matrix is actually an identity matrix with its rows (or columns) permuted.

Sparse Matrix Storage Scheme

A sparse matrix, by definition, contains mostly zeros. Store every element of this matrix consumes too much memory, making a mid-scale problem impossible to be solved in practical time. There are some methods that efficiently stores sparse matrix. The efficiency of many solvers is determined primarily by the matrix-vector product, and, therefore, on the storage scheme used for the matrix.

The Compressed Row Storage (CRS) is the method of choice. Together with the analogous Compressed Column Storage (CCS), they are the most general. They don't make any assumptions regarding the sparsity pattern of the matrix. In fact, this format can even handle dense matrix. Furthermore, this format is especially interesting for this application: it is the standard matrix format used by commercial solvers such as PARDISO and UMFPACK.

The CRS format puts the nonzeros values in continuous memory location. It creates three vectors: **val**, for the floating-point numbers; **col_ind** and **row_ptr** for integers giving the matrix position of the nonzeros values. The vector **col_ind** stores the column indexes of the elements in the **val** vector. The **row_ptr** contains the location in the **val** vector that starts a row.

If a given matrix $n \times n$ (where n is typically the number of DOF), **row_ptr** size is n , **val** and **col_ind** size is $nnz + 1$, where nnz is the number of nonzeros. The storage savings using the CRS approach is significant: instead of storing n^2 elements as a full matrix, only $2nnz + n + 1$ storage locations are necessary.

As an example, consider the matrix \mathbf{M} defined by

$$\mathbf{M} = \begin{pmatrix} 10 & 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 2 & 2 & 7 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 7 & 8 & 0 & 0 \\ 0 & 1 & 9 & 0 & 0 & 12 \\ 0 & 0 & 5 & 0 & 1 & 0 \end{pmatrix}.$$

The arrays `val`, `col_ind` and `row_ptr` for the matrix \mathbf{M} is given below.

<code>val</code>	10	3	1	1	2	2	7	1	...	9	12	5	1
<code>col_ind</code>	1	2	5	3	4	5	6	5	...	3	6	3	5
<code>row_ptr</code>	1	4	8	9	12	15	17						

The CCS format is identical to the CRS format, except that the columns of \mathbf{M} , instead of rows, are stored. CCS format can be think of the CRS format for \mathbf{M}^T .

5.2.2

Symbolic Factorization

The symbolic factorization is the second step towards the solution of system of equations in a direct method. This step is also called analysis. It comprises the analysis of the nonzero structure; row- and column-permutations; and scaling to improve condition number.

The term symbolic implies that the factorization is processed only symbolic, i.e., without numerical values. It means that in the symbolic factorization phase, only the structure of the matrix is used to determine its factorization, that is, it only uses `col_ind` and `row_ptr` arrays.

5.2.3

Numerical Factorization

During the numerical factorization step, in most direct methods for sparse linear systems, the matrix is decomposed into:

- \mathbf{L} and \mathbf{U} matrices, lower and upper triangular, respectively, in case of LU decomposition, or;
- \mathbf{L} matrix, lower triangular, in case of Cholesky (or incomplete Cholesky) factorization.

Numerical factorization algorithms work with nonzeros terms of \mathbf{A} and the data structure resulting from the symbolic factorization phase in order to compute the \mathbf{L} and \mathbf{U} factorization.

LU factorization is usually employed in this step over Gaussian elimination since LU decomposition only uses matrix \mathbf{A} , whereas Gaussian elimination also uses right-hand-side \mathbf{b} . It is worth noting that this step is the most time-consuming in a direct solver.

LU Decomposition

If $\mathbf{A} = \mathbf{LU}$, then $\mathbf{Ax} = \mathbf{b}$ can be solved by :

$$\mathbf{LUx} = \mathbf{b} \quad \Leftrightarrow \quad \mathbf{Ly} = \mathbf{b}, \quad \mathbf{Ux} = \mathbf{y}.$$

As \mathbf{L} and \mathbf{U} are nonsingular lower/upper triangular matrices, then this is simple forward/backward substitution. By convention, \mathbf{L} has an unit diagonal, that is, it has 1's on the diagonal.

Suppose $\mathbf{A} \in \mathbb{R}^{n,n}$. The upper left $k \times k$ corners

$$\mathbf{A}_k = \begin{bmatrix} a_{aa} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{bmatrix} \quad \text{for } k = 1, \dots, n,$$

of \mathbf{A} are called the leading principal submatrices of \mathbf{A} .

Theorem *The square matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ has a unique decomposition $\mathbf{A} = \mathbf{LU}$ if and only if the leading principal submatrices of \mathbf{A} are nonsingular.*

Theorem *If \mathbf{A} is nonsingular, then a row permutation \mathbf{P} can be found in addition to \mathbf{L} and \mathbf{U} such that $\mathbf{PA} = \mathbf{LU}$ exists and is unique.*

If $\mathbf{PA} = \mathbf{LU}$, then instead of solving $\mathbf{Ax} = \mathbf{b}$, one can solve

$$\mathbf{P}^T \mathbf{LUx} = \mathbf{P}^T \mathbf{b} \quad \Leftrightarrow \quad \mathbf{Ly} = \mathbf{P}^T \mathbf{b}, \quad \mathbf{Ux} = \mathbf{y},$$

where the matrix \mathbf{P} is a permutation matrix describing the numerical pivoting used.

The cost of factorizing \mathbf{A} into matrices \mathbf{L} and \mathbf{U} is $O(n^3)$ for dense matrix \mathbf{A} . Once this factorization is completed, the cost of solving $\mathbf{LUx} = \mathbf{b}$ is just $O(n^2)$, since the cost of solving a triangular system scales as $O(n^2)$.

There are special algorithms that deals with large sparse matrix for LU-decomposition. These algorithms tries to find sparse factor \mathbf{L} and \mathbf{U} . They use reordering techniques to reduce fill-in, making them closely related to the choice of algorithms on the previous steps.

Cholesky Factorization

Cholesky factorization is a decomposition of a positive definite matrix $\mathbf{A} \in \mathbb{R}^n$ into the product of a lower triangular matrix and its transpose (conjugate transpose in case of $\mathbf{A} \in \mathbb{C}$).

It can be expressed as:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T. \quad (5.1)$$

If a reordering has not been applied before (during step 1), the matrix \mathbf{L} will probably contain high fill-in. However, using the reordering permutation can result in \mathbf{L} having very low fill-in.

The Incomplete Cholesky factorization, rather than the full Cholesky is also used as a preconditioner for iterative solver (discussed in section 5.3.2).

5.2.4

Forward/Backward Substitution

The last step towards the complete solution of the system of linear equations is the solution of the system of equation per se.

The factorization phase only uses the matrix \mathbf{A} , while the solving phase makes use of the factored form of \mathbf{A} and the right hand side \mathbf{b} to solve the linear system. For each different right-hand side, forward and backward triangular sweeps are executed.

The direct solvers investigated in this work are available commercially, dealing with the four stages described in this section. These solvers package are described in section 5.4.

5.3

Iterative Solvers

As mentioned, iterative solvers, as opposed to direct solvers, has significantly smaller memory requirements for the same sized problem. In general, iterative methods approach the solution gradually, rather than in a single, computationally-expensive step. Depending on the tolerance used, an iterative solver will never get to the exact solution, but rather an *approximate* solution.

The underlying principles of iterative solvers are the following: start with an initial guess \mathbf{x}_0 then iterate until a stop criteria is met, and finally return the final guess $\mathbf{x}^* = \mathbf{x}_k$. The stop criteria is typically that the residual (or error) is less than a specified tolerance.

The most popular class of iterative methods belong to the class of Krylov subspace methods. The Krylov subspace $\mathcal{K}^i(\mathbf{A}, \mathbf{r}_0)$ of dimension i , associated with a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ for a starting vector \mathbf{x}_0 with residual vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ is defined as the subspace spanned by the vectors

$\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{i-1}\mathbf{r}_0\}$. Examples of Krylov subspace methods include Conjugated Gradient (CG), Biconjugate gradient stabilized (BiCGSTAB), Generalized minimal residual (GMRES), among several others.

Different methods can be further classified depending on the type of matrix \mathbf{A} (see [6]), however it is not the intent of this work to do so. For a comprehensive and detailed discussion on iterative solvers for sparse systems, the reader is referred to Saad (2003) [78]. It includes discussion on projection methods, preconditioned approach, preconditioning techniques, parallel and multigrid methods and several other topics on iterative methods.

The performance of an iterative solver can be influenced by its preconditioner. In other words, all iterative solvers perform badly without a good preconditioner. Choosing a good iterative solver is, in reality, choosing a good preconditioner.

The Conjugated Gradient (CG) algorithm is one of the best known iterative techniques and was used in this work (the CG is computationally cheaper than the GMRES, for example). Which preconditioner to use with the CG is discussed in the following section.

5.3.1

Conjugated Gradient Method

The Conjugated Gradient (CG) is an algorithm for solution of linear system of equations. The CG is only suitable for system whose matrix is symmetric and positive-definite. The CG is treated here as an iterative method. We note that the CG is also an optimization method for unconstrained minimization problems.

The conjugated gradient algorithm is presented in Alg. 5.1. Its input is a guess vector \mathbf{x}_0 and the output is the final guess $\mathbf{x} = \mathbf{x}^*$. We should note that \mathbf{x}^* does not solve $\mathbf{Ax} = \mathbf{b}$, but rather approximate the exact within a given tolerance.

The CG algorithm takes advantage as it does not require access to matrix elements, it only performs matrix multiplications (direct solvers do need access to matrix' elements). As previously discussed, the CG algorithm (and any other iterative method in reality) may be inefficient without a preconditioner. In the following section we shall discuss matrix preconditioning for iterative solvers.

5.3.2

Preconditioner

In order to improve the efficiency of an iterative solver, a preconditioner must be used. A preconditioner is simply a means of transforming the original

Algorithm 5.1 Conjugated Gradient algorithm

```

1: procedure CONJUGATED GRADIENT( $\mathbf{x}_0$ )
2:    $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ 
3:    $\mathbf{p}_0 = \mathbf{r}_0$ 
4:    $k = 1$ 
5:   define tolerance  $tol$ 
6:   while  $\mathbf{r}_k \geq tol$  do
7:      $\alpha = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$ 
8:      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ 
9:      $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$ 
10:     $\beta = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$ 
11:     $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$ 
12:     $k = k + 1$ 
13:  end while
14:  return  $\mathbf{x}^* = \mathbf{x}_{k+1}$ 
15: end procedure

```

linear system which is likely to be easier to solve with an iterative solver [78]. The choice of a good preconditioner is extremely challenging, and by using a suitable preconditioner, both the efficiency and robustness of iterative techniques may be improved.

The first step in preconditioning is to find a preconditioning matrix \mathbf{M} . This matrix must satisfy a few minimal requirements, however, from a practical point of view, it is important to mention that solving the system $\mathbf{M}\mathbf{x} = \mathbf{b}$, required in each step of preconditioned algorithms, is computationally inexpensive. This is needed mainly because preconditioned algorithms will require a linear system solution with the matrix \mathbf{M} at each step.

How the matrix \mathbf{M} is obtained will be discussed in the specific preconditioning algorithms sections to follow. Here will be described how the preconditioner is applied to solve the original system of equations.

The solution of the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$ using a preconditioning matrix \mathbf{M} can be achieved through three different known ways. The preconditioner can be applied from the left:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}, \quad (5.2)$$

or, it can be applied to the right:

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{u} = \mathbf{b}, \quad \mathbf{x} \equiv \mathbf{M}^{-1}\mathbf{u}. \quad (5.3)$$

The above formulation amounts to making the changes of variables $\mathbf{u} = \mathbf{M}\mathbf{x}$ and thus solving the system with respect to the unknown \mathbf{u} . Finally, it is also

common the preconditioner be available in the factored form

$$\mathbf{M} = \mathbf{M}_L \mathbf{M}_R, \quad (5.4)$$

where \mathbf{M}_L and \mathbf{M}_R are typically triangular matrices. In this situation, the preconditioning can be split:

$$\mathbf{M}_L^{-1} \mathbf{A} \mathbf{M}_R^{-1} \mathbf{u} = \mathbf{M}_L^{-1} \mathbf{b}, \quad \mathbf{x} \equiv \mathbf{M}_R^{-1} \mathbf{u}. \quad (5.5)$$

It is important that preconditioners preserve features of the original matrix, such as symmetry. Even if \mathbf{M} is not readily available in factored form (5.4) in order to use (5.5), there's alternatives in preserving the symmetry of \mathbf{A} .

Preconditioned Conjugate Gradients

There is an extensive literature that deals with preconditioning for the Conjugate Gradients (CG) method, as well as others iterative methods [21,42]. In this section will be discussed two commonly used preconditioners: Diagonal (Jacobi) and Incomplete Cholesky. Both of them have been implemented, tested and evaluated.

For a matrix to be suitable for the CG method, the following conditions of a positive definite matrix must be satisfied:

1. $\mathbf{A} = \mathbf{A}^T$;
2. $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0, \quad \forall \mathbf{x} \neq 0$.

The Preconditioned Conjugate Gradient (PCG) is a modified version of the CG, dealing with preconditioned residues. The algorithm is shown in Alg. 5.2.

Jacobi Preconditioner (Diagonal)

The Jacobi preconditioner is also called *diagonal*. It is derived from the Jacobi iterative method. It is the simplest preconditioner available and its implementation is straightforward. It consists of just the diagonal of \mathbf{A} :

$$M_{ij} = \delta_{ij} A_{ij}, \quad (5.6)$$

where i, j represents the row and column, respectively, and δ_{ij} is the Kronecker delta. Therefore, the preconditioning matrix \mathbf{M}^{-1} can be easily written as:

$$M_{ij}^{-1} = 1/M_{ij}.$$

Algorithm 5.2 Preconditioned Conjugated Gradient algorithm [42]

```

1: procedure PCG( $\mathbf{x}_0$ )
2:    $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ 
3:    $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$ 
4:    $\mathbf{p}_0 = \mathbf{z}_0$ 
5:    $k = 1$ 
6:   define tolerance  $tol$ 
7:   while  $\mathbf{r}_k \geq tol$  do
8:      $\alpha = \frac{\mathbf{r}_k^T \mathbf{z}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$ 
9:      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$ 
10:     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha \mathbf{A} \mathbf{p}_k$ 
11:     $\mathbf{z}_{k+1} = \mathbf{M}^{-1} \mathbf{r}_{k+1}$ 
12:     $\beta = \frac{\mathbf{z}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{z}_k^T \mathbf{r}_k}$ 
13:     $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta \mathbf{p}_k$ 
14:     $k = k + 1$ 
15:  end while
16:  return  $\mathbf{x}_{k+1}$ 
17: end procedure

```

Incomplete Cholesky

The incomplete Cholesky factorization is similar to the “full” Cholesky. In Cholesky factorization, the original matrix \mathbf{A} is decomposed into matrix \mathbf{L} , in which the product $\mathbf{L}\mathbf{L}^T$ is equal to \mathbf{A} . However, this may lead to a matrix \mathbf{L} with high fill-in. The incomplete Cholesky (IC) factorization, on the other hand, decomposes \mathbf{A} into $\tilde{\mathbf{L}}$, with $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T \approx \mathbf{A}$ and no fill-in.

To achieve no fill-in, the difference between the two algorithms is that IC only visits positions of \mathbf{A} that are different from zero. The left-looking incomplete Cholesky factorization algorithm is show in Alg. 5.3. In this algorithm, the matrix \mathbf{L} is represented by the lower triangular of the final matrix \mathbf{A} .

5.4

Packages

In this section some mathematical libraries are presented and discussed. These libraries include routines that ultimately solve the linear system of equation $\mathbf{KU} = \mathbf{F}$.

Algorithm 5.3 Incomplete Cholesky algorithm

```

1: procedure INCOMPLETE_CHOLESKY
2:   for  $k = 1 : n$  do
3:      $A(k, k) = \sqrt{A(k, k)}$ 
4:     for  $i = k + 1 : n$  do
5:       if  $A(i, k) \neq 0$  then
6:          $A(i, k) = A(i, k)/A(k, k)$ 
7:       end if
8:     end for
9:     for  $j = k + 1 : n$  do
10:      for  $i = j : n$  do
11:        if  $A(i, j) \neq 0$  then
12:           $A(i, j) = A(i, j) - A(i, k)A(j, k)$ 
13:        end if
14:      end for
15:    end for
16:  end for
17: end procedure

```

5.4.1**Basic Linear Algebra Subprograms (BLAS)**

Basic Linear Algebra Subprograms (BLAS) are a set of low-level kernel Fortran subroutines that performs common algebra operations [60]. It has 3 levels, with capabilities described below.

Level 1 Published in 1979, operates on only one of two vectors (or columns of a matrix) at a time. That is, performs vector-vector operations in the form:

$$\mathbf{y} = \alpha \mathbf{x} + \mathbf{y}.$$

Level 2 Published in 1984, operates on larger portions of entire matrices. It was capable of solving $\mathbf{T}\mathbf{x} = \mathbf{y}$ for \mathbf{x} , with \mathbf{T} begin triangular. Perform matrix-vector operations, such as generalized matrix-vector multiplication in the form:

$$\mathbf{y} = \alpha \mathbf{A}\mathbf{x} + \beta \mathbf{y}.$$

Level 3 Published in 1990, performs matrix-matrix operations in the general form:

$$\mathbf{C} = \alpha \mathbf{A}\mathbf{B} + \beta \mathbf{C}.$$

BLAS subroutines are usually standard in many linear algebra libraries and routines packages. For example, the LAPACK package, which uses BLAS, is used by MATLAB since version 6.0, released in late 2000 [66].

5.4.2

Linear Algebra Package (LAPACK)

The Linear Algebra Package (LAPACK) is a library for numerical linear algebra applications. LAPACK provides routines for solving systems of simultaneous linear equations, least-squares solution of linear systems of equations, eigenvalue problems and singular value problems. It also provides tuned LU, QR and Cholesky factorizations. Dense and banded matrices are handled, but not general sparse matrices [1].

LAPACK is designed to exploit the Level 3 BLAS. LAPACK subroutines are written in a way that as much as possible of the computation is performed by calls to the BLAS.

5.4.3

UMFPACK

UMFPACK consists in a set of routines for solving systems of linear equations, $\mathbf{Ax} = \mathbf{b}$, when \mathbf{A} is sparse and unsymmetric. Its algorithm is based on the Unsymmetric-pattern MultiFrontal method [24,25] and direct sparse LU factorization. Multifrontal methods are a generalization of the frontal methods developed primarily for finite element problems [5].

In the present context, the UMFPACK solver was used as a sparse direct solver.

UMFPACK automatically selects different strategies for column and row pre-ordering depending on the sparsity nature of the matrix. The solver has built-in fill reducing algorithms such as COLAMD [26] and AMD (approximate minimum degree) [23]. During factorization phase, a sequence of dense rectangular matrices is created for factorization. A supernodal column elimination tree is generated in which each node in the tree represents a frontal matrix. The chain of frontal matrices is factorized in a single working array [57].

5.4.4

Intel Math Kernel Library (MKL)

Intel Math Kernel Library (MKL) is a math library optimized for the Intel architecture. Intel MKL improves performance with math routines for applications where large computational problems are solved.

The Intel MKL includes a number of groups of subroutines, such as BLAS, LAPACK routines; PBLAS routines; Vector Mathematical Library; Fast Fourier Transform functions; direct and iterative sparse solver routines; and several others [49]. For the topology optimization tool developed, the MKL package have been used mainly because of its direct solver capabilities.

MKL provides sparse solver software to solve real or complex, symmetric, structurally symmetric or nonsymmetric, positive definite, indefinite or Hermitian square sparse linear system of algebraic equations. The PARDISO is the solver used in this work from the Intel MKL. It was originally developed by the Department of Computer Science at the University of Basel [3] and later incorporated into Intel MKL.

Parallel Direct Sparse Solver (PARDISO)

The PARDISO package is a high-performance, robust, memory efficient, and easy to use software package for solving large sparse symmetric and unsymmetric linear systems of equations on shared memory multiprocessors [2, 79]. As the name implies, it is a direct solver.

The PARDISO solver performs the same four tasks described previously:

- analysis and symbolic factorization;
- numerical factorization;
- forward and backward substitution including iterative refinement;
- termination to release all internal solver memory.

The PARDISO solver can store the solution *out-of-core*, which means that they can offload some of the problem onto the hard disk. This feature is specially interesting in large problems. The reader is referred to [79] for an in-depth discussion of PARDISO's inner working.

5.5

Performance Results and Conclusions

The performance of different types of solvers is investigated and the result is presented in this section. It is explored direct and iterative solvers, reordering techniques and the role of preconditioning for an iterative solver.

We show results for two families of mesh:

1. Regular 6-sided polyhedron, essentially a brick but treated as a polyhedron. However, embedded with 6 tetrahedrons only. We note that this is a special case of embedding, since we know a priori the shape and regularity of the element. In this case, only 6 tetrahedrons are embedded in each hexahedron and no new nodes are created (see Fig. 5.4);
2. Unstructured arbitrary polyhedral mesh (as treated in Chapter 4), embedded with tetrahedrons following Algorithm 3.1.

The machine used was an Intel Core i7 Extreme X980 6-core at 3.33Ghz, with 24GB DDR3 RAM, using Microsoft Windows 7.

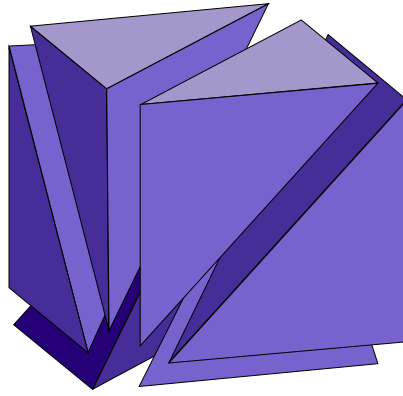


Figure 5.4: Special embedding performed on regular hexahedron, with no new node creation.

Before we begin, let's lay some conclusion/requirements of the two types of solvers studied [7]:

- Direct solvers
 - Always work, for any invertible matrix;
 - No need to think about preconditioners;
 - Fast, as long as it has available RAM;
 - Memory and CPU time becomes an issue for larger problems.
- Iterative solvers
 - May not work even for invertible matrix;
 - Can solve very large problems;
 - Easily parallelized;
 - Choice of preconditioner depends on the problem.

5.5.1 Reordering

Now, let's address the reordering influence on the solution of the topology optimization problem. The reordering techniques presented are tested on the PARDISO and UMFPACK direct solvers. Unstructured polyhedral meshes were used.

Figure 5.5 shows the performance of a given problem on the PARDISO solver. As expected, the only difference is in the solution of the sparse linear system of equation. The original matrix, similar to Fig. 5.3(a) is not effective, as the reader could have imagined. PARDISO's internal reordering (Nested Dissection) is two orders of magnitude faster than RCM and Sloan. The reason is that, although RCM and Sloan's output matrices *looks* better by exhibiting banded pattern, after factorization they do not guarantee low fill-in. Nested Dissection, on the other hand, does.

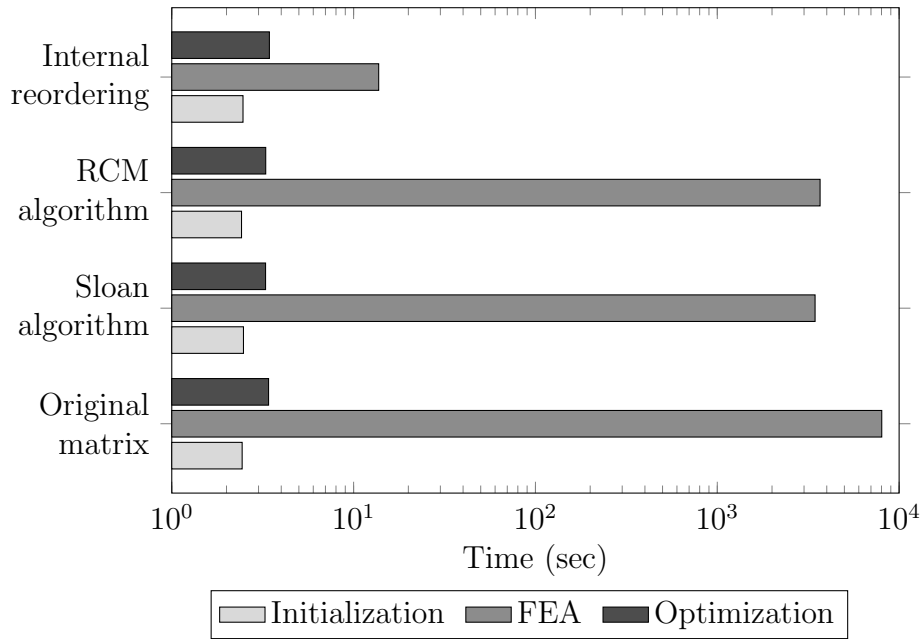


Figure 5.5: Runtimes for each main part for the PARDISO package. Data from a mesh composed of 324 polyhedra, 18,476 tetrahedrons and 4,066 nodes.

UMFPACK package presents faster runtimes (Fig. 5.6) than PARDISO (Fig. 5.5), except for PARDISO's own reordering, which is 2 orders of magnitude faster. Runtimes presented are from a simple benchmark problem, and 100 optimization iterations per penalty value are used, which is increased from 1.0 to 3.5 with increments of 0.5 (continuation).

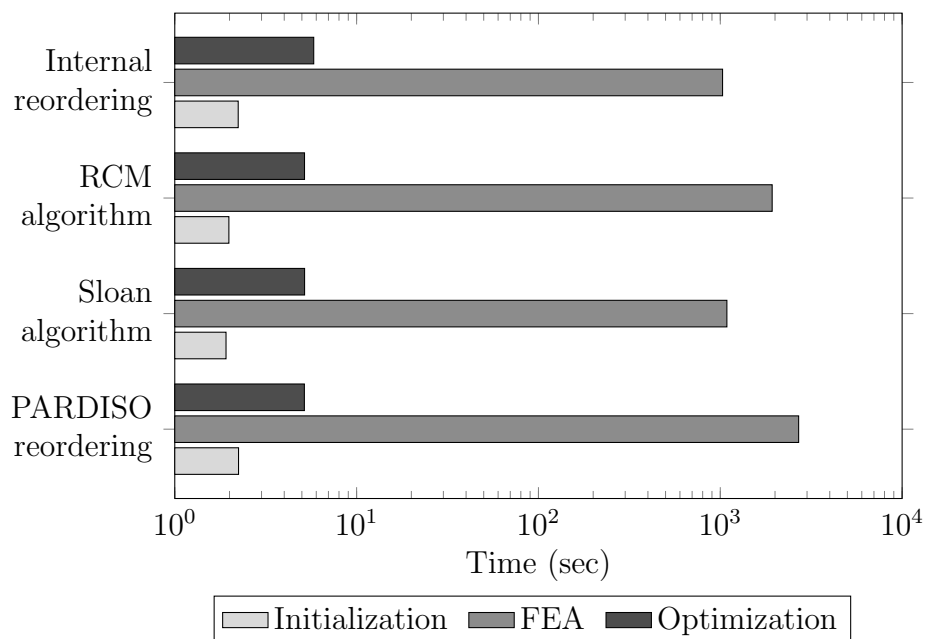


Figure 5.6: Runtimes for each main part for the UMFPACK package. Data from a mesh composed of 324 polyhedra, 18,476 tetrahedrons and 4,066 nodes.

Tasks executed within each group, namely, *initialization*, *FEA* and *optimization* are same outlined in the beginning of the Chapter. Again, post-processing is neglected (3 orders of magnitude smaller than *Initialization*). We note that in all cases the system had enough RAM for the case being executed. We analyzed runtime as a measure of performance because although high fill-in (for example) would demand more RAM, it would also demand more processing power, thus resulting in higher computational time

5.5.2 Preconditioning

The PCG algorithm is evaluated with the Jacobi and Incomplete Cholesky (IC) preconditioner, as well as without any preconditioning (that is, essentially CG).

The meshes used to evaluate preconditioning and the performance of iterative and direct solvers are structured tetrahedral meshes at first (as shown in the beginning of this section) and then we evaluate a polyhedral mesh. The structured meshes' details are presented in Table 5.1.

Table 5.1: Mesh information.

Mesh Size	Elements	Nodes	DOFs
4	4,608	1,125	3,375
8	36,864	7,497	22,491
12	124,416	23,725	71,175
14	197,568	36,975	110,925
16	294,912	54,417	163,251
20	576,000	104,181	312,543
24	995,328	177,625	532,875

In Table 5.2 we present the number of iterations necessary to converge for a single finite element analysis. The convergence numerical tolerance used is the same for every preconditioner. In Table 5.3 we show the average number of iterations for the first 10 topology optimization loops. In this analysis, the stiffness matrix is updated at every iteration, resulting in a new problem, which is then solved.

Table 5.2: Number of iterations for the solution of $\mathbf{Ax} = \mathbf{b}$ using the PCG solver. Values not shown mean that convergence was not achieved in reasonable time.

Mesh size	Preconditioner		
	IC	Jacobi	none
4	48	285	2534
8	93	560	4714
12	139	832	-
14	162	963	-
16	184	1214	-
20	230	1492	-
24	275	1771	-

Table 5.3: Average number of iterations for the solution of the first 10 iterations of a topology optimization problem using the PCG solver.

Mesh size	Preconditioner	
	IC	Jacobi
4	49.6	310.5
8	99.6	620.4
12	149.5	924.6
14	174.0	1064.6
16	198.6	1354.1
20	247.5	1679.8
24	296.3	1998.7

From the values presented we conclude that the IC preconditioner is roughly 6 times faster on structured meshes and becomes even faster as mesh sizes increases.

Now, let's investigate how the PCG iterations behave as the topology optimization evolves. Now, we use an optimized unstructured polyhedral mesh. For this analysis, we set the penalty parameter to $p = 3$ and perform optimization until convergence is achieved. As expected from the structured

meshes results, the absolute number of iterations using IC is higher than Jacobi. Figure 5.7 shows the obtained results.

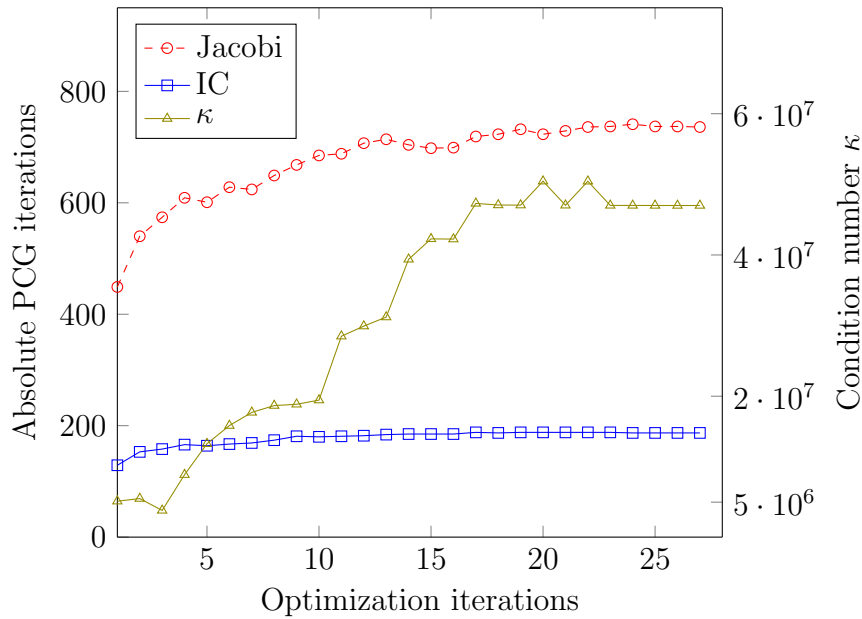


Figure 5.7: Number of PCG iteration with different preconditioner, as well as condition number at each optimization iteration.

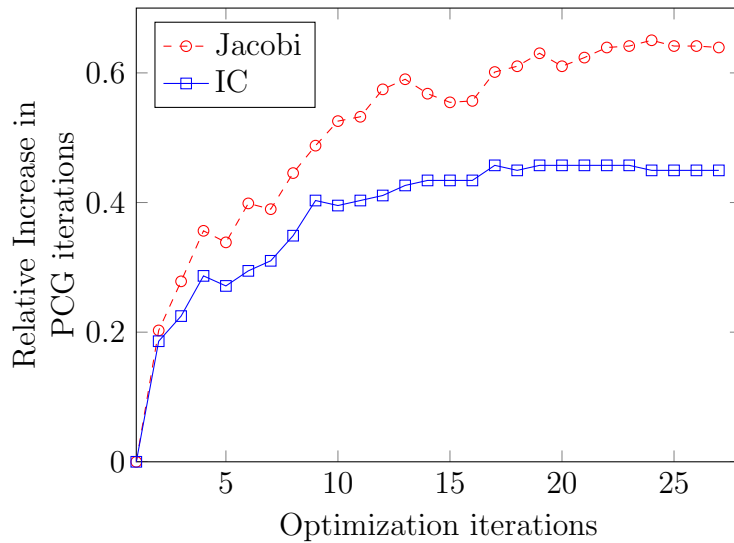


Figure 5.8: Relative increase (each preconditioner relative to itself) in the number of PCG iterations needed to converge in a typical problem using unstructured polyhedral meshes

Further, we note that the number of iterations increased 45% when using IC, whereas using Jacobi the number of PCG iterations increased up to 64% (Fig. 5.8) when solving the last problem (iteration 28). We see that, associated to the number of PCG iterations, the condition number also increases in a

similar fashion. Iterative solvers are sensitive to matrix conditioning. PCG with Jacobi showed more sensitivity than the IC.

Additionally, we note that the increase in the number of iterations is due to matrix conditioning, which, in turn, is dependent upon the material tensor in a final sense. After the very first optimization iteration, several elements in the domain are assigned to “no material”, which is represented by the very compliant material of tensor $\varepsilon \mathbf{C}$.³ In the next iteration, the new stiffness matrix takes into account the value of ε and this results in a less well-conditioned matrix, with increased condition number, resulting in an increase in the number of iterations of the iterative solver.

In fact, the matrix condition number is closely related to the density bound used in the topology optimization formulation, whether is by the Ersatz parameter or by imposing a ρ_{\min} . If we let the Ersatz parameter assume a lower bound (say, for example, 10^{-9} , instead of the 10^{-4} used throughout this work), the stiffness matrix condition number may increase by 5 orders of magnitude (see Fig. 5.9).

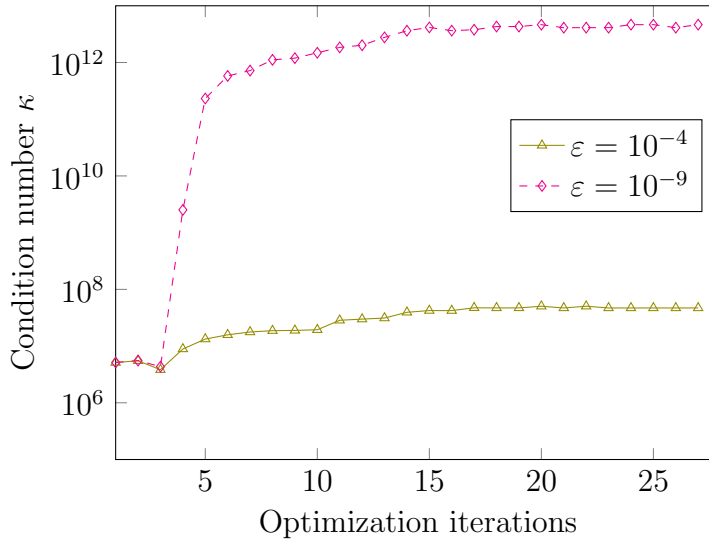


Figure 5.9: Ersatz parameter ε influence in the conditioning of the stiffness matrix as the topology optimization evolves.

The spike on the $\varepsilon = 10^{-9}$ curve occurs in the third iteration, representing the first appearance of “void” in the structure as the move limit set in the OC method prevent an element of becoming “void” in the very first iteration.

³We remind that ε is the Ersatz parameter used to avoid singularities in the stiffness matrix by setting a compliant material. The Ersatz value is usually set to 10^{-4} and $\lim_{\varepsilon \rightarrow 0} \mathbf{K}_e$ is no stiffness (singularity in the global matrix).

5.5.3 Type of Solver

Now, let's compare the best direct solver plus reordering combination with an iterative solver (with discussed preconditioners). The information from meshes evaluated are presented in Table 5.1 and the results in Fig. 5.10.

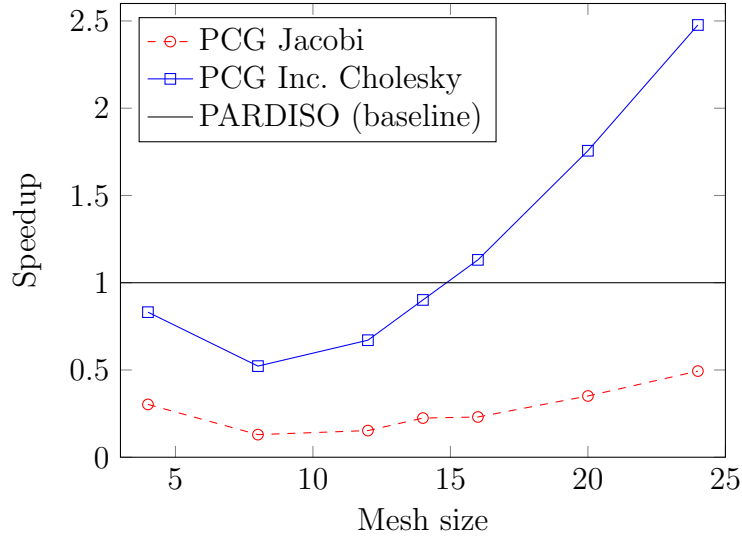


Figure 5.10: Performance of different solver in the solution of differently sized problems. For the “Mesh size” measure, see Tab. 5.1.

Some conclusions can be drawn from the experiments:

- The PARDISO is superior than UMFPACK in solving sparse linear system of equations associated with structured meshes, that is, without deformed elements (except in very coarse cases, which are not of interest);
- PARDISO performs well, but the well-conditioned nature of the stiffness matrix for structured problems lead to great performance of PCG as well;
- From 130k DOFs onwards, PCG solver with Incomplete Cholesky decomposition performed better;
- The low condition number associated to structured meshes makes iterative solvers a better choice.

However, we should note that although these results looks promising, it only applies to structured meshes. We remind that the PARDISO solver is parallel and the iterative solvers investigated are serial. From our investigation, the PCG solver was faster than PARDISO because the matrices \mathbf{K} were *very* well-conditioned, due to the structured and uniformity nature of the polyhedrons (regular hexahedrons, in this case), leading to a very good condition number. Arbitrary polyhedral meshes, on the other hand, present somewhat deficient condition number, therefore making the PARDISO solver faster than any serial solver investigated.

Further, it should be noted the following: the condition number of the problem is also dependent on the material (the density). In optimization loops next to the end of the problem, we usually have a considerable volume with density on the lower bound, set by Eq. (2.6). This number counts towards the stiffness matrix, and plays an important role on the convergence rate of iterative solvers.

We conclude this Chapter by pointing out that our mesh optimization improved the condition number, which leads to faster convergence by PCG with IC. Although on structured meshes the PCG can easily outperform PARDISO, on truly unstructured cases, the matrix related to the polyhedral mesh is still not well-conditioned. Hence, the PARDISO still beats PCG in terms of runtime, largely due to its parallel implementation, and will be the solver used in the problems presented in the next Chapter.

6 Numerical Results

In this Chapter are presented numerical results that demonstrate the capabilities of the proposed framework. Meshes based on arbitrary polyhedral elements are less biased than structured meshes due to its geometry. Additionally, they do not exhibit the checkerboard pattern and do not allow node to node connections.

The obtained topologies shown are colored with respect to the density, unless otherwise noted (all elements with density lower than 0.1 were hidden). Density scale will be provided. In some problems, several figures with plane cuts and projections will be provided to aid visualization. In all examples shown, SIMP with continuation was employed, varying the penalty parameter from 1 to 3.5, with increments of 0.5.

6.1 Three-dimensional Validation

We first begin with mimicking a 2D problem by extruding the third dimension, thus creating a 3D mesh. The objective of this investigation is to (1) evaluate our meshing technique and (2) check the final topology with known results from two-dimensional literature.

For this experiment, we consider the Michell problem [93], with domain, loads and boundary conditions shown in Fig. 6.1(a), while the analytical solution is shown in 6.1(b).

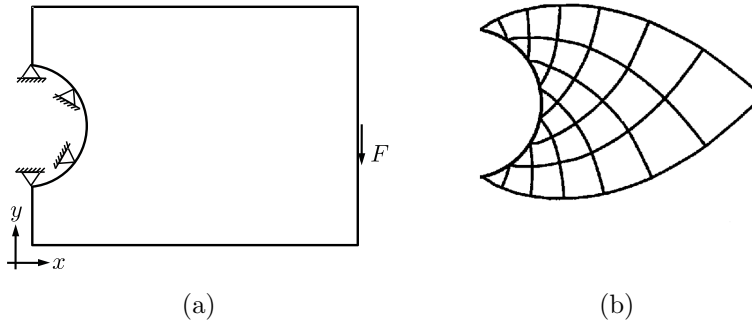


Figure 6.1: Michell beam problem. (a) Extended domain, loads and boundary conditions. (b) Analytical solution from Reference [93].

We set the final structure to have 8.5% of the initial volume. The structure was extruded from a 2D mesh. Figure 6.2 shows the obtained final topology, matching the analytical optimal topology. We note that the Michell example exhibits orthogonal structural members, a common feature in optimal topologies of several type of problems.

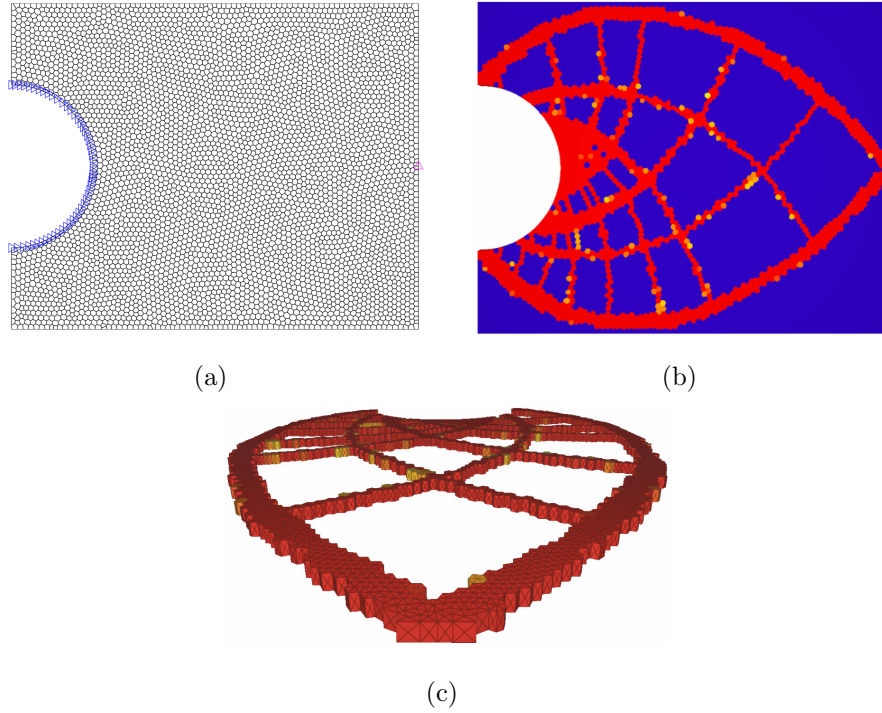


Figure 6.2: Michell beam problem converged topology. Only 1.5% of the polyhedrons presented intermediate density; red elements are solid and blue are void. (a) Initial polyhedral mesh; (b) final result (final compliance 3317.1) with 5000 design elements. (c) Side view showing the extruded mesh.

6.2 Benchmark Problem

Now, we deal with true three-dimensional problems. We first evaluate a benchmark problem, in order to assess the embedding capabilities, regarding physical sense of final topology, numerical pathologies and resolution of structural members.

The benchmark problem [75] used is a cantilever beam fixed at one end and subject to a transverse concentrated load at the opposite end. The problem's boundary conditions is illustrated in Fig. 6.3. The obtained result is presented in Figs. 6.4 and 6.5 . We note that the arrangement of structural members in the obtained solution is similar to the Michell's problem, indicating orthogonal structural members. We also note that the cross section converged to the well known I beam. In this problem, only 0.2% of the polyhedrons have

intermediate density. Then, to aid the visualization of a 3D topology without the need of several figures, we have colored it based on the nodal displacement field along the z axis.

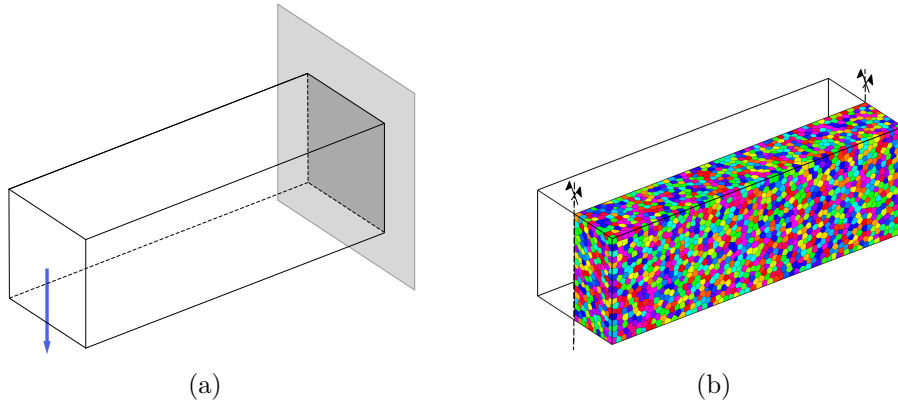


Figure 6.3: Cantilever beam subject to a transverse load problem, with domain $L \times L \times L$. (a) Geometry, load and boundary conditions; (b) Illustration of a symmetric sample mesh.

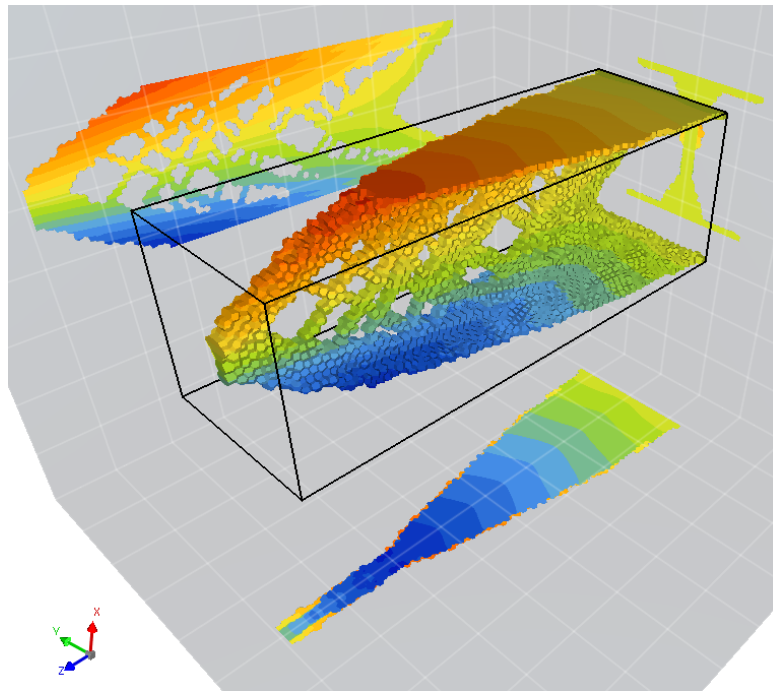


Figure 6.4: Converged topology for the beam problem: 8.5% volume fraction constraint; 20,000 polyhedrons; final compliance 423.5. Three-dimensional view, U_z -colored.

6.2.1

Embedding vs. Conventional Formulation

Let's first evaluate the embedding technique on unstructured polyhedral meshes. Consider the same cantilever beam shown in Fig.6.3. For comparison

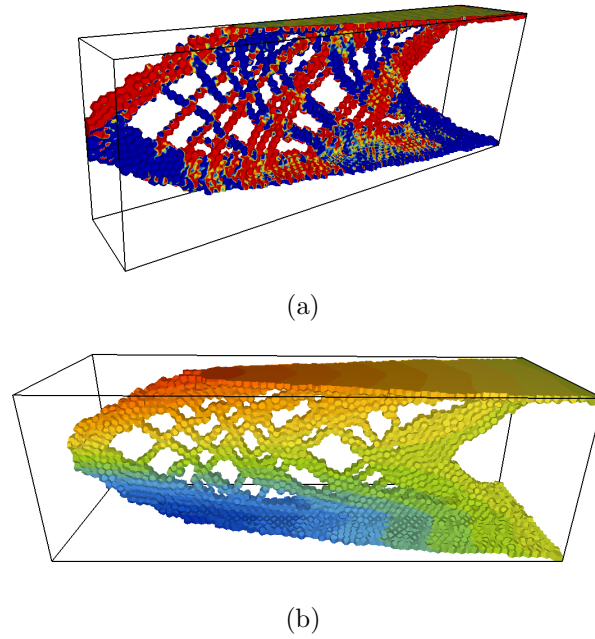


Figure 6.5: Converged topology for the beam problem: Alternatives views (a) σ_{xx} -colored non-symmetric view. Scale: red is the upper positive bound and blue is the lower negative bound. (b) Lateral view.

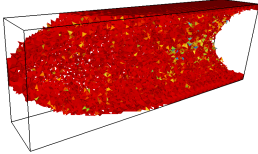
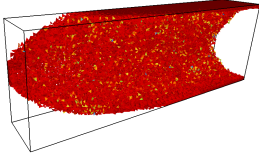
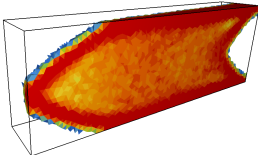
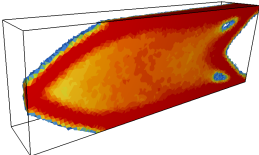
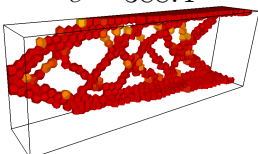
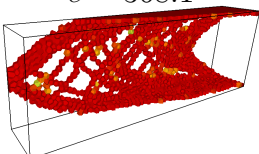
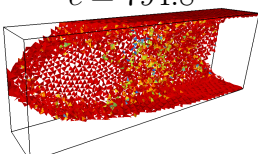
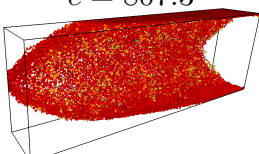
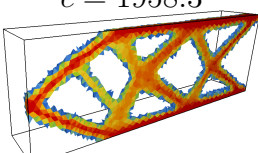
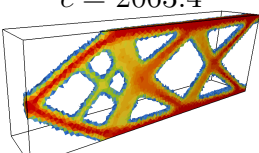
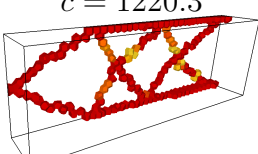
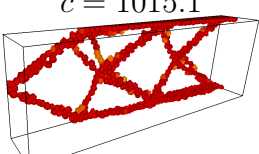

purpose, the same tetrahedral mesh used in the polyhedral embedding technique is considered here.

Three cases are compared: (1) tetrahedral mesh with standard element-base formulation, without embedding, using tetrahedral-based design variables; (2) tetrahedral element-based, without embedding, using tetrahedral-based design variables alongside a filtering scheme; and (3) polyhedral mesh embedded with tetrahedron, no filtering. These comparisons were executed using a coarse and a fine mesh, detailed in Table 6.1. Results are presented in Table 6.2.

Table 6.1: Details of the meshes used for the construction of Table 6.2. Only half of the beam was modeled due to symmetry.

Item	Mesh	
	Coarse	Fine
Nodes	52,432	159,888
DOFs	157,296	479,664
Polyhedrons (case w/ embed.)	4,000	12,000
Tetrahedrons	256,254	794,224
Edges	7,516	22,001
Faces	26,300	80,242
Tetrahedrons per polyhedron	avg. 64.0	avg. 66.2
Faces per polyhedron	avg. 12.7	avg. 13.1
Nodes per face	avg. 5.1	avg. 5.1

Table 6.2: Filtering and embedding investigation results. No symmetry shown. Mesh details in Tab. 6.1. The final compliance c is shown.

Case	Volume fraction	Topology	
		Coarse	Fine
1 – Tetrahedral-based, no embedding, no filter	10%	$c = 315.6$ 	$c = 327.0$ 
2 – Tetrahedral-based, no embedding, with filter ($r = 0.06$)	10%	$c = 532.8$ 	$c = 546.3$ 
3 – Polyhedral-based, with embedding, no filter	10%	$c = 388.4$ 	$c = 368.1$ 
1 – Tetrahedral-based, no embedding, no filter	4%	$c = 794.8$ 	$c = 807.3$ 
2 – Tetrahedral-based, no embedding, with filter ($r = 0.06$)	4%	$c = 1958.3$ 	$c = 2063.4$ 
3 – Polyhedral-based, with embedding, no filter	4%	$c = 1220.3$ 	$c = 1015.1$ 
Density color scale:			

A few aspects can be observed from Table 6.2:

- In case 1, the structure exhibits the expected checkerboard pattern, mostly comprised of one-node connections. Same behavior is observed with lower volume fraction. In these cases, tetrahedron elements were used, with no polyhedron definition whatsoever;

- Case 2 exhibits significant portion of the domain with intermediate densities values. We note that for 10% of volume fraction, a hole would appear is $\rho > 0.8$ was selected. However, for a volume fraction of 4%, structural members were clearly defined and the filter served to ensure, at some level, mesh independency. Here we can note how regularization schemes smooths the density field;
- Case 3 considered polyhedral elements using the embedding, and yielded the expected orthogonal members, with higher resolution on a finer mesh. Finer members were found as a consequence of the imposed low volume fraction and the solution converged to a 2D-like topology.

We finally note that the polyhedral mesh with the embedding technique was able to capture finer structural members, while having a solution close to a solid/void design. Lower order elements, in this case linear tetrahedrons, may result in checkerboarding, independent of the mesh used. The accurate solutions obtained in case 3 of Table 6.2 presents lower compliance value on the finer discretization.

6.2.2

Structured Meshes and Filtering

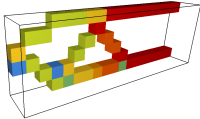
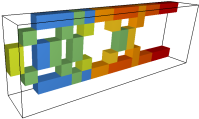
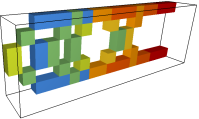
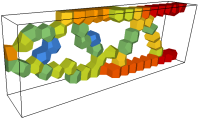
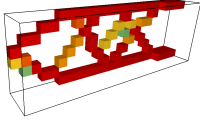
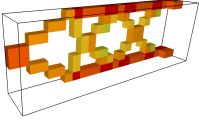
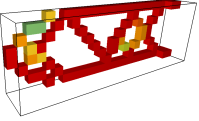
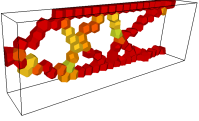
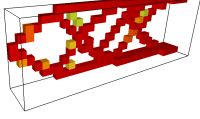
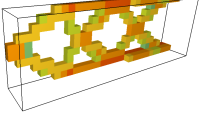
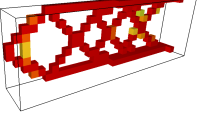
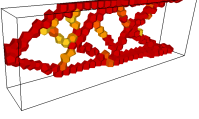
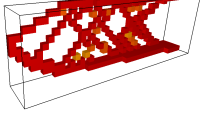
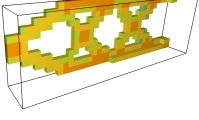
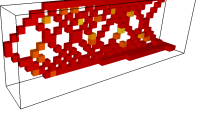
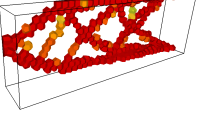
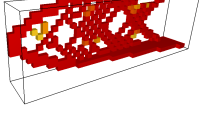
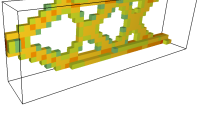
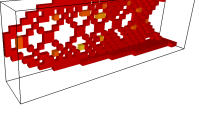
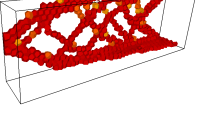
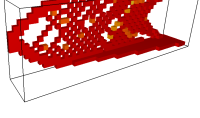
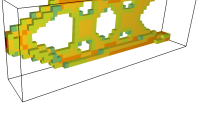
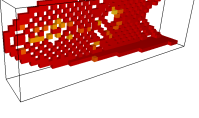
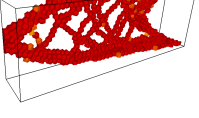
The same cantilever beam from Fig. 6.3 is investigated here. We compare results from the polyhedral embedding technique with (1) regular hexahedral elements¹ with embedding, without filter (same embedding as shown in Fig. 5.4); (2) linear brick elements with a linear density filter of radius $0.15L$ and; (3) linear brick elements without filter. Results are presented in Table 6.3.

Mesh independency was indeed obtained with application of filtering in structured brick meshes. The filter size, however, was unable to capture any of the adjacent elements and therefore, in the most coarse case, the obtained topology (and compliance) are the same as the case without filter. We can observe that one-node connections became rapidly apparent in structured meshes due to the intrinsic nature of the discretization. We note that the embedding was capable of obtaining optimal topologies, approximating analytical results as mesh sizes increases without mesh-biased solutions. No instance of checkerboard was observed and, as extensively discussed, no filter was applied.

The application of filtering, again, resulted in smoother density field, with no clear 0–1 design. We also note that, similar to the previous results, the compliance was minimum at cases without filtering. The compliance value showed a minimization pattern as the number of design variables increased, except when filter was applied. With more design variables, the material

¹Essentially a brick element, but interpreted as an arbitrary polyhedron.

Table 6.3: Comparison between brick and polytopes elements. Brick refers to the linear brick element. Symmetry not shown.

Design Vars.	Hexahedral with embedding	Brick with filter $r_{\min} = 0.15L$	Brick without filter	Polytope with embedding
330	$c = 795.1$ 	$c = 1931.9$ 	$c = 1931.9$ 	$c = 1674.6$ 
750	$c = 424.1$ 	$c = 1348.5$ 	$c = 504.4$ 	$c = 539.4$ 
1,500	$c = 367.9$ 	$c = 1667.7$ 	$c = 409.1$ 	$c = 434.1$ 
2,500	$c = 352.4$ 	$c = 1553.6$ 	$c = 389.3$ 	$c = 407.6$ 
4,000	$c = 343.8$ 	$c = 1500.8$ 	$c = 369.6$ 	$c = 388.3$ 
6,000	$c = 336.4$ 	$c = 1506.1$ 	$c = 354.1$ 	$c = 375.5$ 

distribution can be further optimized, resulting in better minimization of the objective function, namely compliance. The same behavior was not observed in filtered obtained topologies.

Figure 6.6 shows the compliance convergence history for the example corresponding to a polyhedra mesh using the embedding technique. The shape of the curves is product of the SIMP approach, using continuation. Rapid oscillation occurs when the penalty parameter is incremented. It can be observed that a 8,000, 10,000 and 12,000 polyhedra mesh gives the same compliance, $c = 368 \pm 2$. We note that every compliance minimization problem discussed in this work presents *very* similar convergence history plots and therefore will be omitted. Finally, the convergence pattern suggests that a maximum penalty of 3.5 is enough.

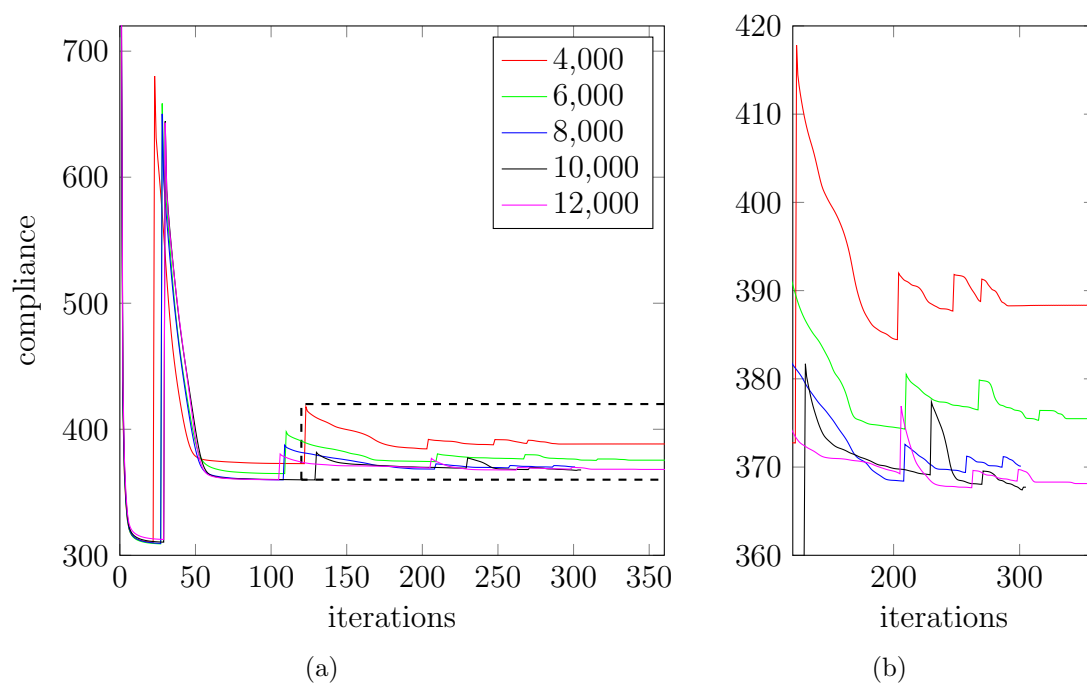


Figure 6.6: Compliance convergence history for differently-sized fixed-beam problem. Legend indicates number of design variables (polyhedrons). (b) shows a zoom from dashed box on (a).

6.3

Edge Supported Cantilever Beam

Now, a series of examples is presented to demonstrate the capabilities of the framework. We begin with the same cantilever beam from Fig. 6.3, however fixed at the two opposite vertical edges, also subject to a transverse load at the other end. Figure 6.7 shows the problem's boundary conditions and obtained topology for a 10% volume fraction. Due to symmetry, only half the domain was discretized, using 12,000 polyhedrons. The embedding is capable of resulting in topologies really close to a 0–1 design: in this example, only 0.5% of the polyhedrons have intermediate density. Since density-colored figures would be basically single-colored, we have shown the resulting topology colored with respect to the displacement along the longitudinal axis.

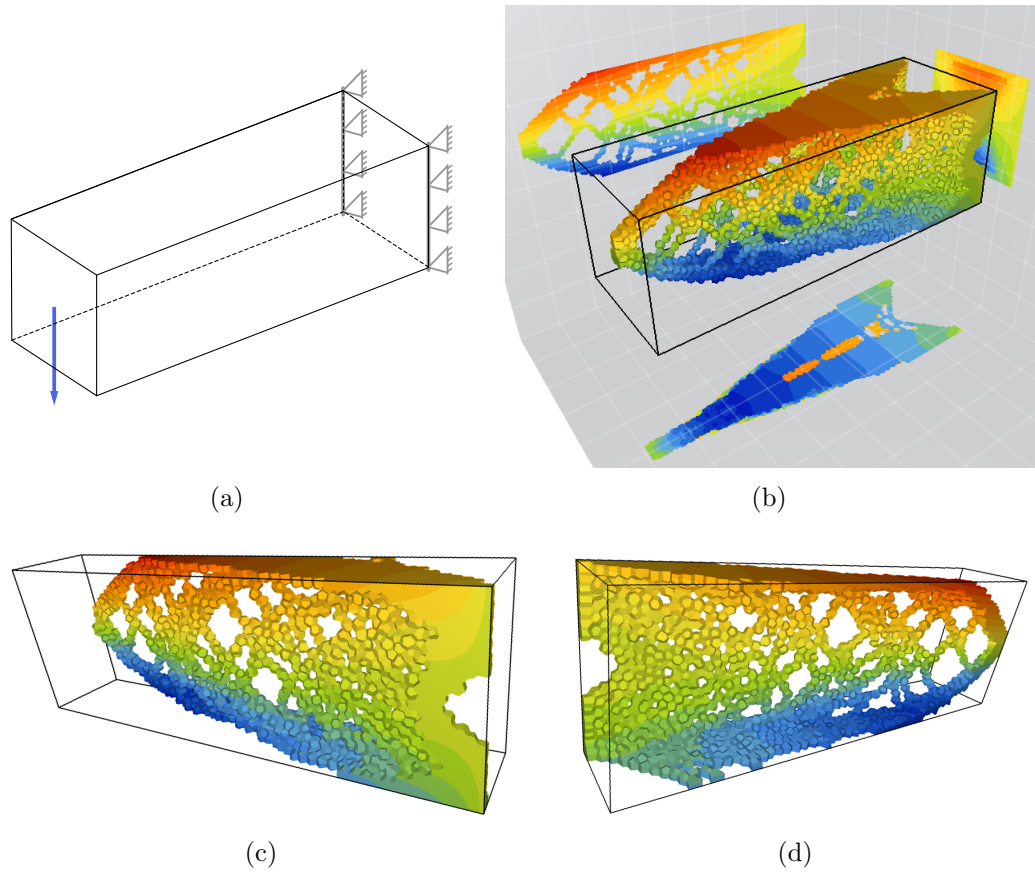


Figure 6.7: Cantilever Beam supported by two edges. Domain: $L \times L \times 3L$ discretized using 12,000 polyhedrons. (a) Supports and loads; (b) final topology; (c) and (d) detailed views of half domain

6.4

Michell-like Domain Subject to a Torsional Load

Curved domains are inherently difficult to accurately discretize using Cartesian grids. Polyhedron elements offer flexibility in discretizing such do-

mains. In this example, a Michell-like domain is extended to 3D and a torsional load is applied on the opposite face rather than a single shear load. Looking to achieve the optimal solution without prior knowledge about it, we use a box domain, except for the fixed support boundary condition, a semi-sphere.

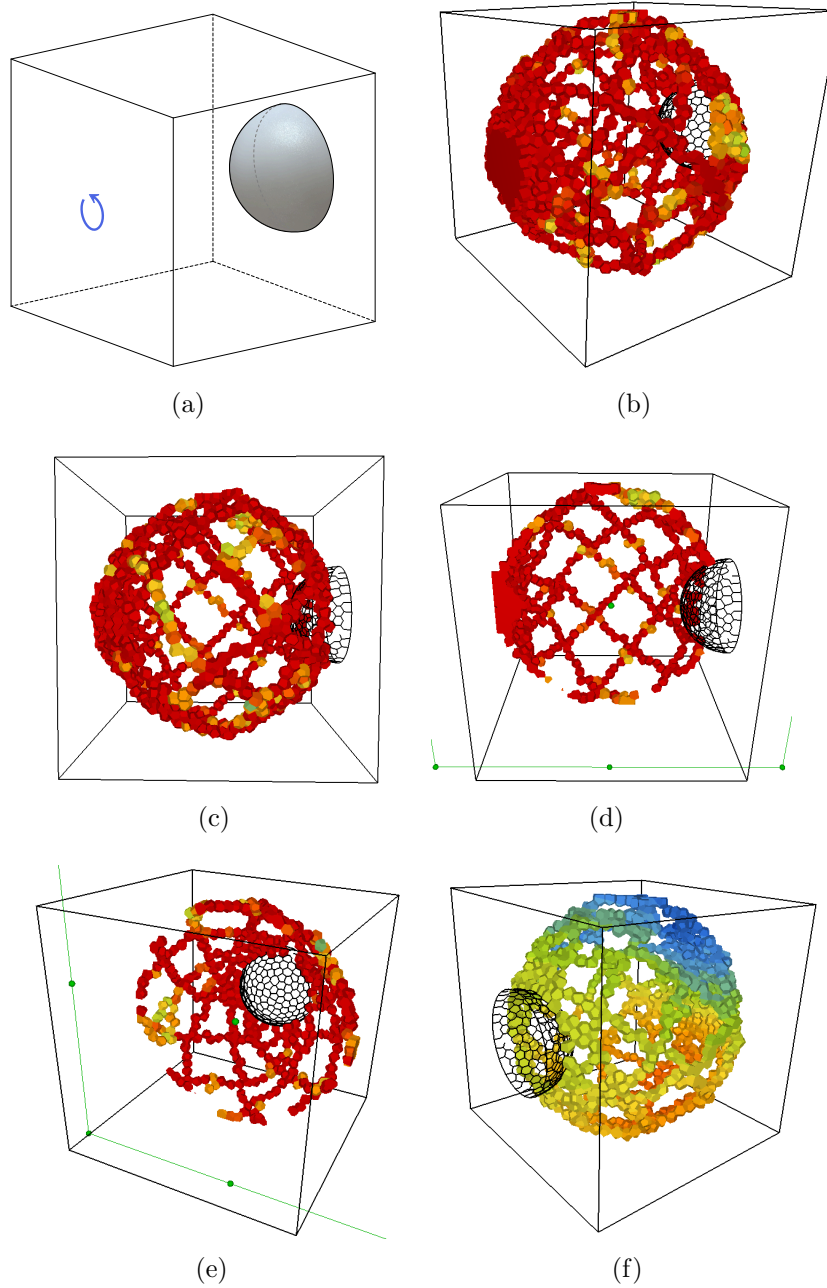


Figure 6.8: Three-dimensional Michell domain subject to a torsional load. (a) Domain and boundary condition; (b, c, d, e, f) Different views of the obtained topology, imposing a volume constraint of 4%. Plane cuts were used to detail the resulting shape.

The geometry and the optimal topologies obtained are shown in Figure 6.8. It can be seen that, as expected, the topology converged to a sphere configuration. We set the volume fraction low enough so structural members

would be visible, rather than a shell. The expected orthogonal members also appear in this problem. Domain was discretized using 20,000 polyhedral elements, yielding approximately 1,370,000 tetrahedrons and 272,000 nodes.

This example constitutes an extension of the classical Michell 2D problem (from section 6.1) into 3D. It is similar, though, to the classical Michell's sphere (or torsion ball). The Michell sphere is the optimal analytical structure loaded by a moment couple. The solution is a sphere-like structure containing orthogonal members. This Michell's solution is shown in Fig. 6.9(a) and in 6.9(b) we show its boundary conditions. We note that the final structure characteristics is dependent upon the boundary conditions, that is, the angle ϕ in which the moment is applied.

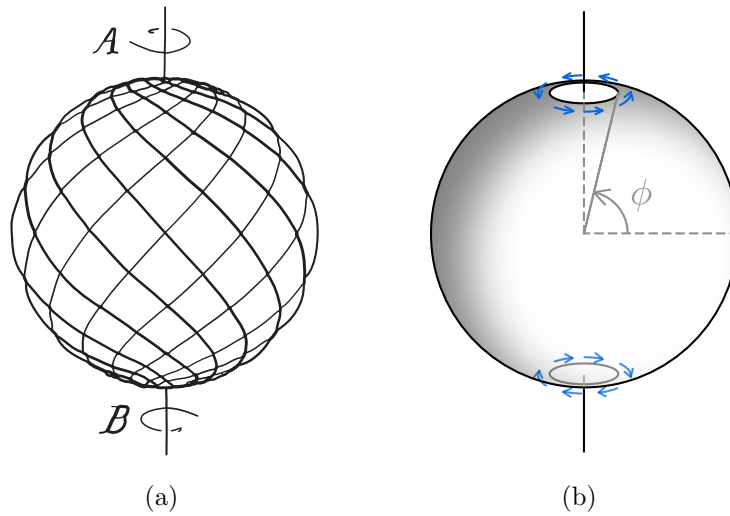


Figure 6.9: Michell original problem of the torsion ball. The features of the final structure are dependent upon the angle ϕ in which defines the small circles where the moment is applied. (a) Michell's solution [65]; (b) Boundary conditions, showing angle ϕ .

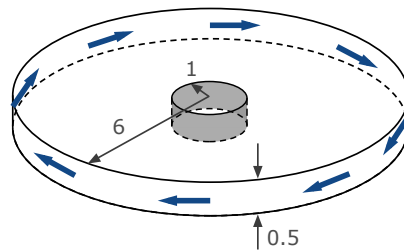
In the *modified* problem executed here (Fig. 6.8), our focus was not in the investigation of the solution behavior when changing the angle in which the force representing the moment was applied. The objective of this example was to assess if the solution were, indeed, converging to a spherical-like structure and whether orthogonal members could be observed or not. For an in-depth analysis of the influence of the angle ϕ in the solution (using the ground structure method), we refer the reader to [107]².

²We quickly point out some results: if the angle ϕ is maintained constant over mesh refinement, the solution converges. If the angle ϕ gets ever bigger (ϕ goes to $\pi/2$) as meshes gets finer, the solution will not converge (it will actually diverge).

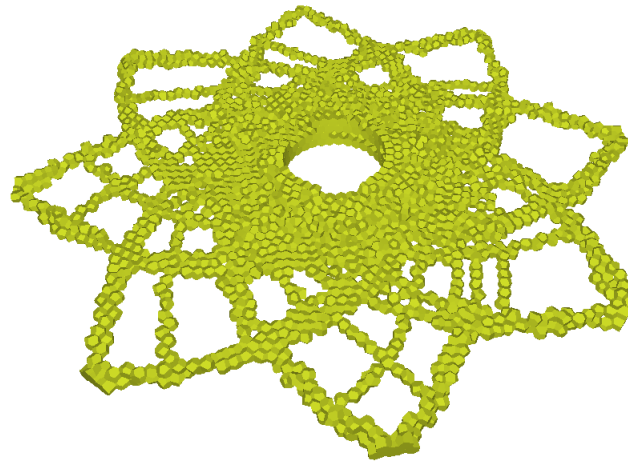
6.5

Shear Loaded Thin Disk

Now we investigate a thin disk subject to 8 shear loads [35]. The 0.5 units thick disk has an external radius of 6 units and an internal radius of 1 unit. Eight equidistant and equal in value loads are applied along the external surface and all the nodes along the hole are fixed. Figure 6.10 shows the geometry and BCs used and the obtained topology. We have used a 20,000 polyhedrons mesh and a volume constraint of 10% were imposed. Here, 0.9% of design elements contain intermediate densities.



(a)



(b)

Figure 6.10: Thin disk subject to equidistant shear loads problem. (a) Domain and boundary conditions: fixed at center with 8 equidistant shear loads. (b) Converged topology with 20,000 polyhedrons.

The results obtained are in accordance with Michell's analytical result, which for this problem is composed of 8 Michell's cantilevers (see Fig. 6.1(b)) uniformly distributed around the disk. We note that the mesh is not symmetric with respect to the 8 cantilevers.

6.6

Beam Subject to a Torsional Load

Looking to address torsional problems in three-dimensional structures, here we consider at a building-like geometry with square cross section, domain $L \times L \times 3L$. Four unitary loads were applied at the mid-point of each of the top edges. For the discretization of the domain, 20,000 polyhedral elements were used. The final volume was set to 10% of the initial volume.

Firstly, without prior knowledge about the solution, we consider a solid domain. Then, after learning that the solution lies close to the boundaries, we create a new mesh based on a hollow building. We note that the volume fraction was adjusted so the final volume of the structure remained the same. Figure 6.11 shows the solution. A thin, shell-like, closed structure was expected, given the volume and fine discretization. The average element volume was decreased 56% from the solid to the hollow domain.

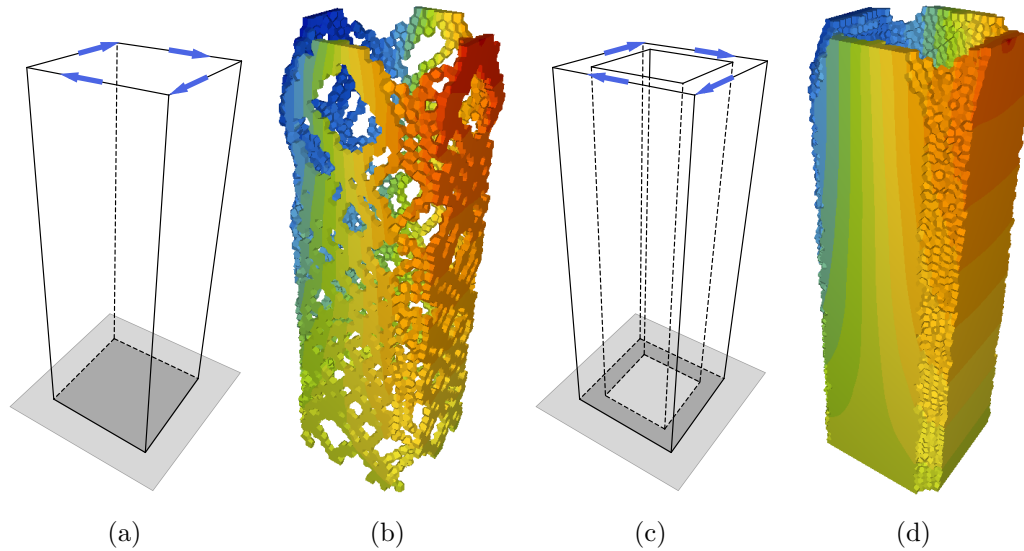


Figure 6.11: Box under torsional loads domain. (a) Geometry and BCs; (b) Result for 20,000 polyhedrons, 10% volume of solid domain; (c) New geometry and BCs, based on initial result, $0.17L$ wall thickness; (d) Finer results, using 20,000 polyhedrons and the same final volume.

In order to recover the Michell-like patterns on the hollow domain, a smaller volume fraction is imposed, as shown in Fig. 6.12(a). In this example, we have also investigated the use of a filter on the tetrahedral mesh. The filter radius was set to $r_{\min} = 0.1L$. Results are shown in Figure 6.12. It can readily be seen that this filter imposed a minimum length scale to the final topology, penalizing several thin members, thus leading to a “manufacturable” solution rather than the analytical one. However, as shown in Fig. 6.12, the final density field is no longer a true solid/void design, consisting in several elements with

intermediate densities, as opposed to the optimal solution found in polytopes with embedding (same density scale as shown in Tab. 6.2). We note that the solution is not perfectly symmetrical as no symmetry was imposed, nor the mesh was symmetric. The arbitrary shape of the polyhedrons yielded different structural member arrangement in each side of the domain.

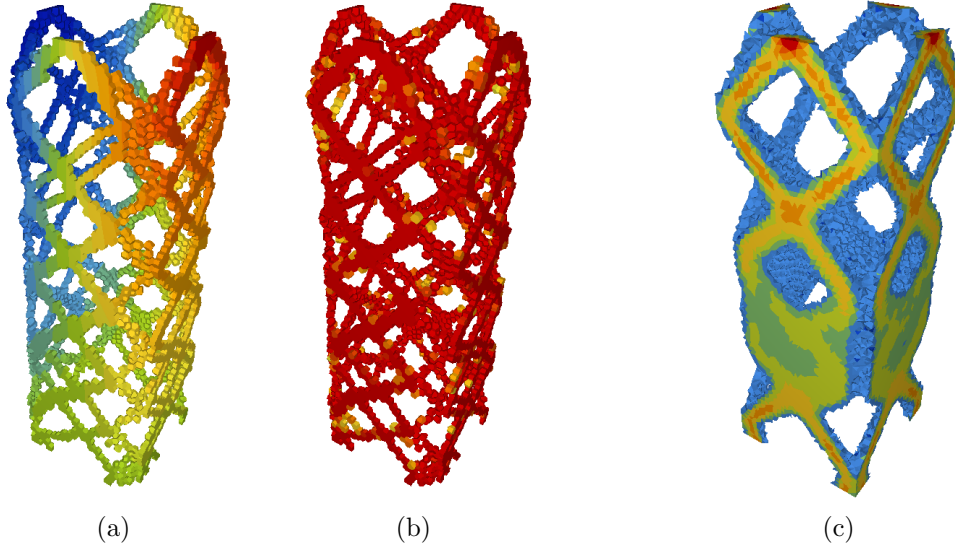


Figure 6.12: Other results for the box under torsional load. (a) 10% volume fraction on hollow domain, displacement colored. Note that the majority of the elements have $\rho = 1$; (b) same result as (a), density colored; (c) Solution of the building under torsional load using a minimum length scale filter of radius $r = 0.1L$, density colored.

6.7

Final Remarks

Topology optimization using the embedding technique on an unstructured polyhedral mesh provided results in agreement with analytical solutions found in the literature. Further, no instance of checkerboard pattern or one-node (and one-edge) connection were observed. The members of the structure intersect nearly at right angles, even for a coarse polyhedral mesh, indicating that mesh bias is alleviated with polyhedral elements.

Our obtained solutions approach analytical solutions as mesh sizes increases. Figure 6.12 briefly shows that the use of a filter would impose minimum length scale constraint thus indicating the elimination of mesh dependency. From our investigation, the obtained topologies using the embedding resulted in true solid and void design, with very few instances of intermediate densities (2% of the design elements, at most).

7

Conclusions and Extensions

In this dissertation, a topology optimization approach using polytopes is presented. We call it embedding of elements. We have investigated structural topology optimization using unstructured polyhedral meshes. A tetrahedral sub-discretization was used to perform FEA. The optimization design variables, and therefore the density (which defines the final topology), are associated with polyhedrons, thus we benefit of every aspect of using a polyhedral mesh for topology optimization, but without the complexity and issues associated with the finite element analysis on the same domain discretization.

The polytope-based discretization offers several advantages over commonly used elements. They allow the representations of complex design domains, better capture of local effects, elimination of one-node and one-edge connection in final topologies and they also avoid the checkerboard arrangement on optimal solutions. FEA is performed on tetrahedral elements, in which an efficient implementation is easy to achieve compared to more complex interpolants (harmonic shape functions, for example) [41].

Higher resolution topology, though, means that our solutions may approach theoretical/analytical solutions as mesh sizes increases. Our approach provides optimal topologies free of numerical instabilities without the use of any explicit regularization scheme. The proposed technique is promising for high fidelity topology optimization, as indicated by fine numerical results. The embedding is actually a linear mapping that is applied through a mapping matrix, which, in turn, may also be used to impose minimum length scale filters, for example. This amounts to the definition of a mapping in which can be composed by several linear filters. Further, the code provides facility in applying features such as symmetry alongside the embedding in the final topology by means of a composition of linear filters.

Through our approach of embedding of elements, analysis variables are completely separated from the optimization variables, i.e., routines are decoupled. Further, we define a *super-element* as the collection of finite element that lie in the support of a design element. Super-elements contains information from displacement and optimization routines. In our computational scheme,

three distinct fields are defined: displacement, for FEA; design variable, for optimization; and density, representing the final topology. The presence of design and density mesh indicates that a number of filters schemes can be implemented and tested without much effort. The application of filters may be beneficial in a sense that can further smooth the obtained topology or impose constraints such as minimum (or maximum) length scale. We note that filters may also be used in sensitivities. Our embedding framework is broad and expandable.

From an efficient meshing algorithm for polytope-based discretizations based on CVT meshes, we have further optimized it for the penalization of small edges. The method amounts to effectively removing small edges, leading to a better conditioned stiffness matrix. An iterative solver may not be capable of solving the finite element system of equations due to the very ill-conditioned nature of the stiffness matrix obtained from the embedding problem on a CVT-only mesh. A CVT mesh contains a number of badly shaped polyhedrons, that can lead to severely distorted tetrahedrons, sometimes making the system of equations not suitable for an iterative solver. Our mesh optimization procedure, on the other hand, was able to improve the mesh, increasing the smallest edge length about 5 orders of magnitude and the condition number associated with it in 2 orders of magnitude. After the mesh optimization, PCG solver was able to achieve the correct solution in a practical time. We also note that our embedding algorithm tetrahedralizes the polyhedrons, thus creating conforming tetrahedral elements that are suitable for posterior FEA.

We have found, though, that on structured tetrahedral meshes, the PCG solver (iterative) is faster than PARDISO (direct) for larger meshes, even when there's available system RAM. The key point in these meshes is the low condition number associated with the stiffness matrix, due to the uniformity of the tetrahedron elements, with low spread on the edge length measure across the entire mesh. On typical polyhedral meshes used here, the PCG was not able to beat PARDISO in terms of runtime. Arbitrary polyhedrons geometry naturally induces a higher standard deviation on the edge length, leading to worse condition number than on structured grids.

For large scale problems, solving the linear system with a direct solver is simply not an option. Therefore, one should rely on iterative solvers. From our experience, problems with more than 25,000 polyhedrons with embedded tetrahedrons (yielding roughly 1.5 millions tetrahedrons) becomes hard to solve on a computer with 24GB of RAM using a direct solver (e.g., PARDISO). We suggest further treatment of the meshes, specially in non-Cartesian coordinates in order to achieve faster solution on larger problems using iterative

solvers. Although we have obtained good results, we believe there's still much room for improvement on the tetrahedrons quality. Additionally, we point out that as the optimization process evolves, many elements in the domain starts taking the lower bound value of the density (or design variable, in this case). In subsequent iteration, when solving the associated linear elasticity problem, the associated matrix will have an increased condition number, thus decreasing the efficiency of the iterative solver.

The numerical results obtained here indicates no presence of checkerboarding and no one-node connections have been observed, even in coarse meshes. Further, some structures, in accordance with analytical results, follow the Michell-type patterns. For instance, the beam subject to one load and torsion problems indicates clear presence of orthogonal structural members.

We conclude by pointing out that polyhedral meshes can be applied to different topology optimization problems such as on elasticity, structural dynamics, fluid flow, among several others. Moreover, the polyhedral meshing and optimization algorithm can be useful for several areas of computational mechanics. It is able to generate non-uniform meshes with desirable gradation by selecting an appropriate density function, suitable for fracture mechanics problems, for example.

7.1

Suggestions for Future Work

With the new ideas introduced by this work, we see much room for improvement, whether by exploring in depth the new concepts presented or by incorporating new approaches to the current framework. Below we list some research directions, including further analysis that could be carried out using the present tool.

- Extension and improvement of the meshing algorithm regarding boundary representation in curved domain. Although the current algorithm has satisfactory performance, we see room for improvement in non-Cartesian domain representation. Additionally, the implementation of graded meshes can be easily achieved if such meshes are desired;
- The embedding matrix \mathbf{P} , detailed in section 3.6 can be further explored in the form it is presented in Eq. (3.8). Incorporation of new features to the allowable design space through the mapping matrix might results in an elegant and efficient solution;
- For future work, we suggest executing a real-world example. The results presented in the last chapter validated the polyhedral embedding with known optimal results. Recommendation for such problems (with known

- solution for comparison) includes high-rise building design [9] and craniofacial bones reconstruction [92];
- Implementation of finite element computations carried out in the same polyhedral discretization using the recently developed Wachspress 3D shape functions [31], and compare the results obtained. The current approach decomposes each polyhedron into approximately 60 tetrahedrons, which in turn results in expensive FE computation. Wachspress shape function could represent a significant performance gain;
 - Further, with FE computations on polyhedrons, a MTOP [67] approach could be conceived in which polyhedral elements are embedded into polyhedral elements. This approach is promising for achieving high resolution topologies, while maintaining all the features of a polyhedral mesh and even incorporating state-of-the-art displacement numerical computations through polyhedral finite elements;
 - Related to last item, a Multiresolution approach could also be implemented using the Virtual Element Method as well, instead of Wachspress coordinates. It had been already shown that VEM performed well with polyhedral topology optimization problems [35, 38];
 - Lastly, we point out that greater performance may be obtained with parallel implementation, specially with the use of clusters and current-generation GPUs.

Bibliography

- [1] **LAPACK – Linear Algebra PACKage**. <http://www.netlib.org/lapack/>. Retrieved January 2014.
- [2] **PARDISO – Parallel Direct Sparse Solver Interface**. <http://software.intel.com/sites/products/documentation/hpc/mkl/mklman/>. Retrieved January 2014.
- [3] **Pardiso solver project**. <http://www.pardiso-project.org>.
- [4] AMBROSIO, L.; BUTTAZZO, G. An optimal design problem with perimeter penalization. **Calculus of Variations and Partial Differential Equations**, v.1, n.1, p. 55–69, 1993.
- [5] AMESTOY, P. R.; OTHERS. Vectorization of a multiprocessor multifrontal code. **International Journal of High Performance Computing Applications**, v.3, n.3, p. 41–59, 1989.
- [6] BAI, Z. **Templates for the solution of algebraic eigenvalue problems: a practical guide**, volume 11. Siam, 2000.
- [7] BANGERTH, W. **MATH 676: Finite element methods in scientific computing – Lectures 34 and 35**. <http://www.math.tamu.edu/~bangerth/videos.html>, 2013. Retrieved February 2014.
- [8] BATHE, K. J. **Finite Element Procedures**. Prentice Hall, 1996.
- [9] BEGHINI, L. L.; BEGHINI, A.; KATZ, N.; BAKER, W. F. ; PAULINO, G. H. Connecting architecture and engineering through structural topology optimization. **Engineering Structures**, v.59, p. 716–726, 2014.
- [10] BEIRÃO DA VEIGA, L.; BREZZI, F.; CANGIANI, A.; MANZINI, G.; MARINI, L. D.; RUSSO, A. Basic principles of virtual element methods. **Mathematical Models and Methods in Applied Sciences**, v.23, n.01, p. 199–214, 2013.
- [11] BENDSØE, M.; SIGMUND, O. **Topology Optimization: Theory, Methods and Applications**. Engineering online library. Springer, 2003.
- [12] BENDSØE, M. P. **Optimization of structural topology, shape, and material**. Springer, 1995.

- [13] BENDSØE, M. P.; SIGMUND, O. Material interpolation schemes in topology optimization. **Archive of Applied Mechanics**, v.69, n.9-10, p. 635–654, Nov. 1999.
- [14] BISHOP, J. A displacement-based finite element formulation for general polyhedra using harmonic shape functions. **International Journal for Numerical Methods in Engineering**, v.97, n.1, p. 1–31, 2013.
- [15] BOURDIN, B. Filters in topology optimization. **International Journal for Numerical Methods in Engineering**, v.50, n.9, p. 2143–2158, 2001.
- [16] BRUNS, T. A reevaluation of the SIMP method with filtering and an alternative formulation for solid–void topology optimization. **Structural and Multidisciplinary Optimization**, v.30, n.6, p. 428–436, 2005.
- [17] BURGER, M.; HACKL, B.; RING, W. Incorporating topological derivatives into level set methods. **Journal of Computational Physics**, v.194, n.1, p. 344 – 362, 2004.
- [18] CAVAZZUTI, M.; COSTI, D.; BALDINI, A.; MORUZZI, P. **Automotive chassis topology optimization: a comparison between spider and coupé designs**. In: Proceedings of the World Congress on Engineering, volume 3, 2011.
- [19] CHRISTENSEN, P.; KLARBRING, A. **An Introduction to Structural Optimization**. Solid mechanics and its applications. Springer Netherlands, 2009.
- [20] CUTHILL, E.; MCKEE, J. **Reducing the bandwidth of sparse symmetric matrices**. In: Proceedings of the 1969 24th National Conference, ACM '69, p. 157–172, New York, NY, USA, 1969. ACM.
- [21] DATTA, B. N. **Numerical linear algebra and applications**. Siam, 2010.
- [22] DAVIS, T. A. **User's guide for the unsymmetric-pattern multifrontal package UMFPACK**, 1993.
- [23] DAVIS, T. A. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. **ACM Transactions on Mathematical Software (TOMS)**, v.30, n.2, p. 165–195, 2004.
- [24] DAVIS, T. A.; DUFF, I. S. An unsymmetric-pattern multifrontal method for sparse LU factorization. **SIAM Journal on Matrix Analysis and Applications**, v.18, n.1, p. 140–158, 1997.

- [25] DAVIS, T. A.; DUFF, I. S. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. **ACM Transactions on Mathematical Software (TOMS)**, v.25, n.1, p. 1–20, 1999.
- [26] DAVIS, T. A.; GILBERT, J. R.; LARIMORE, S. I.; NG, E. G. Algorithm 836: COLAMD, a column approximate minimum degree ordering algorithm. **ACM Transactions on Mathematical Software (TOMS)**, v.30, n.3, p. 377–380, 2004.
- [27] DEATON, J. D.; GRANDHI, R. V. A survey of structural and multidisciplinary continuum topology optimization: post 2000. **Structural and Multidisciplinary Optimization**, v.49, n.1, p. 1–38, 2014.
- [28] DÍAZ, A.; SIGMUND, O. Checkerboard patterns in layout optimization. **Structural and Multidisciplinary Optimization**, v.10, p. 40–45, 1995.
- [29] DUARTE, L.; CELES, W.; PEREIRA, A.; MENEZES, I.; PAULINO, G. **PolyTop++: An efficient alternative for serial and parallel topology optimization on CPU & GPUs**. Manuscript submitted for publication, 2014.
- [30] FLOATER, M.; GILLETTE, A.; SUKUMAR, N. Gradient bounds for wachspress coordinates on polytopes. **SIAM Journal on Numerical Analysis**, v.52, n.1, p. 515–532, 2014.
- [31] FLOATER, M.; GILLETTE, A.; SUKUMAR, N. Gradient bounds for Wachspress coordinates on polytopes. **SIAM Journal on Numerical Analysis**, v.52, n.1, p. 515–532, 2014.
- [32] FLOATER, M. S. Mean value coordinates. **Computer aided geometric design**, v.20, n.1, p. 19–27, 2003.
- [33] FLOATER, M. S.; HORMANN, K.; KÓS, G. A general construction of barycentric coordinates over convex polygons. **Advances in Computational Mathematics**, v.24, n.1-4, p. 311–331, 2006.
- [34] FLOATER, M. S.; KÓS, G.; REIMERS, M. Mean value coordinates in 3D. **Computer Aided Geometric Design**, v.22, n.7, p. 623–631, 2005.
- [35] GAIN, A. **Polytope-based topology optimization using a mimetic-inspired method**. 2014. PhD thesis, University of Illinois at Urbana-Champaign.
- [36] GAIN, A. L.; PAULINO, G. H. Phase-field based topology optimization with polygonal elements: a finite volume approach for the evolution equation. **Structural and Multidisciplinary Optimization**, v.46, n.3, p. 327–342, 2012.

- [37] GAIN, A. L.; PAULINO, G. H. A critical comparative assessment of differential equation-driven methods for structural topology optimization. **Structural and Multidisciplinary Optimization**, v.48, n.4, p. 685–710, 2013.
- [38] GAIN, A. L.; PAULINO, G. H.; DUARTE, L.; MENEZES, I. F. Topology optimization using polytopes. **arXiv preprint arXiv:1312.7016**, 2013.
- [39] GEORGE, A.; LIU, J. W. H. An implementation of a pseudoperipheral node finder. **ACM Trans. Math. Softw.**, v.5, n.3, p. 284–295, 1979.
- [40] GIBBS, N. E.; POOLE, JR, W. G.; STOCKMEYER, P. K. An algorithm for reducing the bandwidth and profile of a sparse matrix. **SIAM Journal on Numerical Analysis**, v.13, n.2, p. 236–250, 1976.
- [41] GILLETTE, A. **Geometric criteria for generalized barycentric finite elements**. SIAM–ACM Joint Conference on Geometric and Physical Modeling, Orlando, FL. Available at <http://math.arizona.edu/~agillette/research/gd11-talk.pdf>, 2011.
- [42] GOLUB, G. H.; VAN LOAN, C. F. **Matrix Computations**, volume 3. JHU Press, 2012.
- [43] GOULD, N. I.; SCOTT, J. A.; HU, Y. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. **ACM Transactions on Mathematical Software (TOMS)**, v.33, n.2, p. 10, 2007.
- [44] GROENWOLD, A. A.; ETMAN, L. On the equivalence of optimality criterion and sequential approximate optimization methods in the classical topology layout problem. **International Journal for Numerical Methods in Engineering**, v.73, n.3, p. 297–316, 2008.
- [45] GUEST, J. K.; PRÉVOST, J.; BELYTSCHKO, T. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. **International Journal for Numerical Methods in Engineering**, v.61, n.2, p. 238–254, 2004.
- [46] GUIBAS, L. J.; KNUTH, D. E.; SHARIR, M. Randomized incremental construction of delaunay and voronoi diagrams. **Algorithmica**, v.7, n.1-6, p. 381–413, 1992.
- [47] HAFTKA, R. T. Simultaneous analysis and design. **AIAA journal**, v.23, n.7, p. 1099–1103, 1985.
- [48] HUGHES, T. J. R. **The Finite Element Method: Linear Static and Dynamic Finite Element Analysis (Dover Civil and Mechanical Engineering)**. Dover Publications, 2000.

- [49] INTEL. **Intel Math Kernel Library – Reference Manual, MKL 11.1.** <http://developer.intel.com/software/products/mkl/>.
- [50] IRONS, B. M. A frontal solution program for finite element analysis. **International Journal for Numerical Methods in Engineering**, v.2, n.1, p. 5–32, 1970.
- [51] JANG, G.-W.; JEONG, J. H.; KIM, Y. Y.; SHEEN, D.; PARK, C.; KIM, M.-N. Checkerboard-free topology optimization using non-conforming finite elements. **International Journal for Numerical Methods in Engineering**, v.57, n.12, p. 1717–1735, 2003.
- [52] JANG, G.-W.; LEE, S.; KIM, Y. Y.; SHEEN, D. Topology optimization using non-conforming finite elements: three-dimensional case. **International Journal for Numerical Methods in Engineering**, v.63, n.6, p. 859–875, 2005.
- [53] JAYABAL, K.; MENZEL, A. Application of polygonal finite elements to two-dimensional mechanical and electro-mechanically coupled problems. **CMES, Comput. Model. Eng. Sci.**, v.73, n.2, p. 183–208, 2011.
- [54] KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. **SIAM Journal on Scientific Computing**, v.20, n.1, p. 359–392, 1998.
- [55] KARYPIS, G.; KUMAR, V. **MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0.** <http://www.cs.umn.edu/~metis>, 2009.
- [56] KAVEH, A. **Structural mechanics: graph and matrix methods.** Research Studies Press New York, 1995.
- [57] KHAITAN, S. K.; GUPTA, A. **High Performance Computing in Power and Energy Systems.** Springer, 2012.
- [58] KRAUS, M.; STEINMANN, P. Finite element formulations for 3D convex polyhedra in nonlinear continuum mechanics. **Computer Assisted Methods in Engineering and Science**, v.19, n.2, p. 121–134, 2012.
- [59] KROG, L.; TUCKER, A.; ROLLEMA, G. **Application Of Topology, Sizing And Shape Optimization Methods To Optimal Design Of Aircraft Components.** Airbus UK Ltd – Advanced Numerical Simulations Department, Bristol, BS99 7AR, 2011.
- [60] LAWSON, C. L.; HANSON, R. J.; KINCAID, D. R.; KROGH, F. T. Basic linear algebra subprograms for Fortran usage. **ACM Transactions on Mathematical Software (TOMS)**, v.5, n.3, p. 308–323, 1979.

- [61] LLOYD, S. Least squares quantization in PCM. **Information Theory, IEEE Transactions on**, v.28, n.2, p. 129–137, 1982.
- [62] MALSCH, E. A.; LIN, J. J.; DASGUPTA, G. Smooth two-dimensional interpolations: a recipe for all polygons. **Journal of Graphics, GPU, and Game Tools**, v.10, n.2, p. 27–39, 2005.
- [63] MANZINI, G.; RUSSO, A.; SUKUMAR, N. New perspectives on polygonal and polyhedral finite element methods. **Mathematical Models and Methods in Applied Sciences**, v.24, n.08, p. 1665–1699, 2014.
- [64] MARTIN, S.; KAUFMANN, P.; BOTSCH, M.; WICKE, M.; GROSS, M. **Polyhedral finite elements using harmonic basis functions**. In: *Computer Graphics Forum*, volume 27, p. 1521–1529. Wiley Online Library, 2008.
- [65] MICHELL, A. G. M. **The limits of economy of materials in frame structures**, 1904.
- [66] MOLER, C. **MATLAB Incorporates LAPACK**. <http://www.mathworks.com/company/newsletters/articles/matlab-incorporates-lapack.html>, 2000.
- [67] NGUYEN, T. H.; PAULINO, G. H.; SONG, J.; LE, C. H. A computational paradigm for multiresolution topology optimization (MTOP). **Structural and Multidisciplinary Optimization**, v.41, n.4, p. 525–539, 2010.
- [68] OSHER, S.; FEDKIW, R. **Level set methods and dynamic implicit surfaces**. Applied Mathematical Sciences. Springer, 2003.
- [69] PAULINO, G. H.; LE, C. H. A modified Q4/Q4 element for topology optimization. **Structural and Multidisciplinary Optimization**, v.37, n.3, p. 255–264, 2009.
- [70] PAULINO, G. H.; PEREIRA, A.; TALISCHI, C.; MENEZES, I. F.; CELES, W. **Embedding of superelements for three-dimensional topology optimization**. In: *Proceedings of Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*, 2008.
- [71] PEREIRA, A.; THEDIN, R.; CARVALHO, M.; MENEZES, I.; PAULINO, G. **Embedding of super-elements for three-dimensional topology optimization using polytopes**. EMI 2013 Engineering Mechanics Institute Conference, 2013.
- [72] PERSSON, P.-O. **Mesh generation for implicit geometries**. 2004. PhD thesis, Massachusetts Institute of Technology.

- [73] PERSSON, P.-O.; STRANG, G. A simple mesh generator in MATLAB. **SIAM review**, v.46, n.2, p. 329–345, 2004.
- [74] RIETZ, A. Sufficiency of a finite exponent in SIMP (power law) methods. **Structural and Multidisciplinary Optimization**, v.21, p. 159–163, 2001.
- [75] ROZVANY, G. Exact analytical solutions for some popular benchmark problems in topology optimization. **Structural Optimization**, v.15, n.1, p. 42–48, 1998.
- [76] ROZVANY, G. Stress ratio and compliance based methods in topology optimization– A critical review. **Structural and Multidisciplinary Optimization**, v.21, n.2, p. 109–119, 2001.
- [77] RYCROFT, C. Voro++: A three-dimensional Voronoi cell library in C++, 2009.
- [78] SAAD, Y. **Iterative methods for sparse linear systems**. Siam, 2003.
- [79] SCHENK, O.; GÄRTNER, K. Solving unsymmetric sparse systems of linear equations with PARDISO. **Future Generation Computer Systems**, v.20, n.3, p. 475–487, 2004.
- [80] SETHIAN, J. A. **Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science**, volume 3. Cambridge University Press, 1999.
- [81] SHEWCHUK, J. What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures (preprint). **University of California at Berkeley**, v.73, 2002.
- [82] SIEGER, D.; ALLIEZ, P.; BOTSCH, M. **Optimizing Voronoi diagrams for polygonal finite element computations**. In: Proceedings of the 19th International Meshing Roundtable, p. 335–350. Springer, 2010.
- [83] SIGMUND, O. Morphology-based black and white filters for topology optimization. **Structural and Multidisciplinary Optimization**, v.33, n.4-5, p. 401–424, 2007.
- [84] SIGMUND, O.; PETERSSON, J. Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. **Structural Optimization**, v.16, p. 68–75, 1998.
- [85] SLOAN, S. An algorithm for profile and wavefront reduction of sparse matrices. **International Journal for Numerical Methods in Engineering**, v.23, n.2, p. 239–251, 1986.

- [86] STOLPE, M.; SVANBERG, K. An alternative interpolation scheme for minimum compliance topology optimization. **Structural and Multidisciplinary Optimization**, v.22, n.2, p. 116–124, 2001.
- [87] STOLPE, M.; SVANBERG, K. On the trajectories of penalization methods for topology optimization. **Structural and Multidisciplinary Optimization**, v.21, n.2, p. 128–139, 2001.
- [88] SUKUMAR, N. Construction of polygonal interpolants: a maximum entropy approach. **International Journal for Numerical Methods in Engineering**, v.61, n.12, p. 2159–2181, 2004.
- [89] SUKUMAR, N.; BOLANDER, J. Voronoi-based interpolants for fracture modelling. **Tessellations in the Sciences**, 2009.
- [90] SUKUMAR, N.; MALSCH, E. A. Recent advances in the construction of polygonal finite element interpolants. **Arch. Comput. Meth. Engng.**, v.13, p. 129–163, 2006.
- [91] SUKUMAR, N.; TABARRAEI, A. Conforming polygonal finite elements. **International Journal for Numerical Methods in Engineering**, v.61, n.12, p. 2045–2066, 2004.
- [92] SUTRADHAR, A.; PAULINO, G. H.; MILLER, M. J.; NGUYEN, T. H. Topological optimization for designing patient-specific large craniofacial segmental bone replacements. **Proceedings of the National Academy of Sciences**, v.107, n.30, p. 13222–13227, 2010.
- [93] SUZUKI, K.; KIKUCHI, N. A homogenization method for shape and topology optimization. **Computer Methods in Applied Mechanics and Engineering**, v.93, n.3, p. 291–318, 1991.
- [94] SVANBERG, K. The method of moving asymptotes – a new method for structural optimization. **International Journal for Numerical Methods in Engineering**, v.24, n.2, p. 359–373, 1987.
- [95] SVANBERG, K.; SVÄRD, H. Density filters for topology optimization based on the Pythagorean means. **Structural and Multidisciplinary Optimization**, v.48, n.5, p. 859–875, 2013.
- [96] TAKEZAWA, A.; NISHIWAKI, S.; KITAMURA, M. Shape and topology optimization based on the phase field method and sensitivity analysis. **Journal of Computational Physics**, v.229, n.7, p. 2697–2718, 2010.
- [97] TALISCHI, C.; PAULINO, G. H.; LE, C. H. Honeycomb Wachspress finite elements for structural topology optimization. **Structural and Multidisciplinary Optimization**, v.37, p. 569–583, 2009.

- [98] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A.; MENEZES, I. F. M. Polygonal finite elements for topology optimization: A unifying paradigm. **International Journal for Numerical Methods in Engineering**, v.82, n.6, p. 671–698, 2010.
- [99] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A.; MENEZES, I. F. M. PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab. **Struct. Multidiscip. Optim.**, v.45, n.3, p. 309–328, 2012.
- [100] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A.; MENEZES, I. F. M. PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. **Struct. Multidiscip. Optim.**, v.45, n.3, p. 329–357, 2012.
- [101] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A.; MENEZES, I. F. M. **PolyMesher3D: a general-purpose mesh generator for polyhedral elements written in Matlab**. To be submitted, 2014.
- [102] VAN DIJK, N. P.; MAUTE, K.; LANGELAAR, M.; VAN KEULEN, F. Level-set methods for structural topology optimization: a review. **Structural and Multidisciplinary Optimization**, v.48, n.3, p. 437–472, 2013.
- [103] WACHSPRESS, E. L. **A rational finite element basis**. Academic Press New York, 1975.
- [104] WARREN, J. Barycentric coordinates for convex polytopes. **Advances in Computational Mathematics**, v.6, n.1, p. 97–108, 1996.
- [105] WICKE, M.; BOTSCH, M.; GROSS, M. **A finite element method on convex polyhedra**. In: Computer Graphics Forum, volume 26, p. 355–364. Wiley Online Library, 2007.
- [106] XIE, Y.; STEVEN, G. P. A simple evolutionary procedure for structural optimization. **Computers & structures**, v.49, n.5, p. 885–896, 1993.
- [107] ZEGARD, T. **Structural optimization: from continuum and ground structures to additive manufacturings**. 2014. PhD thesis, University of Illinois at Urbana-Champaign.
- [108] ZHOU, M.; ROZVANY, G. The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. **Computer Methods in Applied Mechanics and Engineering**, v.89, p. 309–336, 1991.
- [109] ZHOU, M.; ROZVANY, G. On the validity of ESO type methods in topology optimization. **Structural and Multidisciplinary Optimization**, v.21, n.1, p. 80–83, 2001.