UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Study of optimal design of 3D mechanical metamaterials

BACHELOR FINAL THESIS

Document:

Final Report

Author:

Ariadna Sorribas Bono

Director:

Alex Ferrer Ferre

Degree:

Aerospace Technology Engineering

Examination session:

Spring, 2023

Abstract

This thesis aims to develop and extend numerical methods for solving the elastic problem in 2D to a 3D solution. The primary objective is to investigate and identify the modifications and extensions necessary to adapt existing techniques used in 2D to a 3D framework. The focus is on developing a robust and efficient numerical method that accurately models the behavior of elastic materials in three dimensions.

The study encompasses several objectives to achieve a comprehensive understanding of 3D material design and topology optimization in the context of mechanical metamaterials. Firstly, an introduction to topology optimization is provided, including the formulation and regularization of the problem, and an explanation of density-based and the Level Set method. The thesis further investigates the SWAN repository's code, an object-oriented MATLAB software, assessing its applicability for conducting simulations.

The thesis explores the optimization of both normal materials and metamaterials microstructures. For normal materials, a comparison is made between different optimization approaches, specifically the MMA optimizer utilizing a density-based method and the Null Space optimizer employing a level set-based method. Additionally, the impact of varying final volume fractions on the optimization outcomes is studied. This investigation provides valuable insights into the influence of different parameter variations on the resulting microstructures and optimization performance.

Furthermore, the study focuses on metamaterials microstructures and explores their optimization using the Null Space optimizer, and different α and β values are employed to examine their effects on the final design. The optimization process is also conducted for different final volume fractions to evaluate the influence of volume fraction on metamaterial performance.

This study on material design and topology optimization has yielded several important conclusions: the simulations showcased the relationship between dimensionality and convergence speed, with 2D simulations demonstrating faster convergence compared to 3D simulations; analysis of parameters such as the cost function and the number of iterations has been conducted comparing different optimizers, it has been highlighted the challenges and unique considerations involved in optimizing metamaterials; and, overall, the research has contributed to the understanding of optimization processes and the generation of innovative material configurations.

Resumen

El objetivo de esta tesis es desarrollar y ampliar métodos numéricos para resolver el problema elástico en 2D y adaptarlos a una solución en 3D. Se busca investigar e identificar las modificaciones y extensiones necesarias para adaptar las técnicas existentes utilizadas en 2D a un marco en 3D. El enfoque principal se centra en desarrollar un método numérico robusto y eficiente que modele con precisión el comportamiento de los materiales elásticos en tres dimensiones.

El estudio abarca varios objetivos para lograr una comprensión integral del diseño de materiales y la optimización de la topología en 3D, especialmente en el contexto de los metamateriales mecánicos. En primer lugar, se proporciona una introducción a la optimización de topología, incluyendo la formulación y regularización del problema, así como una explicación del método basado en densidad y el método de Level Set. Además, se realiza un análisis del código del repositorio SWAN, un software orientado a objetos en MATLAB, evaluando su aplicabilidad para realizar simulaciones.

La tesis explora la optimización tanto de materiales normales como de microestructuras de metamateriales. Para los materiales normales, se realiza una comparación entre diferentes enfoques de optimización, en particular el optimizador MMA que utiliza un método basado en densidad y el optimizador Null Space que emplea un método basado en la función level set. Asimismo, se estudia el impacto de variar las fracciones de volumen final en los resultados de la optimización. Esta investigación proporciona valiosos conocimientos sobre la influencia de las variaciones de distintos parámetros en las microestructuras resultantes y el rendimiento de la optimización.

Además, el estudio se enfoca en las microestructuras de metamateriales y explora su optimización utilizando el optimizador Null Space, empleando diferentes valores de α y β para examinar sus efectos en el diseño final. También se lleva a cabo el proceso de optimización para diferentes fracciones de volumen final con el fin de evaluar la influencia de la fracción de volumen en el rendimiento de los metamateriales.

Este estudio sobre diseño de materiales y optimización de topología ha generado varias conclusiones importantes: las simulaciones han demostrado la relación entre la dimensionalidad y la velocidad de convergencia, siendo las simulaciones en 2D más rápidas en converger en comparación con las simulaciones en 3D; se ha llevado a cabo un análisis de parámetros como la función de costo y el número de iteraciones, comparando diferentes optimizadores; se han destacado los desafíos y consideraciones únicas involucrados en la optimización de metamateriales; y, en general, la investigación ha contribuido a la comprensión de los procesos de optimización y a la generación de configuraciones de materiales innovadoras.

Acknowledgements

I would like to express my absolute gratitude to my tutor, Alex Ferrer, whose dedication and passion for material design have been invaluable throughout my thesis journey. Alex's guidance, expertise and mentorship have shaped my understanding of the subject and inspired me to push my boundaries.

I would also like to extend my deepest appreciation to the other department workers, particularly Ton and Jose. Their support, commitment and contributions have been invaluable throughout this demanding process, particularly during the challenging moments.

Equally important, I also want to send my most lovely thanks to my family, friends and partner for their constant support during these months. Their encouragement and understanding have meant the world to me, and I couldn't have done it without them.

Lastly, all my acknowledgement to all the people who have played a part in my thesis journey, whether directly or indirectly. Their contributions, advice and feedback have shaped my work and made it possible for me to reach this point.

Contents

List of figures

List of tables

List of codes

Acronyms

AM	Additive Manufacturing
FEM	Finite Element Method
LHS	Left Hand Side
MMA	Method of Moving Asymptotes
MS	Master-Slave
OOP	Object-Oriented Programming
RAMP	Rational Approximation of Material Properties
RHS	Right Hand Side
RVE	Representative Volume Element
SLERP	Spherical Linear Interpolation
SLP	Sequential Linear Programming
SIMP	Solid Isotropic Material with Penalization
TO	Topology Optimization
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Objective

The main objective of this final thesis is to develop and extend the current numerical methods for solving the elastic problem in 2D to a 3D solution. This will be achieved by investigating and identifying modifications and extensions required to adapt the existing techniques used in 2D to a 3D framework. The focus will be on developing a robust and efficient numerical method that accurately models the behavior of elastic materials in three dimensions.

In addition to developing the 3D elastic numerical method, this final thesis aims to apply topology optimization techniques to design 3D metamaterials. An introduction to topology optimization will be provided, including the formulation and regularization of the problem, density-based methods, level set method, and a comparison between different optimizers. Also, a study of the TO code on the Swan repository will be conducted and an analysis and UML of the code will be performed.

The final objective of this thesis is to investigate the elastic inverse homogenization problem and study it by applying density-based and level set methods. To better comprehend the topic, simulations of the J_1, J_2 and J_3 cases will be computed. Additionally, 3D material design will be studied considering density-based and level set methods.

Overall, this final thesis aims to extend the current numerical methods for solving the elastic problem in 2D to a 3D solution, apply topology optimization techniques to design 3D metamaterials and investigate the elastic inverse homogenization problem. By accomplishing these objectives, this thesis will contribute to a better understanding of the behavior of elastic materials in three dimensions and may have implications for the design of novel materials with unique properties.

1.2 Scope

Throughout this investigation, the tasks that will be addressed are:

- **Introduction to Swan.** In order to be able to work in the Swan software, first it will be necessary to learn to program following an object-oriented approach (classes, types of data access, inheritance...) and applying clean code practices. To do it, a previous FEM code from the *Aerospace Structures* subject will be refactored.

- **Finite Element Method.** To understand the FEM code on the Swan repository, an analyisis and UML of it will be done studying it at macro and micro levels. Additionally, the expected results will be visualized with GID and compared to the expected ones.
- **Topology Optimization.** An introduction to Topology Optimization will be done to comprehend the TO code on the Swan repository; that includes: formulation and regularization of the problem, density-based methods, level set method... Also, an analysis and UML of the code will be done studying it at macro and micro levels. Lastly, a comparison between different optimizers will be realized.
- **Material Design.** Simulations of the J_1, J_2 and J_3 cases will be computed for a better comprehension of the topic. In addition, the 3D material design will be studied considering density-based and level set methods.
- **Inverse Homogenization.** An investigation of the elastic inverse homogenization problem will be realized, and a study of it will be done applying density-based and level set methods.

1.3 Requirements

In order to successfully complete this study, the following conditions must be met:

- The programming language used for the solution must be **MATLAB**
- To ensure readability and ease of interpretation by outside parties, the implemented code must adhere to **clean code practices**.
- The underlying **object-oriented approach** of the architecture must be respected in the implementation.
- The **previous architecture** must remain fully functional during the development of the new architecture
- The accuracy of the solution should be **validated** using analytical or experimental data, if available.
- All code modifications should be **documented** thoroughly, including any changes to the underlying data structures, algorithms, or numerical methods used in the implementation.
- The code should be **version-controlled** using Git to track changes and enable collaboration among team members.

The list of the tasks and its necessary calendarization (presented in a Gantt diagram) is provided in the Appendix A.

1.4 Justification

Topology Optimization is an emerging area of study whose purpose is to automatize the designing processes of any structural field. It investigates the process of defining the form, connectivity and position of voids in the interior of a given design domain. This enables a higher design liberty other than shape and dimension optimization, as it also considers thickness and cross-sectional regions of structural members (dimension) and geometrical characteristics (shape), having established in advance which configurational specifications are going to be optamized.

Still, new advances on this field allow the implementation of it in late developed design stages, enveloping all the design phases from conditions delineation to the fabricating point. The latest extensive studies and investigation concerting this topic include works from Bendsøe and Sigmund [1], Rozvany [2] and Liu and Ma [3].

Moreover, this area has experienced a period of fast expansion, specially in the investigation field and the manufacturing application (it has been the most active investigation field in the interdisciplinary optimization for the last 20 years). This expansion is the result of the development of the traditional topology optimization methods and the persisting evolution of innovative new ones.

On the other hand, "metamaterial" is the designation that any material receives when it has been engineered to possess a property which isn't encountered in naturally happening materials. Examples of them can be materials that present a negative Poisson coefficient, negative effective mass density or negative volume contractibility.

These last two decades, metamaterials have received a lot of support through different types of research: from the acoustic ones [4] to the elastic ones [5], going through other investigation fields such as metaoptics [6].

Metamaterials have been such a scientific discovery that even the distinguished journal Nature published a revolutionary work exhibiting a new mechanical metamaterial (negative Poisson coefficient). It stated that it was one of the strongest, lightest and hardest existing material. [7]

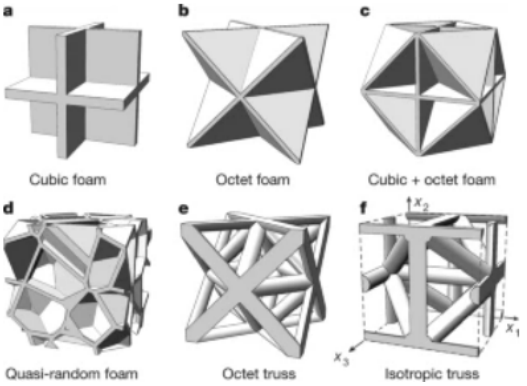


Figure 1.1: New mechanical metamaterials presented in Nature. Extracted from [7].

Those properties of metamaterials also make them of great interest for the aerospacial and aeronautical sector. For example, mechanical metamaterials have ideal characteristics for spacial actuators, and their ability to form lightweight structures while maintaining strength makes them a desirable material.

Due to the potential metamaterials offer in various applications, there is an increasing interest in creating efficient methods for their design. Topology Optimization is a promising tool for achieving this goal by optimizing the distribution of materials and voids to create metamaterials with tailored properties and improved performance.

However, the application of Topology Optimization is still a relatively new and unexplored area of research. Although some studies have investigated the use of Topology Optimization for 2D metamaterial design [8], there is still much to be done to extend these methods to 3D metamaterials. This represents a significant opportunity for research and innovation.

Combining topology optimization with metamaterials can lead to the creation of materials with unprecedented properties that traditional design methods cannot achieve, which could offer new possibilities in diverse applications ranging from aerospace to biomedical engineering.

Overall, the application of Topology Optimization to 3D metamaterial design is a promising area of research that can lead to the creation of novel materials with tailored properties and improved performance. By exploring this area, this study can contribute to the advancement of both the fields of Topology Optimization and metamaterials, as well as their applications in various domains.

Chapter 2

State of the Art

Topology optimization and material design have been actively researched areas in recent years, with a focus on developing new and innovative 3D mechanical metamaterials. In this chapter, the state of the art techniques and approaches in these fields will be reviewed, discussing their strengths and limitations.

2.1 Topology Optimization

Topology optimization is a computational design technique that seeks to optimize the layout and connectivity of a material within a design domain, subject to specified boundary conditions and loading scenarios, in order to achieve desired performance objectives (such as minimizing weight or maximizing stiffness) [9]. This method has been applied to different engineering fields, including structural mechanics, fluid dynamics and heat transfer.

This technique emerged from the field of structural optimization in the late 1980s. One of the pioneering works in this field was the seminal paper by Bendsøe and Kikuchi [10], who proposed a material distribution approach to optimal shape design. In their work, they introduced the concept of a density field to represent the material distribution within a design domain and formulated the optimization problem as a continuous problem involving the density variables. This laid the foundation for the development of various topology optimization methods in the following years.

Figure 2.1: Topology Optimization process. Extracted from [11].

One of the most popular material interpolation method used in topology optimization is the Solid Isotropic Material with Penalization (SIMP) method. In the SIMP approach, the material properties within the design domain are interpolated between the properties of the base material and those of a void (or a very weak material) using a density variable. The interpolation is accompanied by a penalization factor that discourages intermediate density values, leading to a

more distinct material distribution. The optimization problem is typically solved using gradient-based optimization algorithms, such as the Method of Moving Asymptotes (MMA) or Sequential Linear Programming (SLP).

Level set methods, initially introduced in the context of computational fluid dynamics, have been adapted for topology optimization as well [12]. In level set-based topology optimization, the material boundary is implicitly represented using a scalar level set function. The optimization is performed by updating the level set function to iteratively evolve the material boundary, ultimately achieving an optimal material layout. Level set methods offer several advantages over density-based approaches, such as the ability to handle topological changes naturally and the potential for more accurate representation of boundaries. However, they often require more complex numerical implementations and can be computationally expensive.

While topology optimization has been successfully applied to the design of mechanical metamaterials, there are still several challenges and open questions in this field. One of the main challenges is the handling of large-scale and high-resolution design problems, which often require significant computational resources and advanced optimization algorithms. Additionally, the integration of manufacturing constraints, such as those arising from additive manufacturing have to be taken into account.

2.2 Mechanical Metamaterials

Mechanical metamaterials are artificial materials with unique mechanical properties, often derived from their underlying structure rather than their constituent materials. These properties include negative Poisson's ratio, negative thermal expansion and programmable mechanical responses. Several approaches have been proposed to optimize the topology of lattice structures, truss networks and continuum materials to achieve desired performance characteristics [13]. Additionally, topology optimization has been combined with other design paradigms, such as origami and kirigami, to create mechanical metamaterials with programmable deformation and shape-memory behavior.

Figure 2.2: Graphic example of non-auxetic Mechanical Metamaterials. Extracted from [14].

One of the most common approaches to designing mechanical metamaterials is the use of lattice structures or truss networks, which consist of periodic arrangements of interconnected beams or struts [15]. By tuning the geometry and connectivity of these elements, it is possible to control the material's effective mechanical properties, such as stiffness, strength and Poisson's ratio. For example, hierarchical lattice structures inspired by natural materials, such as trabecular bone or plant tissue, have been shown to exhibit exceptional combinations of stiffness and strength with low material density [16].

On the other hand, origami and kirigami are the arts of paper folding and cutting, respectively, which have inspired the design of mechanical metamaterials with unique deformation and shape-memory behavior [17, 18]. Origami-based metamaterials often utilize fold patterns to create structures with programmable bending and twisting motions, while kirigami-based metamaterials leverage cut patterns to achieve tunable stretching and shearing behavior. These materials have found applications in soft robotics, deployable structures and impact absorption systems.

Although mechanical metamaterials offer a wide range of unique mechanical properties and applications, there are several challenges and open questions in this field. The manufacturing of these materials on a large scale, especially for microscale and nanoscale structures, is one of the key problems. Additive manufacturing and microfabrication techniques have been employed to fabricate mechanical metamaterials with complex geometries, but further advancements are needed to improve the resolution and material compatibility of these methods. Additionally, the development of efficient computational design tools, such as topology optimization methods, is critical for the optimal design of mechanical metamaterials with tailored properties and performance.

2.3 Advanced Manufacturing Techniques

The fabrication of mechanical metamaterials with complex geometries and fine-scale features relies on advanced manufacturing techniques, such as additive manufacturing (AM) and microfabrication.

Additive manufacturing, commonly known as 3D printing, is a family of manufacturing processes that build objects layer-by-layer from digital models [19]. Various AM techniques, such as stereolithography (SLA), selective laser sintering (SLS) and fused deposition modeling (FDM), have been used to fabricate mechanical metamaterials with complex geometries and tailored mechanical properties. The use of AM enables the rapid prototyping and testing of new metamaterial designs, as well as the fabrication of customized materials for specific applications.

Microfabrication techniques, such as photolithography and electron-beam lithography, are used to create structures with features on the micrometer scale [20]. These techniques have been employed to fabricate mechanical metamaterials with fine-scale features, enabling the investigation of size effects and the realization of unique mechanical properties at small length scales.

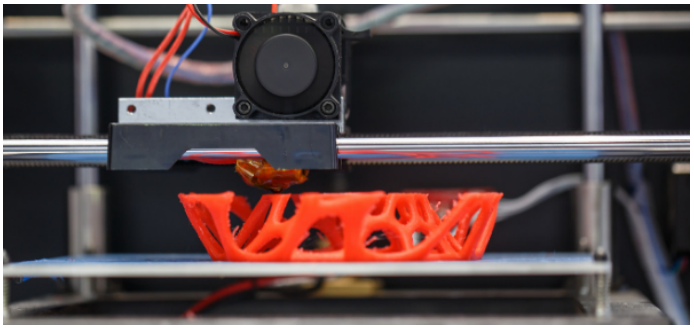


Figure 2.3: Additive Manufacturing. Extracted from [21].

2.4 Emerging Applications

Mechanical metamaterials have potential applications across various fields, from aerospace and automotive industries to biomedical and wearable devices.

- **Impact Absorption and Energy Dissipation.** Mechanical metamaterials with negative Poisson's ratios or auxetic behavior have been shown to exhibit excellent impact absorption and energy dissipation properties [16]. These materials can be used in the design of protective gear, such as helmets and armor, as well as in the development of energy-absorbing components for vehicles and infrastructure.
- **Soft Robotics.** Mechanical metamaterials with programmable mechanical responses have been used in the development of soft robotic systems [22]. These materials enable the design of actuators, sensors and grippers that can adapt to various shapes and environments. Origami and kirigami-based metamaterials have been particularly useful in this context, providing lightweight and flexible structures with tunable stiffness and shape memory properties.
- **Biomedical Devices.** Mechanical metamaterials have also found applications in the design of biomedical devices, such as prosthetics, orthotics and implants [23]. The ability to tailor the mechanical properties of these materials allows for the development of devices that closely mimic the performance of natural tissues and structures, improving patient outcomes and comfort. Additionally, the use of additive manufacturing techniques enables the fabrication of patient-specific devices that conform to individual anatomies.
- **Acoustic and Vibration Control.** Acoustic metamaterials are a class of mechanical metamaterials that exhibit unique wave propagation properties, such as negative effective mass or stiffness [24]. These materials have been used in the design of noise barriers, acoustic cloaks and vibration dampers, providing effective control of sound and vibration in various applications. Lattice structures and resonant elements, such as Helmholtz resonators, have been employed in the design of acoustic metamaterials to achieve desired performance characteristics.

Chapter 3

Elasticity Theory

Topology optimization is a powerful design technique that involves the use of algorithmic models to optimize the layout of materials within a defined space, with the aim of achieving the best possible shape. This process takes into consideration specific loads, conditions and constraints that are pre-defined by the user. Its implementation process relies on the fundamental theory behind the linear elasticity problem, which operates under the assumptions of small deformations, linear materials and the disregard of any possible dynamic effects.

Elasticity theory is a branch of continuum mechanics that deals with the deformation and stresses in solid materials subjected to external forces or changes in temperature. It is based on the fundamental principles of equilibrium, compatibility and constitutive relationships, which describe the balance of forces, the kinematics of deformation and the material's mechanical response, respectively. By applying elasticity theory, engineers and designers can predict how a structure will deform and distribute stresses under various loading conditions.

In this chapter, the key components of elasticity theory will be explored. These components include equilibrium equations (which govern the balance of forces), constitutive equations (which describe the relationship between stress and strain) and strain-displacement relations (which link the displacements of material points to the strain tensor components). Additionally, the boundary conditions will be stated and different practical examples will be discussed to illustrate the application of elasticity theory in the context of structural design and analysis. Understanding these concepts will provide a strong foundation for the subsequent sections on topology optimization and its applications.

3.1 Equilibrium Equations

Equilibrium equations describe the balance of forces and moments in a continuous medium. In elasticity theory, these equations are derived from Newton's second law of motion and the principles of linear and angular momentum conservation. For a deformable solid body, the equilibrium equations are given by:

$$\frac{\partial \sigma_{ij}}{\partial x_j} + f_i = 0 \tag{3.1}$$

where σ_{ij} represents the stress tensor components, x_j the spatial coordinates and f_i the body force vector per unit volume acting on the material.

3.2 Constitutive Equations

Constitutive equations describe the relationship between the stress and strain in a material, capturing its mechanical behavior under deformation. These equations provide a means to determine the internal stresses in a solid body as a function of the strains it experiences, thereby characterizing the material properties.

In linear elasticity, the constitutive relationship is given by Hooke's Law, which states that the stress tensor σ is linearly related to the strain tensor ε:

$$\sigma_{ij} = C_{ijkl} \cdot \varepsilon_{kl} \tag{3.2}$$

where C_{ijkl} are the components of the fourth-order elastic stiffness tensor, which depends on the material's properties, such as its elastic modulus (E) and Poisson's ratio (ν). For a 2D homogeneous linear elastic material, the elastic constants matrix C can be expressed as:

$$C = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{3.3}$$

3.3 Strain-Displacement Relations

Strain-displacement relations establish a connection between the displacement field and the strain experienced by a deformable body. These relations are essential for solving elasticity problems, as they enable the calculation of strains from known displacement fields or the determination of displacements from known strains.

In the context of linear elasticity, the strain tensor ε is related to the displacement field u through the symmetric gradient. The strain-displacement relations can be expressed as:

$$\varepsilon_{ij} = \frac{1}{2} \cdot \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{3.4}$$

where ε_{ij} are the components of the strain tensor, u_i the components of the displacement field and x_j the spatial coordinates.

The strain-displacement relations, together with the constitutive equations and equilibrium equations, form a complete system of equations that describe the mechanical behavior of elastic materials. By solving this system of equations with appropriate boundary conditions; the stress, strain and displacement fields of a deformable body subjected to external loads can be determined. These relations are fundamental to the analysis of elastic materials and are widely used in various engineering applications, including finite element analysis and topology optimization, to predict the behavior of structures under mechanical loads.

3.4 Boundary Conditions

In order to solve any problem related to the elasticity theory, it is essential to clearly define the boundary conditions that govern the behavior of the structure. These apply to the entire structure or to a specific part of it, and they describe the interaction between a structure and its environment, which can result from the application of external forces or restraints imposing displacements.

On a domain Ω, the boundary Γ can have two different types of constraints: displacement (Γ_u) or force (Γ_t).

- **Dirichlet boundary conditions**, which prescribe the values on the boundary Γ_u of the displacement field such that $u = \hat{u}$. This means that the displacement is fixed at specific points on the boundary, and can be used to model scenarios such as clamping or fixed supports.
- **Neumann boundary conditions**, which prescribe the values on the boundary Γ_t of the force field such that $\sigma_n = \hat{t}$. This means that the forces acting on the boundary are known, and can be used to model scenarios such as applied loads or pressure.

It is important to note that there are numerous other macroscopic boundary conditions that are not discussed, as this section specifically addresses the ones that have been applied to the studied case.

3.4.1 2D case

The macroscopic boundary conditions considered in the 2D cases of this type of problem are the Dirichlet ones.

These conditions require specifying the nodes to be constrained, their degrees of freedom and its prescribed value. For example, on the case of a squared mesh, the 4 corner nodes are the ones constrained and their displacements in both directions are set to 0 (a visual representation of this restrain is shown in Figure 3.1). This process is done by the GID software during the mesh generation process.

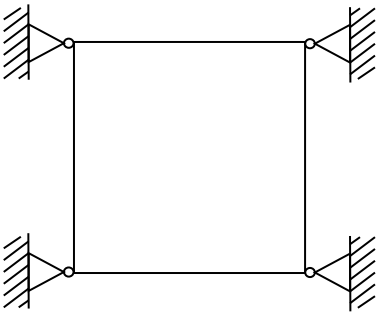


Figure 3.1: 2D square mesh with Dirichlet conditions (prescribed displacement = 0) imposed on the corner nodes.

3.4.2 3D case

Similar to the 2D case, the 3D analysis considers Dirichlet conditions.

They follow the same structure as in 2D: a matrix containing the nodes that are being constrained, their degrees of freedom and its prescribed value. Concretely, this study focuses on a cubic mesh with the corner nodes being fixed and their displacement set to 0 (see Figure 3.2).

In this case, for convenience, a custom MATLAB code has been developed to compute it based on the cube's geometry (see Section 7.3.2).

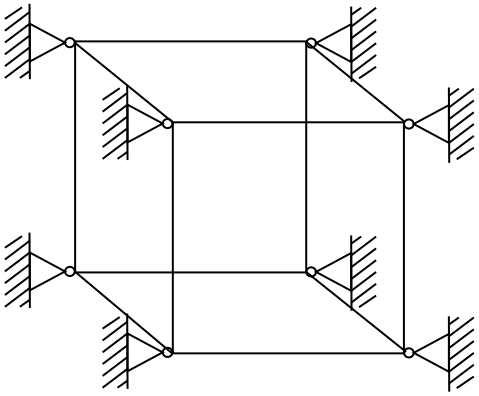


Figure 3.2: 3D cubic mesh with Dirichlet conditions (prescribed displacement = 0) imposed on the corner nodes.

3.5 Examples

In order to demonstrate the application of elasticity theory in practical scenarios, the typical cantilever problem will be analyzed. This problem involves a cantilever beam subjected to fixed displacements on the left end and a vertical load applied to the right extreme. This example will serve as a representative case to showcase the utilization of elasticity theory in analyzing the mechanical behavior of structures.

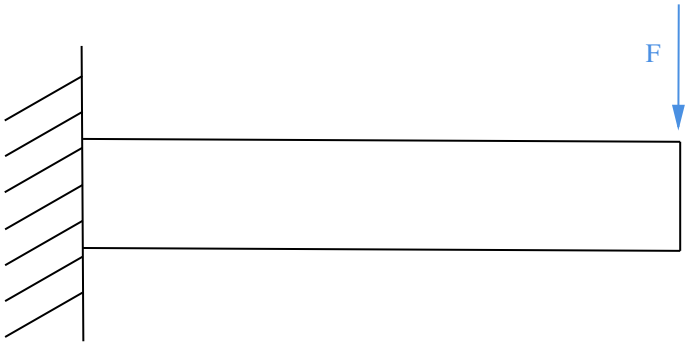
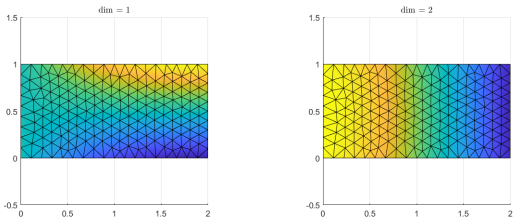
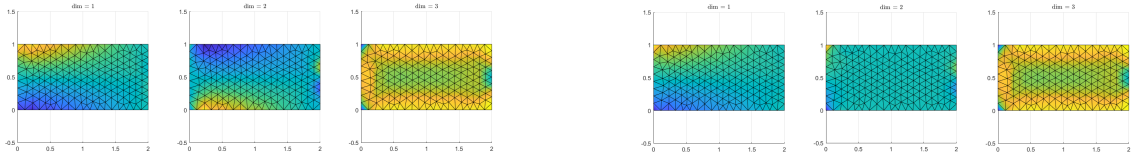


Figure 3.3: Cantilever problem diagram

Figure 3.4 displays the resulting displacements, strain and stress obtained upon solving the elastic problem.



(a) Displacement Distribution in the x and y directions, respectively.



(b) Strain Distribution in the x, y and xy directions, respectively.

(c) Stress Distribution in the x, y and xy directions, respectively.

Figure 3.4: Elastic Problem solution of the cantilever problem.

The observed results align with the anticipated outcome for the analyzed elastic problem.

First, the displacements along the x-direction are significantly smaller compared to those along the y-direction, which is consistent with the fixed displacement boundary condition on the left end of the cantilever and the vertical load. Moreover, stress peaks occur precisely where expected, considering the specific problem configuration involving a vertical load applied to the right end.

These findings confirm the validity and accuracy of the analysis, providing valuable insights into the deformation and stress distribution within the cantilever system. Further comparison with theoretical results has been discussed in previous thesis [25].

Chapter 4

Material Characterization

4.1 Introduction

In topology optimization, the material distribution within a given design space is optimized to achieve the desired performance, so accurately modeling the behavior of the material is crucial to the success of the task. This is especially important when dealing with composite materials, which are composed of multiple phases or constituents, like fibers and matrices, arranged in specific patterns that can significantly impact overall mechanical properties.

Composite materials are gaining increasing popularity in the field of structural design, thanks to advancements in material technology that have expanded their application possibilities. This broader range of options has led to an increased interest in creating optimized composite structures. The advent of 3D printing technologies, such as fused-deposition modeling, has further facilitated the manufacturing of high-quality fiber-reinforced composites. Taking advantage of this technological progress, a topology optimization scheme can be employed to optimize the design of composite materials, offering solutions that are demanding in terms of implementation but can now be realized.

However, analyzing and designing structures composed of nanomaterials can be computationally intensive. To address this challenge, multi-scale analysis offers an attractive approach by breaking down the complex structural model into simpler layers. Multi-scale optimization naturally follows, utilizing the insights gained from multi-scale analysis to facilitate the computationally intensive task of optimization. This approach allows for efficient and effective exploration of optimal solutions in the design of structures made from nanomaterials.

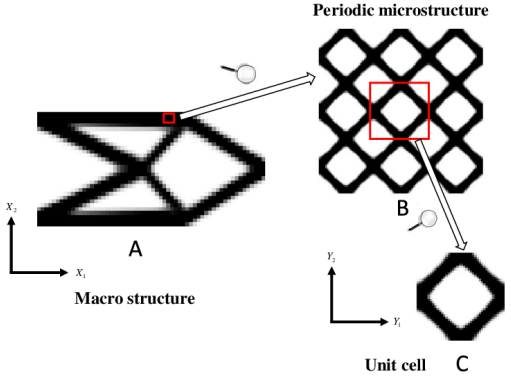


Figure 4.1: Representation illustrating the process of multi-scale topology optimization. Extracted from [26].

In formulating the optimal material design problem, two distinct scales are employed throughout the optimization process. The first scale pertains to the macro level, where the structural domain is defined and subjected to applied loads. The second scale focuses on the micro level, where the design domain, known as the unit cell, is formulated. To apply the topology optimization approach for achieving optimal material design, a key modification is made to the design variables, specifically the material densities: instead of assigning unknown material densities ρ to every finite element within the discretized macro domain, the unknowns ρ are assigned to each finite element within the micro design domain. This modification allows for a bridging between the two scales, which is accomplished through the utilization of homogenization theory. By employing this approach, the optimization process can effectively integrate and optimize materials at both the macro and micro scales. [27]

This chapter begins by presenting the concept of multi-scale analysis as a solution to the computational complexities associated with microstructural design. It then formulates the microscopic problem of material characterization in topology optimization and it states the equations that describe it. Also, the boundary conditions that must be considered for the 2D and 3D cases are defined. Finally, this chapter will showcase practical examples that demonstrate the values of the resulting matrix $\mathbb{C}$, the rendering of the 3D mesh after applying the boundary conditions and the solution to a specific micro problem.

4.2 Multi-scale Framework

For elastic materials, the interaction between the macro and micro scales is governed by two essential components: [28]

- From the macro to the micro scale: the localized macroscopic strain field $\varepsilon(x)$
- From the micro to the macro scale: the homogenized elasticity tensor $\mathbb{C}$

The core concept behind the multi-scale modeling approach lies in the assumption that each point x within the domain corresponds to a localized Representative Volume Element (RVE) consisting of a microscopic domain Ω_μ which captures the variations in material properties that are representative at a local scale. [29]

This enables the association of the macroscopic variable x with the coordinates of the macroscopic domain Ω, while the microscopic variable y corresponds to the coordinates of the microscopic domain Ω_μ.

In terms of kinematics, the microscopic strain $\varepsilon_\mu(x, y)$ can be decomposed into two components: the macroscopic strain $\varepsilon(x)$ and the fluctuation strain $\widetilde{\varepsilon}_\mu(x, y)$, as expressed in Equation (4.1).

$$\varepsilon_\mu(x, y) = \varepsilon(x) + \widetilde{\varepsilon}_\mu(x, y) \tag{4.1}$$

It is important to note that the fluctuation strain $\widetilde{\varepsilon}_\mu(x, y)$ is dependent on both the coordinates x and y and must satisfy the constraint of having a zero mean value over the microscopic domain Ω_μ due to its periodicity:

$$\int_{\Omega_\mu} \widetilde{\varepsilon}_\mu(x, y) = 0 \tag{4.2}$$

By combining both equations, it can be ensured that the average value of the microscopic strain corresponds to the macroscopic strain:

$$\varepsilon(x) = \frac{1}{\Omega_\mu} \int_{\Omega_\mu} \varepsilon_\mu(x, y) \tag{4.3}$$

Furthermore, considering the Hill-Mandel Principle of Macro-Homogeneity, it is postulated that all permissible macroscopic deformations δ_ε, when multiplied by the stress, would be equivalent to a potential combination of the macro deformation and its microscopic fluctuations.

$$\sigma : \delta_\varepsilon = \frac{a}{|\Omega_\mu|} \int_{\Omega_\mu} \sigma_\mu : \delta\varepsilon_\mu \tag{4.4}$$

So, the presence of micro fluctuations does not affect the total energy of the system, but it does contribute to the associated deformation. This gives rise to the mechanical equilibrium problem of the RVE, which involves determining an admissible microscopic displacement fluctuation field $\widetilde{u}_\mu$ for a given macroscopic strain ε.

4.3 Problem Formulation

The topology optimization (TO) micro problem for designing materials involves optimizing the distribution of material volume fraction in the unit cell design domain to optimize the structural response. This differs from the original TO problem where the design variables were density values X of the finite elements used to discretize the macro design domain, now they are densities x of the finite elements used to discretize the micro-unit cell design domain. The optimization procedure, therefore, involves two different scales - the macro scale and the micro scale (as shown in 4.2). The design variables and volume constraint are defined at the micro scale, while the objective function is set on the macro scale, but still expressed as a function of x, and the homogenization method is used to transition between the two scales through the elasticity tensor. [30]

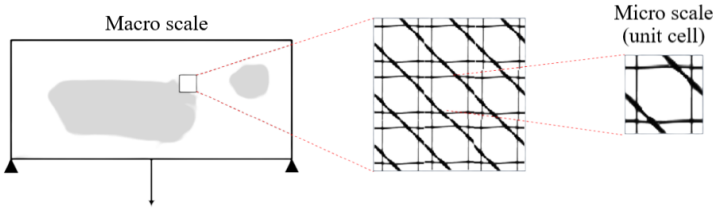


Figure 4.2: Diagram illustrating the micro-scale periodic unit cell (right) contained within the macro-scale structure (left). Extracted from [30].

Therefore, the mathematical expression for the conventional topology optimization problem is modified as shown in Equation (4.5); where $C(x)$ represents the compliance, F denotes the load vector, $U(x)$ the displacements derived from the structural analysis, $F = K \cdot U(x)$ the linear system of equations obtained from the FEM, $V(x)$ the volume of material obtained as a result of the densities x, V_0 denotes the material volume of the entire domain and f is the constraint applied in the form of volume fraction.

$$\begin{cases} min & C(x) = F^T \cdot U(x) \\ s.t. & F = K \cdot U(x) \\ & V(x)/V_0 = f \\ & 0 \leq x_e \leq 1 \end{cases} \tag{4.5}$$

The adjoint method is used to obtain the derivative of the objective function in order to minimize the compliance, which can be expressed as follows in Equation (4.6),

$$\frac{\partial C}{\partial x_e} = -U^T \cdot \frac{\partial K}{\partial x_e} \cdot U \tag{4.6}$$

where the derivative $\partial K/\partial x_e$ is computed as shown in Equation (4.7):

$$\frac{\partial K}{\partial x_e} = \frac{\partial \mathbb{C}}{\partial x_e} \cdot K_0 \tag{4.7}$$

On the other hand, when considering the microscopic scale, the problem can be formulated by assuming a linear relationship between the stress and strain tensors, as mentioned earlier. This implies that the micro problem involves providing a specific strain value and determining the corresponding stress value by applying the homogenized elasticity tensor $\mathbb{C}$.

Subsequently, by recognizing that the micro problem involves utilizing the homogenization theory, which involves the application of unit strains on both the unit cell domain and the finite elements used for discretization of the unit cell domain, and solving the Lagrangian function defined in Equation (4.8),

$$L(\omega_{ij}, \lambda) = \frac{1}{2}\frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}]\, \mathbb{C}\, [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] + \lambda \frac{1}{|\Omega|}\int_\Omega \varepsilon(\omega_{ij}) \tag{4.8}$$

it is determined that the Lagrange multiplier that satisfies this equation, and which plays a crucial role in solving the microscopic topology optimization problem, can be identified as:

$$\lambda = -\frac{1}{|\Omega|}\int_\Omega \sigma_\mu(\varepsilon_{ij}) = -\sigma^h(\varepsilon_{ij}) \tag{4.9}$$

For a more comprehensive understanding of the mathematical foundations of the homogenization theory, including the derivation of these equations, please refer to Appendix B.

Also, Equation (4.10) yields the expression for the resulting homogenized elasticity tensor.

$$\mathbb{C}_{i,j} = \frac{1}{V}\sum_{e=1}^{N}\int_{x_e}\left(u_e^{0(i)} - u_e^{(i)}\right)\cdot\mathbb{C}_{\mu}^{0}\cdot\left(u_e^{0(j)} - u_e^{(j)}\right)dV_e \tag{4.10}$$

Note that u are the displacements fields that result from the unit strains applied locally and the superscript 0 indicates the globally applied strains, so u^0 are the displacement fields that arise from the unit strains applied globally.

Hence, the resulting matrix form of the homogenized elasticity tensor $\mathbb{C}$ in a two-dimension system can be expressed as:

$$\mathbb{C} = \begin{pmatrix} (\mathbb{C})_{1111} & (\mathbb{C})_{1122} & (\mathbb{C})_{1112} \\ (\mathbb{C})_{1122} & (\mathbb{C})_{2222} & (\mathbb{C})_{2212} \\ (\mathbb{C})_{1112} & (\mathbb{C})_{2212} & (\mathbb{C})_{1212} \end{pmatrix} \tag{4.11}$$

4.4 Boundary Conditions

The boundary conditions discussed in Section 3.4 (the Dirichlet and Neumann conditions) pertain to the macroscopic scale of analysis of the studied problem. However, when solving micro topology optimization problems, it becomes essential to establish boundary conditions on a microscopic scale as well.

These microscopic boundary conditions are designed to maintain the periodicity of the material. This way, by implementing periodic boundary conditions, the analysis can be confined to a single representative unit cell, ensuring the consistent behavior of the material throughout the analysis process. [31]

4.4.1 2D case

The periodic boundary conditions considered in the 2D cases of this type of problem are enforced through a Master-Slave matrix that relates nodes on opposite sides of the domain to ensure consistent topology during optimization (refer to Figure 4.3 for a clearer illustration). So, its main objective is to guarantee that the behavior and characteristics observed on the left side of the unit cell are replicated on the right side, as well as between the lower and upper edges.

On Appendix C.1.1, the MATLAB class responsible of the generation of the MS matrix can be found.

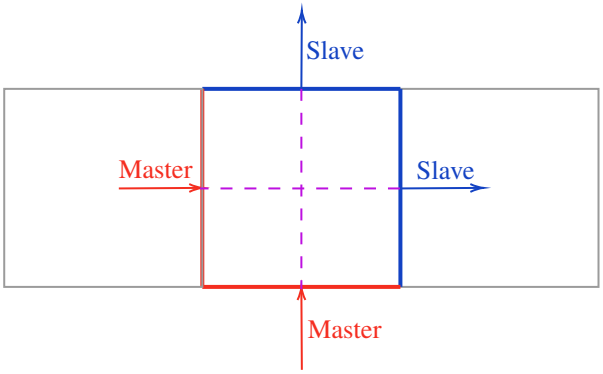


Figure 4.3: 2D square mesh cell with Master-Slave relations represented.

4.4.2 3D case

As for the periodic constraints in the 3D case, these are written in a Master-Slave matrix following the same methodology as in the 2D case. This matrix relates the nodes on opposite faces of the domain to ensure a consistent topology during optimization (see Figure 4.4 for a visual explanation). To compute it, the 2D code has been extended to three dimensions to generate the Master-Slave matrix (see Appendix C.1.2 for more details).

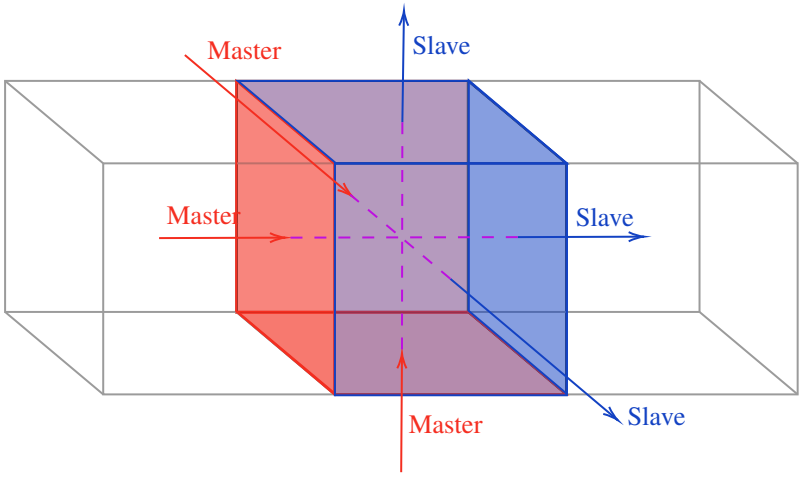


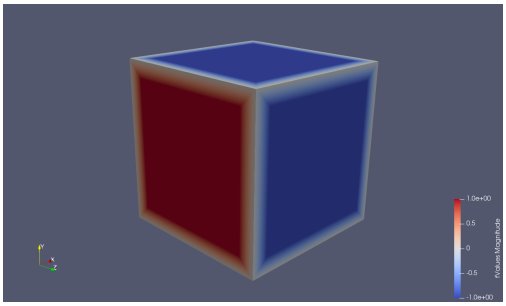
Figure 4.4: 3D cubic mesh cell with Master-Slave relations represented.

4.5 Examples

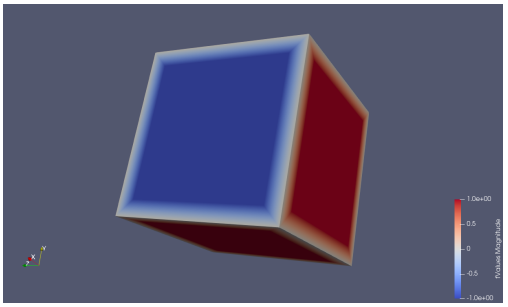
In this subsection, the material characterization parameters for the studied case and the micro problem solution will be presented as examples.

The mesh under study is a 3D cubic mesh with a width of 1 which includes a spherical hole at its center with a radius of 0.3 (see Appendix C.2). This example serves as a representative case to showcase the utilization of elasticity theory in analyzing the mechanical behavior of micro structures. The boundary conditions applied in this example are Dirichlet, which impose a displacement of 0 on the vertex nodes in all three dimensions, and the periodic conditions.

A visualization of the Master-Slave matrix in Paraview can be seen in Figure 4.5. Note that the Master nodes are highlighted in red and have associated a value of '1', while the Slave nodes are shown in blue and are labeled as '-1'.



(a) View 1.



(b) View 2.

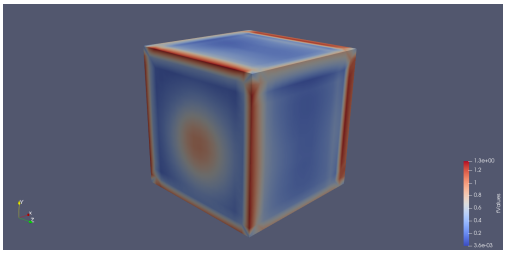
Figure 4.5: Paraview visualization of the Master-Slave matrix in a 3D cubic mesh cell.

Regarding the obtained homogenized elasticity tensor $\mathbb{C}$, it exhibits the following values:

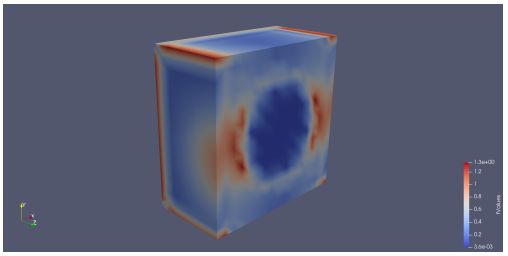
$$\mathbb{C} = \begin{pmatrix} 1.0168 & 0.4549 & 0.4549 & 2.4249 \cdot 10^{-4} & -3.3605 \cdot 10^{-4} & 1.0312 \cdot 10^{-4} \\ 0.4549 & 1.0169 & 0.4549 & 3.9157 \cdot 10^{-4} & -1.0664 \cdot 10^{-4} & 3.1842 \cdot 10^{-4} \\ 0.4549 & 0.4549 & 1.0169 & 7.9842 \cdot 10^{-5} & -3.2961 \cdot 10^{-4} & 3.2165 \cdot 10^{-4} \\ 2.4249 \cdot 10^{-4} & 3.9157 \cdot 10^{-4} & 7.9842 \cdot 10^{-5} & 0.2718 & -1.2982 \cdot 10^{-4} & 1.3310 \cdot 10^{-4} \\ -3.3605 \cdot 10^{-4} & -1.0664 \cdot 10^{-4} & -3.2961 \cdot 10^{-4} & -1.2982 \cdot 10^{-4} & 0.2718 & -1.3130 \cdot 10^{-4} \\ 1.0312 \cdot 10^{-4} & 3.1842 \cdot 10^{-4} & 3.2165 \cdot 10^{-4} & 1.3310 \cdot 10^{-4} & -1.3130 \cdot 10^{-4} & 0.2718 \end{pmatrix}$$

As observed, the matrix exhibits the expected values: non-zero values along the principal and orthogonal directions, while the remaining numbers possess such negligible magnitudes that they can be considered as approximately zero. It is worth noting that the obtained matrix exhibits symmetry, which aligns with the expected behavior given the specific case being studied.

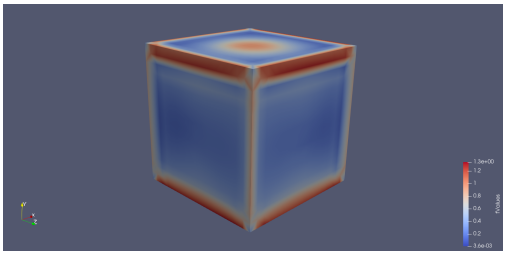
Lastly, using the established strain-displacement relations, equilibrium equations and constitutive equations, the micro elastic problem of the 3D cubic mesh was solved. The code successfully computed the displacement, strain and stress values throughout the structure. Results of the stress and displacement values in the x, y and z directions can be seen in Figures 4.6 and 4.7.



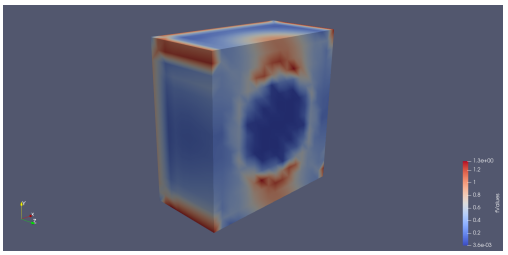
(a) Stress Distribution in the x-direction (whole cube).



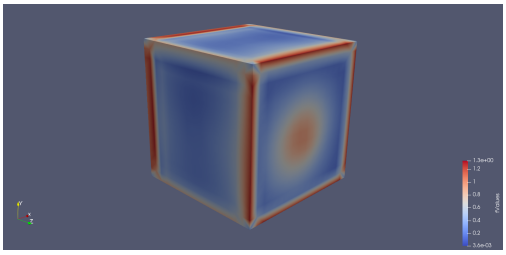
(b) Stress Distribution in the x-direction (half cube).



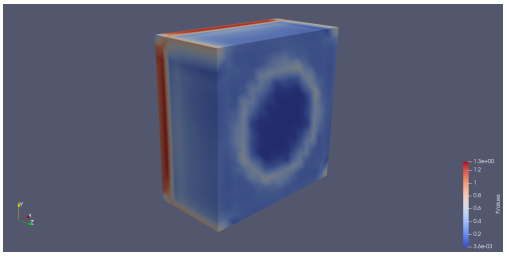
(c) Stress Distribution in the y-direction (whole cube).



(d) Stress Distribution in the y-direction (half cube).

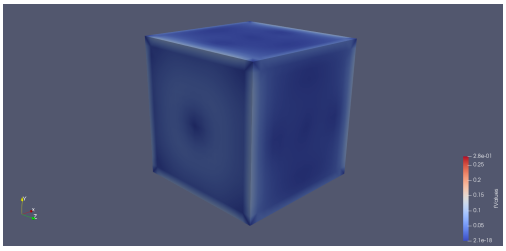


(e) Stress Distribution in the z-direction (whole cube).

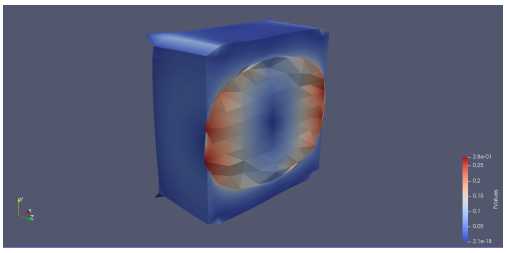


(f) Stress Distribution in the z-direction (half cube).

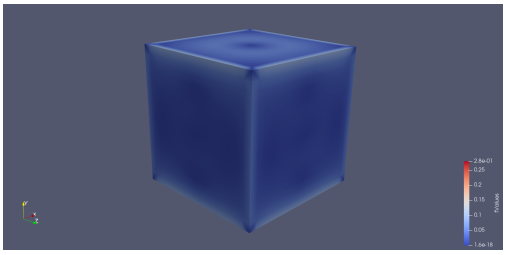
Figure 4.6: Stress Analysis of 3D Cubic Mesh with Hole.



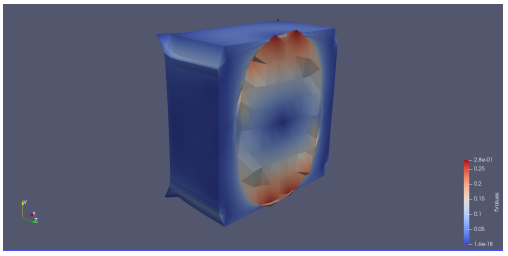
(a) Displacement Field in the x-direction (whole cube).



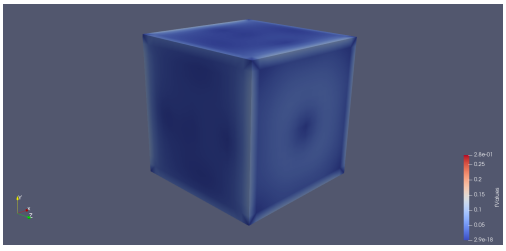
(b) Displacement Field in the x-direction (half deformed cube).



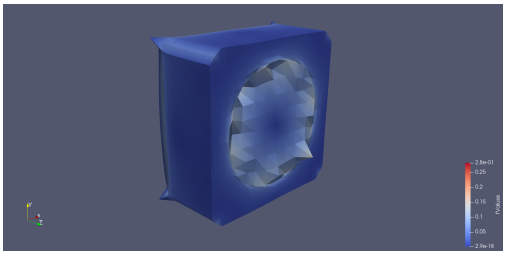
(c) Displacement Field in the y-direction (whole cube).



(d) Displacement Field in the y-direction (half deformed cube).



(e) Displacement Field in the z-direction (whole cube).



(f) Displacement Field in the z-direction (half deformed cube).

Figure 4.7: Displacement Analysis of 3D Cubic Mesh with Hole.

Chapter 5

Topology Optimization

As it has been mentioned on previous chapters, the goal of topology optimization is to find the optimal distribution of material within a given design space that maximizes or minimizes a certain performance criterion. This criterion can vary depending on the specific design problem, but it is typically related to the structural or mechanical behavior of the system under consideration. Some common performance criteria include maximizing stiffness, minimizing weight or optimizing heat transfer.

To begin the topology optimization process, the design space is usually defined as a geometric region with prescribed boundary conditions. These boundary conditions can include fixed displacements or forces, temperature gradients, or other relevant parameters that affect the behavior of the system. Once the design space is defined, the next step is to discretize it into a finite number of elements or cells, which form the basis for the optimization process. The boundary conditions for the problem under investigation have been specified in Section 3.4 and 4.4. In terms of the geometry, the study centers around a 3D cubic mesh composed of hexahedral elements with the addition of a spherical hole at its center.

The optimization process itself involves iteratively changing the distribution of material within the design space to achieve the desired performance criterion. This is typically done using mathematical algorithms that seek to optimize the topology of the structure by adding or removing material from certain regions of the design space.

One common approach in TO is to use the Finite Element Method (FEM) to model the behavior of the system. FEM involves dividing the design space into small elements, where the material and geometric properties of each element are specified. This allows to model the behavior of the system in a computationally efficient manner and to carry out the optimization process efficiently. See Appendix D for a more detailed explanation of the FEM method.

This chapter will present the topology optimization (TO) problem and the equations that describe it. It will also explain the shape functionals that describe the material properties and introduce two different approaches to solving the problem: density-based and level set-based methods. Finally, a set of examples will be presented to illustrate the application of these methods and algorithms in practical problems.

5.1 General Problem Formulation

From a mathematical perspective, topology optimization aims to find the material distribution that minimizes a specified objective function J, while adhering to various constraints C_k. These constraints can include a volume constraint ($C_0 \leq 0$) and additional design variable constraints ($C_k \leq 0$, where $k : 1...N$). Also, the optimization process is governed by a state equation that can be either linear or non-linear in nature. [32]

In order to maintain the generality of the TO problem definition, the variable $\chi(x)$ will be introduced, which represents the design variable at a given point x. Depending on the specific method employed, $\chi(x)$ will be denoted as $\rho(x)$ or $\phi(x)$ for density-based and level set-based approaches, respectively. Further details regarding these methods will be elaborated in the subsequent sections.

Building upon this concept, the traditional mathematical formulation of the general topology optimization problem is expressed as follows in Equation (5.1).

$$\begin{cases} min & J\left(u\left(\chi\right),\chi\right) \equiv \int_{\Omega} j\left(u\left(\chi\right),\chi,x\right) d\Omega \\ s.t. & C_0(\chi) \equiv \int_{\Omega} c_0\left(\chi,x\right) d\Omega \leq 0 \\ & C_k(\chi) \leq 0; \quad k : 1,...,N \\ & \text{state equation} \end{cases} \tag{5.1}$$

In this formulation, the objective function J is defined as the integration of a local function $j(u(\chi),\chi,x)$ across the entire domain, while the constraint functional C_0 represents the volume constraint in relation to the design variable χ. Moreover, specific constraint equations $C_k(\chi)$ can be included to incorporate additional constraints on the design variables, which may vary depending on the chosen approach.

Additionally, the state equation provides the solution in the form of an unknown field $u(\chi)$ for a given optimal design χ. In this thesis, the state equation considered corresponds to the discretization of the domain into finite elements, and the formulation of the TO problem is expressed as follows:

$$\begin{cases} min & J\left(u\left(\chi\right),\chi\right) \equiv \int_{\Omega} j\left(u\left(\chi\right),\chi,x\right) d\Omega \\ s.t. & C_0(\chi) \equiv \int_{\Omega} c_0\left(\chi,x\right) d\Omega \leq 0 \\ & C_k(\chi) \leq 0; \quad k : 1,...,N \\ & K\left(\chi\right) U\left(\chi\right) = F\left(\chi\right) \end{cases} \tag{5.2}$$

where the stiffness matrix is represented by $K\left(\chi\right)$, while the external force vector is denoted as $F\left(\chi\right)$.

As an illustrative instance, the widely known compliance problem can be formulated by seeking to minimize the objective function of structural compliance, given by $J = U^T K U$, while imposing a constraint on the material usage expressed as $C_0 = V/V_0 - V^* \leq 0$. [33]

5.2 Shape Functionals

As it has been stated, in topology optimization shape functionals serve as objective or constraint functions that quantify the performance of a given design. The performance can be defined in various ways, such as maximizing stiffness or minimizing stress concentrations, and the shape functionals reflect these objectives.

The volume shape functional, in particular, is a type of objective function used to optimize the size of a design domain. Volume is a crucial parameter in topology optimization, playing a significant role in achieving structurally efficient designs with minimized cost and weight. Typically, compliance is minimized while the volume is commonly utilized as a constraint in this process, as it is a more tractable problem for optimization solvers. [34]

The computation of the volume can be seen in Equation (5.3), where ρ corresponds to the nodal design variable, V^* denotes the target volume fraction and V_0 represents the total geometric volum of the design domain.

$$V = \frac{\int_\Omega \rho d\Omega}{V^* V_0} - 1 \tag{5.3}$$

Regarding its gradient, it can be determined through the following computation:

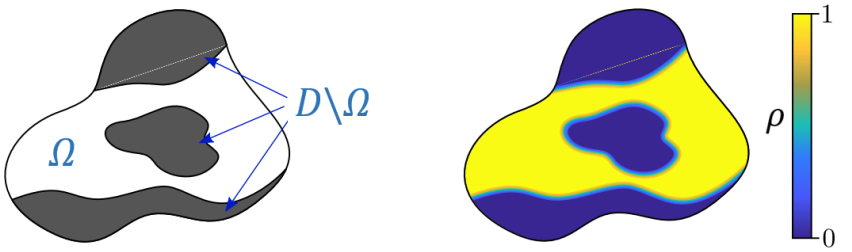
$$\frac{\partial V}{\partial \rho} = \frac{\int_\Omega d\Omega}{V^* V_0} - 1 \tag{5.4}$$

5.3 Density-Based Methods

Topology optimization is a form of structural optimization that offers significant design flexibility by optimizing the arrangement of a limited quantity of material within a specified design domain denoted as D [35]. Typically, the material distribution within D is represented by an indicator function $\rho(x)$ defined over the domain D, such that:

$$\rho(x) = \begin{cases} 1 & \forall x \in \Omega \quad \text{(material)} \\ 0 & \forall x \in D \setminus \Omega \quad \text{(void)} \end{cases} \tag{5.5}$$

where the open set Ω corresponds to the region occupied by the material. An example depicting the continuous design representation of a generic domain, with black representing the void region and white representing the material region, can be seen in Figure 5.1.



(a) Discrete separation of domains (b) Continuous design representation

Figure 5.1: Example of a 2D density-based domain. Extracted from [36].

When considering the binary-valued variable ρ as the independent optimization variable, the problem transforms into a combinatorial optimization problem which can pose significant challenges in terms of computational complexity. Even for relatively moderate problem sizes, solving such problems can become highly difficult. Addressing this concern is frequently regarded as the primary obstacle in topology optimization, and numerous strategies have been studied in order to solve it.

In density-based approaches, the binary constraint (5.5) is relaxed and the design variables ρ are treated as continuous volume densities, allowing them to assume values within the range of [0,1]. This transformation results in a continuous nonlinear optimization problem.

Density-based methods offer several advantages in the field of topology optimization [37]. Firstly, they exhibit computational efficiency by effectively utilizing storage capacity and reducing CPU time. This is achieved by employing only one free variable per element or node, depending on the implementation.

Secondly, they demonstrate robustness as they can be applied to various combinations of design constraints. Unlike other methods that are limited to compliance or equivalent designs, density-based methods can handle more complex design conditions. This flexibility arises from the fact that the optimal microstructure is not predetermined and can be determined based on the specific design requirements.

Additionally, they allow for the adjustment of penalization, providing the opportunity to optimize computational efficiency by selecting the most suitable penalization value.

However, it is important to note that these methods are not without limitations. The solutions obtained from them are influenced by the degree of penalization or interpolation, such as the p-value in the case of SIMP. As a result, there is a possibility that the solutions may not converge to the optimal solution. Nonetheless, it is worth mentioning that other methods face a similar limitation to a similar extent. [38]

5.3.1 SIMP Method

The widely used Solid Isotropic Material with Penalization (SIMP) technique is employed to achieve optimized designs that predominantly consist of material ($\rho = 1$) or void ($\rho = 0$) [39]. In the SIMP approach, intermediate densities are penalized by adjusting the density/stiffness interpolation as illustrated below:

$$C = \rho^p C^0 \tag{5.6}$$

where C stands for the constitutive tensor and C^0 for the elasticity tensor of the material. Note that, to discourage intermediate densities, a penalization parameter $p > 1$ is introduced, which reduces their efficiency. Usually, the value of p is set to 3 to facilitate algorithm convergence.

Moreover, the penalization factor can be determined based on the Poisson's ratio, as the SIMP model assumes it to be density-independent [10]. To ensure compliance with the Hashin-Shtrikman (HS) bounds for two-phase materials, the SIMP interpolation must satisfy the condition stated in Equation (5.7), which inherently implies a value greater than 3. Note that the first inequality stems from the bulk modulus bound, while the second is a result of the shear modulus bound.

$$p \geq p^*\left(\nu^0\right) = \max\left\{\frac{2}{1-\nu^0}, \frac{4}{1+\nu^0}\right\} \tag{5.7}$$

Numerous modifications of the SIMP method have been suggested, such as the SIMP-ALL technique [40]. This approach incorporates the concept of topological derivative to improve the interpolation process, resulting in more rigid structures at a lower computational expense. Additionally, it establishes a more distinct correlation with microstructures.

5.3.2 RAMP Method

The Rational Approximation of Material Properties (RAMP) is another interpolation method that addresses certain limitations of the SIMP approach, particularly for low density values when loads depend on the design variable. One key advantage of the RAMP model is its ability to maintain nonzero sensitivity even at zero density, a feature lacking in SIMP. This characteristic enables the RAMP method to overcome certain numerical challenges associated with structures containing regions of extremely low density. [41]

The RAMP method offers a solution by theoretically guaranteeing a concave design space, in contrast to the SIMP method when the penalty factor exceeds a specific threshold. This improvement in the RAMP method enhances its performance and reliability for optimizing structures with varying density distributions, thereby providing more accurate results.

The mathematical formulation of the RAMP method is expressed as follows:

$$C = \frac{\rho}{1 + p\,(1 - \rho)} C^0 \tag{5.8}$$

In the RAMP interpolation scheme, the value of the penalization parameter p should be chosen to be greater than 0. Based on numerical experiments, it has been found that a value of $p = 4$ yields good results.

5.3.3 Comparative Analysis: SIMP vs RAMP

Figure 5.2 illustrates the interpolation of the Young's modulus for the SIMP and RAMP material schemes for different p values in function of the relative density.

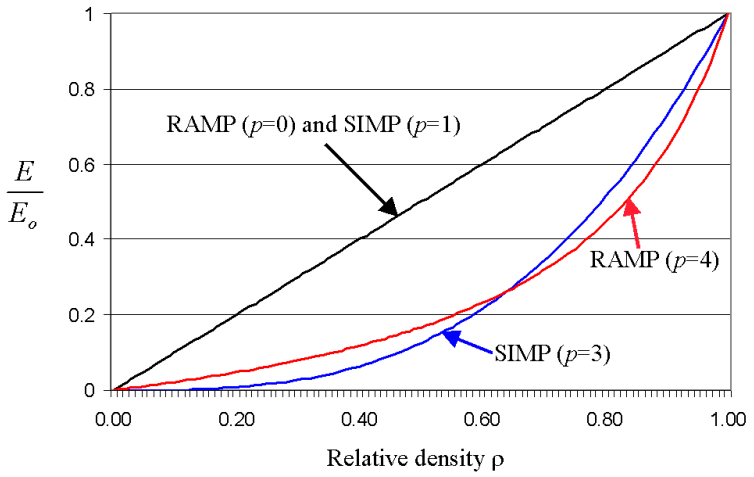


Figure 5.2: Diagram illustrating the SIMP and RAMP methods for different p values in function of the relative density. Extracted from [42].

In the case of SIMP with $p = 1$ and RAMP with $p = 0$, the material interpolation corresponds to the thickness optimization problem, resulting in many elements with intermediate densities. Increasing the penalization factor p leads to higher computational costs for elements with intermediate densities, as they require a higher density to achieve the same stiffness as in the thickness optimization problem. Consequently, a larger penalty factor reduces the number of elements with intermediate densities. However, it is important not to choose an excessively large penalty factor, as it significantly increases the likelihood of converging to a local minimum during the optimization process.

Therefore, a suitable compromise for the penalty factor lies within the range of 2 to 4, taking into account the results of numerical experiments.

5.3.4 Problem Formulation

Finally, rewritting Equation (5.2), the mathematical formulation of the optimization problem employing density-based methods can be expressed as:

$$\begin{cases} \underset{\rho}{min} & J\left(u\left(\rho\right),\rho\right) \equiv \int_{\Omega} j\left(u\left(\rho\right),\rho,x\right) d\Omega \\ s.t. & 0 \leq \rho \leq 1 \\ & C_k(\rho) \leq 0; \quad k:1,...,N \\ & K\left(\rho\right) U\left(\rho\right) = F\left(\rho\right) \end{cases} \tag{5.9}$$

In this formulation, the design variable is represented by the density ρ, which is constrained to the interval between 0 and 1. This range ensures that the density values reflect the amount of material present in the design, where $\rho = 0$ corresponds to void or empty regions, and $\rho = 1$ represents fully dense or solid regions. By defining the design variable as a density value within this range, it enables the representation of varying material distributions throughout all the design domain.

5.4 Level Set-Based Methods

Level set-based methods are another widely used approach in the field of topology optimization. Unlike the density-based ones, they rely on flexible implicit functions to define structural boundaries instead of explicitly parameterizing the design domain. [43]

In them, the level set function is utilized to describe the structural domain, where the zero-level contour defines the boundary of the structure. Throughout the optimization process, the zero-level contour of the level set function is displaced in a favorable direction using shape sensitivity analysis. The implicit nature of the interface allows for topological modifications of the zero-level contour, enabling crisp boundary designs, where the intermediate densities are constrained to a narrow band around the structure's boundary.

From a mathematical perspective, the configuration of the material domain Ω, delimited by the boundary Γ, can be expressed in relation to the level set function $\Phi(x)$ as follows:

$$\begin{cases} \Phi(x) < 0 & \forall x \in \Omega & \text{(material)} \\ \Phi(x) = 0 & \forall x \in \Gamma & \text{(interface)} \\ \Phi(x) > 0 & \forall x \in D \setminus \bar{\Omega} & \text{(void)} \end{cases} \tag{5.10}$$

Figure 5.3 presents an illustration of a level set function along with the corresponding material domain.

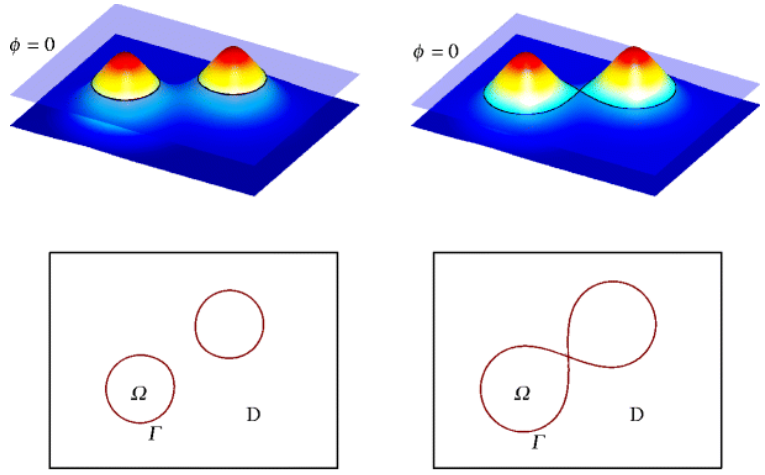


Figure 5.3: Example of a 2D level set function and its corresponding zero-level contours representing the material domain. Extracted from [44].

5.4.1 Level Set Function Characterization

The advancement of conventional level set methods originated from the work of Wang et al. [45], who recognized the importance of linking point velocity, design sensitivity and the level set method for defining structural boundaries in the optimization process. By representing the structural boundary as an implicit level set iso-surface defined by a scalar function in 3D, where k is an arbitrary iso-value and x is a point on the surface S, the critical relationship was established as:

$$S = \{x : \Phi(x) = k\} \tag{5.11}$$

In the course of the structural optimization process, multiple iterations are performed to modify the surface. Consequently, the level set function can be formulated to incorporate the temporal dimension:

$$S = \{x(t) : \Phi(x(t), t) = k\} \tag{5.12}$$

By taking the derivative with respect to time and applying the chain rule, the following Hamilton-Jacobi-type equation can be obtained [46], which establishes an initial value problem for the time-dependent function ($\Phi(x, 0) = \Phi_0(x)$):

$$\frac{\partial \Phi(x,t)}{\partial t} + \nabla \Phi(x,t) \frac{dx}{dt} = 0 \tag{5.13}$$

During the solution process, the movement of a point (dx/dt) is determined based on the optimization objective, taking into account the position of x and the surface geometry at that point. So, the optimal structural boundary is obtained as the solution to a partial differential equation defined by Equation (5.14), where $\Gamma(x, \Phi)$ represents the speed vector of the level set, which is influenced by the optimization objective.

$$\frac{\partial \Phi(x,t)}{\partial t} = -\nabla\Phi(x,t)\frac{dx}{dt} \equiv -\nabla\Phi(x,t)\Gamma(x,\Phi) \tag{5.14}$$

5.4.2 Problem Formulation

Finally, by reexpressing Equation (5.2), the mathematical representation of the optimization problem utilizing level set-based methods can be stated as follows:

$$\begin{cases} \underset{\Phi}{min} & J(u,\Phi) \equiv \int_\Omega j(u,\Phi,x)\, d\Omega \\ s.t. & C_k(\rho) \leq 0; \quad k:1,...,N \\ & K(H(\Phi))\, U(\Phi) = F(\Phi) \\ & \int_\Omega \frac{H(\Phi) d\Omega}{V_0} \leq V^* \end{cases} \tag{5.15}$$

where $H(\Phi)$ represents a Heaviside function that takes the value of 1 for positive Φ values and 0 otherwise. [47]

However, the level set function in the employed SWAN repository's code is updated using a SLERP algorithm [48], leading to the following outcome:

$$\phi_{n+1} = \frac{1}{sin\theta_n}\left[sin\left((1-\kappa_n)\,\theta_n\right)\phi_n + sin\left(\kappa_n\theta_n\right)\frac{g_n}{||g||}\right] \tag{5.16}$$

where θ represents the angle that quantifies the similarity between the level set function and the gradient defined in Equation (5.17), κ denotes the line search and can range from 0 to 1, and $g_n/|g|$ corresponds to the normalized gradient.

$$\theta = arccos\left[\frac{(\phi_n, g_n)}{||\phi_n||\,||g_n||}\right] \tag{5.17}$$

5.5 Examples

In this section, a comparison between density-based and level set-based methods will be conducted by visualizing the optimization process of the same case. The case involves a 3D cubic mesh with a width of 1, featuring a hole inclusion of 0.2. The mesh is subject to Dirichlet conditions, which prescribe a zero displacement at its vertex nodes, and periodic constraints, implemented using the master-slave matrix discussed in the previous chapter.

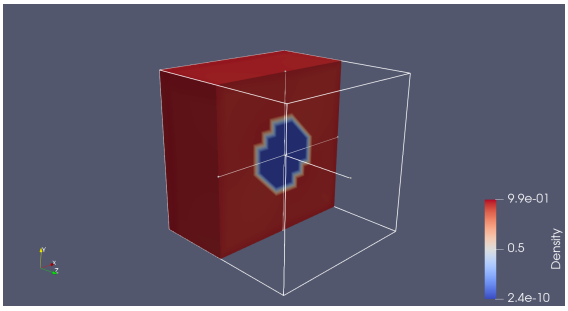
Table 5.1 summarizes the shared conditions of the simulations. Mention that the cost, the α and β parameters will be defined on Chapter 6.

SIMULATION CONDITIONS	
Initial case	'sphereInclusion'
Cost	'chomog_alphabeta' (J_1)
Constraint	'volumeConstraint'
Final volume fraction	0.7
Alpha	[1 1 0 1 0 0]'
Beta	[1 1 0 1 0 0]'

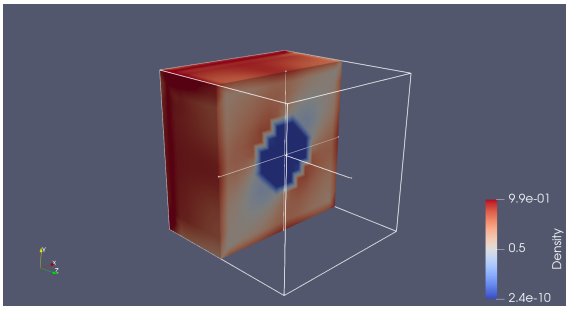
Table 5.1: Common conditions in Density-based vs Level Set-based method simulations

Next, the results obtained from both methods are presented. It is important to note that, for improved visualization of the optimization process within the mesh, the cubic mesh has been cropped in half to expose its interior.

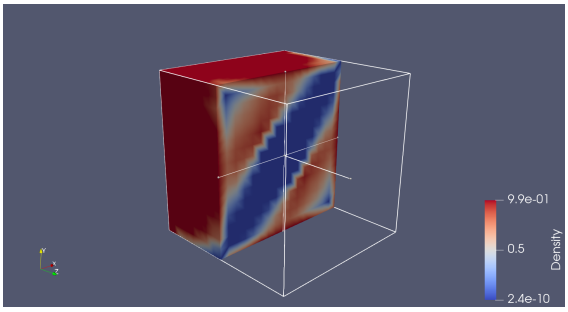
Density-Based Methods



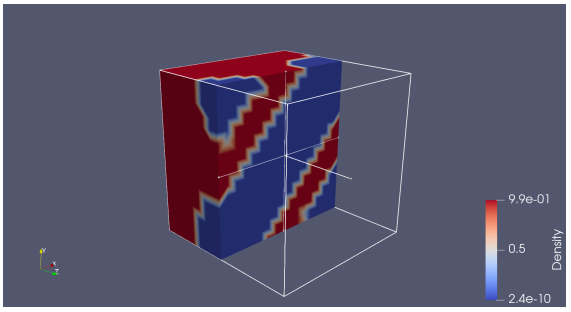
(a) Step 1.



(b) Step 25.



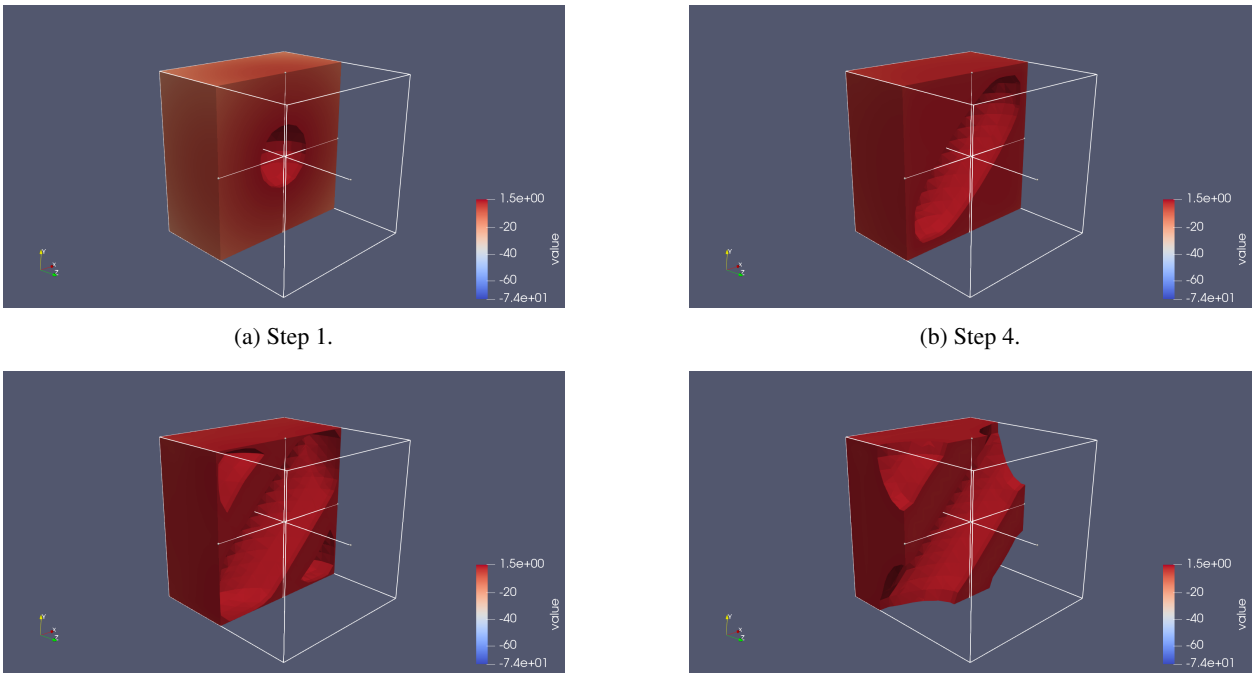
(c) Step 40.



(d) Step 150

Figure 5.4: Density results from the MMA - Density-based optimizer.

Level Set Methods



(a) Step 1.

(b) Step 4.

(c) Step 33.

(d) Step 150

Figure 5.5: Density results from the Null Space (SLERP) - Level set-based optimizer.

As can be seen, Figure 5.4 displays the characteristic grayscale values associated with density-based methods. As previously explained, these methods relax the density and allow the domain to assume values between 0 and 1, as indicated by the range bar.

On the other hand, it is important to mention that in order to enhance the visualization of the level set results, the Paraview tool *IsoVolume* was utilized. This tool facilitated the elimination of all values greater than 0, considering the range of values that can be assumed by the level set function defined in Equation (5.10), as it is shown in Figure 5.5.

Both methods, both methods led to the same resulting mesh, which aligns with the expected outcome, as the directions to optimize defined by the α and β vectors were the x, y and xy directions. However, the speed and manner in which material was removed varied due to the distinct mathematical approaches employed by each method.

Chapter 6

Material Design

Achieving desired material design requires the utilization of objective functions in the optimization process. These objective functions, denoted as J, play a crucial role in shaping the optimization problem and enabling the attainment of optimal solutions. By carefully defining these objective functions, it can be specified the desired material properties, performance criteria and constraints, effectively guiding the optimization process towards finding the most suitable material designs.

This chapter delves into the topic of material design, focusing on the challenges and techniques involved in designing structures at the micro scale. It introduces three important matrix functions: the weighted inverse homogenized elasticity matrix function, the rational weighted inverse homogenized elasticity matrix function and the inverse problem matrix function. These functions provide mathematical expressions for evaluating the gradient in optimization processes, offering insights into optimizing the effective properties of microstructures. Finally, the chapter concludes with practical examples of material design in both 2D and 3D settings, showcasing the application and effectiveness of the discussed techniques.

6.1 Weighted Inverse Homogenized Elasticity Matrix Function

As previously mentioned, the homogenized two-dimensional elasticity tensor $\mathbb{C}$ for microstructural design is represented by Equation (6.1).

$$\mathbb{C} = \begin{pmatrix} (\mathbb{C})_{1111} & (\mathbb{C})_{1122} & (\mathbb{C})_{1112} \\ (\mathbb{C})_{1122} & (\mathbb{C})_{2222} & (\mathbb{C})_{2212} \\ (\mathbb{C})_{1112} & (\mathbb{C})_{2212} & (\mathbb{C})_{1212} \end{pmatrix} \tag{6.1}$$

In the case of orthotropic symmetry, the effective material properties, including the Young's modulus, bulk modulus, shear modulus and Poisson ratios, can be explicitly determined from the components of the compliance tensor $\mathbb{C}^{-1}$. The matrix representation of the compliance tensor is given by Equation (6.2), where E_1 and E_2 represent the effective Young's moduli along the orthotropy directions e_1 and e_2, respectively, G corresponds to the effective in-plane shear modulus, and ν_{12} and ν_{21} represent the effective Poisson ratios. [49]

$$\mathbb{C}^{-1} = \begin{pmatrix} (\mathbb{C}^{-1})_{1111} & (\mathbb{C}^{-1})_{1122} & 0 \\ (\mathbb{C}^{-1})_{1122} & (\mathbb{C}^{-1})_{2222} & 0 \\ 0 & 0 & (\mathbb{C}^{-1})_{1212} \end{pmatrix} = \begin{pmatrix} \frac{1}{E_1} & \frac{-\nu_{12}}{E_1} & 0 \\ \frac{-\nu_{21}}{E_2} & \frac{1}{E_2} & 0 \\ 0 & 0 & \frac{1}{G} \end{pmatrix} \tag{6.2}$$

Note that the effective Poisson ratios and Young's moduli are subject to the following condition:

$$\frac{\nu_{21}}{E_2} = \frac{\nu_{12}}{E_1} \tag{6.3}$$

In this particular problem, the objective function to be minimized is represented by Equation (6.4), where the vectors α and β are user-defined. This construction allows the user to optimize specific components of the homogenized elasticity tensor through the use of α and β vectors.

$$min \;\; J_1\left(\mathbb{C}(\chi)\right) = \alpha^T \mathbb{C}^{-1} \beta \tag{6.4}$$

Furthermore, it is worth noting that the derivative of the function J_1 can be expressed as:

$$\frac{\partial J_1(\mathbb{C})}{\partial \chi} = -\alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\chi} \cdot \mathbb{C}^{-1} \right) \cdot \beta \tag{6.5}$$

where $d/d\chi$ represents the topological derivative, which can alternatively be denoted as D_T.

For a more detailed mathematical demonstration, please refer to Appendix E.2.

6.2 Rational Weighted Inverse Homogenized Elasticity Matrix Function

This specific function is a modified version of the preceding one sated on Equation (6.4), also employed for generating microstructures with optimized effective properties. It is referred to as the fraction-weighted inverse elasticity matrix, and it can be seen in Equation (6.6).

$$min \;\; J_2\left(\mathbb{C}(\chi)\right) = \frac{\alpha^T \mathbb{C}^{-1} \beta}{\alpha^T \mathbb{C}^{-1} \alpha} + \frac{\beta^T \mathbb{C}^{-1} \alpha}{\beta^T \mathbb{C}^{-1} \beta} \tag{6.6}$$

To calculate the gradient of J_2, the derivative of the function can be determined as follows:

$$\frac{\partial J_2(\mathbb{C})}{\partial \chi} = \frac{g_1}{(\alpha^T \mathbb{C}^{-1} \alpha)^2} + \frac{g_2}{(\beta^T \mathbb{C}^{-1} \beta)^2} \tag{6.7}$$

$$\begin{cases} g_1 = \alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\chi} \cdot \mathbb{C}^{-1}\right) \cdot \left[\alpha \cdot \left(\alpha^T \mathbb{C}^{-1} \beta\right) - \beta \cdot \left(\alpha^T \mathbb{C}^{-1} \alpha\right)\right] \\ g_2 = \beta^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\chi} \cdot \mathbb{C}^{-1}\right) \cdot \left[\beta \cdot \left(\beta^T \mathbb{C}^{-1} \alpha\right) - \alpha \cdot \left(\beta^T \mathbb{C}^{-1} \beta\right)\right] \end{cases} \tag{6.8}$$

For a more comprehensive mathematical proof, please consult Appendix E.3.

6.3 Inverse Homogenization Matrix Function

This matrix formulation follows a distinct approach known as the Inverse Homogenization Problem, which sets it apart from the previously discussed methods. This problem represents a paradigm shift in material design, offering a complementary approach to the traditional homogenization methods.

6.3.1 Inverse Homogenization Problem

Material design through inverse homogenization involves determining the optimal micro-architecture configuration of a composite material, characterized by the presence or absence of material, to achieve an effective elasticity tensor $\mathbb{C}$ that matches a desired target tensor $\mathbb{C}^*$. [50]

This process considers two characteristic scale lengths: the macro-scale length (denoted as l) which corresponds to the overall structural size and the micro-scale length (denoted as l_μ) which pertains to the characteristic length of the material's micro-architecture. Mention that the effective elasticity tensor is defined and analyzed at the macro-scale, as illustrated in Figure 6.1, while the actual material design is carried out at the micro-scale.

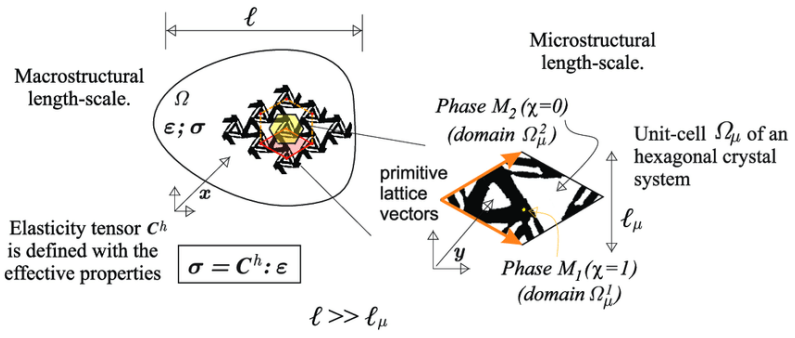


Figure 6.1: Inverse homogenization problem for the determination of the characteristic function χ in a periodic composite material. Extracted from [51].

So, the essence of inverse homogenization lies in starting from desired macroscopic properties and tailoring the microscale to match these features, thereby interchanging the roles of known and unknown scales compared to direct homogenization approaches.

6.3.2 Problem Formulation

Consider a periodic composite structure consisting of two isotropic elastic phases, M_1 and M_2. Within a basic micro-cell denoted as Ω_μ, the domains occupied by M_1 and M_2 are depicted as Ω_1 and Ω_2, respectively (refer to Figure 6.1). [52]

The function $\chi(x)$ within the micro-cell Ω_μ indicates the positions occupied by phase M_1, as defined by Equation (6.9). It is important to note that $\chi(x)$ is a binary function acting as a design variable that determines the distribution of the two materials on the domain.

$$\chi(x) = \begin{cases} 1 & \forall x \in \Omega_\mu^1 \quad \text{(material 1)} \\ 0 & \forall x \in \Omega_\mu^2 \quad \text{(material 2)} \end{cases} \tag{6.9}$$

The homogenized elasticity tensor of the composite, denoted as $\mathbb{C}$, is clearly influenced by the geometrical arrangement of the phases M_1 and M_2 within the micro-cell domain. To make this dependency explicit, it is introduced the notation $\mathbb{C}(\chi)$. Mention that the evaluation of this tensor within Ω_μ involves enforcing periodic boundary conditions on displacement fluctuations too.

So, the inverse design problem is then formulated as a topology optimization problem, which can be stated as follows: given the design domain Ω_μ and the desired effective elasticity tensor $\mathbb{C}^*$, determine the characteristic function χ that satisfies the following conditions:

$$\begin{cases} \underset{\chi}{min} & J(\chi) \equiv \frac{1}{|\Omega_\mu|} \int_{\Omega_\mu} \chi d\Omega \\ s.t. & \|\mathbb{C}(\chi) - \mathbb{C}^*\| = 0 \end{cases} \tag{6.10}$$

Note that when the M_2 phase is assumed to be void, the problem can be characterized as a minimum weight problem.

6.3.3 Matrix Function

The objective function that characterizes the inverse homogenization problem, aiming to minimize the discrepancy with a desired homogenized elasticity matrix, can be expressed as follows:

$$min \;\; J_3\left(\mathbb{C}(\chi)\right) = \left[\mathbb{C} - \mathbb{C}^*\right]^2 = \left[\begin{pmatrix} \mathbb{C}_{11} & \mathbb{C}_{12} & \mathbb{C}_{13} \\ \mathbb{C}_{21} & \mathbb{C}_{22} & \mathbb{C}_{23} \\ \mathbb{C}_{31} & \mathbb{C}_{32} & \mathbb{C}_{33} \end{pmatrix} - \begin{pmatrix} \mathbb{C}^*_{11} & \mathbb{C}^*_{12} & \mathbb{C}^*_{13} \\ \mathbb{C}^*_{21} & \mathbb{C}^*_{22} & \mathbb{C}^*_{23} \\ \mathbb{C}^*_{31} & \mathbb{C}^*_{32} & \mathbb{C}^*_{33} \end{pmatrix} \right]^2 \tag{6.11}$$

where $\mathbb{C}$ is the homogenized elasticity tensor and $\mathbb{C}^*$ the desired effective elasticity tensor.

However, mention that this set of 9 equations can be simplified to 6 through the utilization of symmetry considerations [53]. Additionally, these equations have to be appropriately rescaled to prevent the occurrence of small values (except for $\mathbb{C}_{13}$ and $\mathbb{C}_{23}$, which are set to 0 in the target elasticity matrix for the studied case). The resulting set of equations is provided below.

$$\begin{cases} \dfrac{\mathbb{C}_{11} - \mathbb{C}^*_{11}}{\mathbb{C}^*_{11}} = 0 \\ \dfrac{\mathbb{C}_{22} - \mathbb{C}^*_{22}}{\mathbb{C}^*_{22}} = 0 \\ \dfrac{\mathbb{C}_{33} - \mathbb{C}^*_{33}}{\mathbb{C}^*_{33}} = 0 \end{cases} \qquad \begin{cases} \sqrt{2}\left(\mathbb{C}_{13} - \mathbb{C}^*_{13}\right) = 0 \\ \sqrt{2}\left(\mathbb{C}_{23} - \mathbb{C}^*_{23}\right) = 0 \\ \sqrt{2}\left(\dfrac{\mathbb{C}_{12} - \mathbb{C}^*_{12}}{\mathbb{C}^*_{12}}\right) = 0 \end{cases} \tag{6.12}$$

Regarding its gradient, the derivative of the objective function can be defined as:

$$\frac{\partial J_3\left(\mathbb{C}\right)}{\partial \chi} = 2\left(\mathbb{C} - \mathbb{C}^*\right)\frac{\partial \mathbb{C}}{\partial \chi} \tag{6.13}$$

For a more comprehensive mathematical proof, please consult Appendix E.4.

6.4 Examples

Next, to provide a visual explanation, 2D and 3D simulations using the cost functions J_1, J_2 and J_3 will be presented. These simulations serve as concrete examples to illustrate the effectiveness and versatility of the different proposed approaches in material design.

6.4.1 2D case

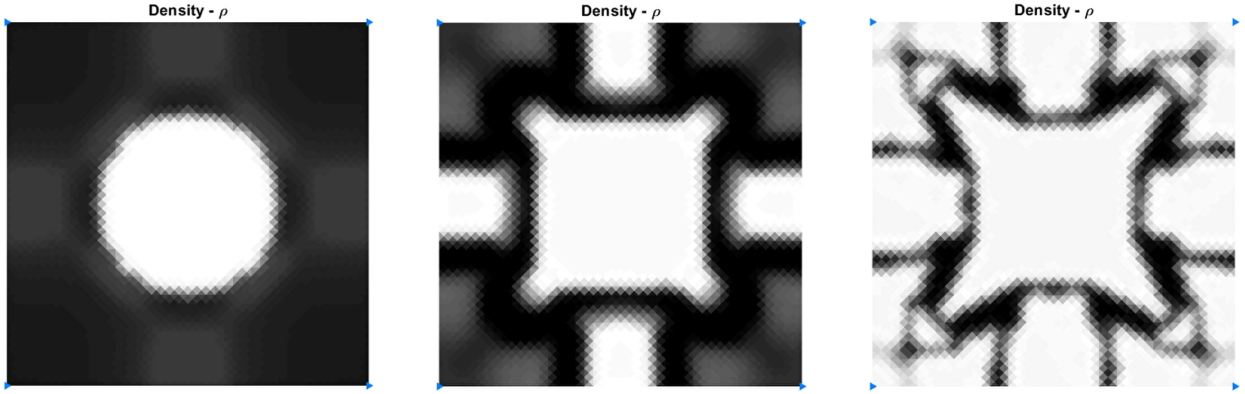
For the 2D problem, the same test will be conducted using both cost functions to enable an efficient comparison. The studied case consist of a squared mesh with a circle inclusion on its center, and the parameters used in the optimization process are as follows:

SIMULATION CONDITIONS	
Initial case	'circleInclusion'
Constraint	'volumeConstraint'
Constraint case	'INEQUALITY'
Optimizer	'MMA'
Design variable	'Density'
Final volume fraction	0.35
Alpha	[1 0 0]'
Beta	[0 -1 0]'

Table 6.1: Common conditions in 2D J_1, J_2 and J_3 simulations

The results obtained from optimizing the square mesh for both cost functions are presented below:

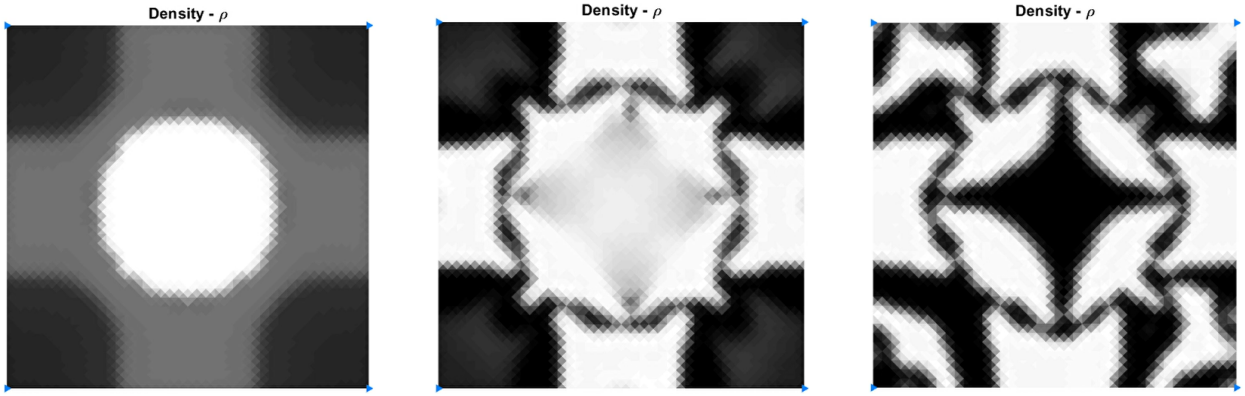
J_1 - cost = 'chomog_alphabeta'



(a) Step 1. (b) Step 15. (c) Step 30.

Figure 6.2: Density results from the 2D J_1 cost function.

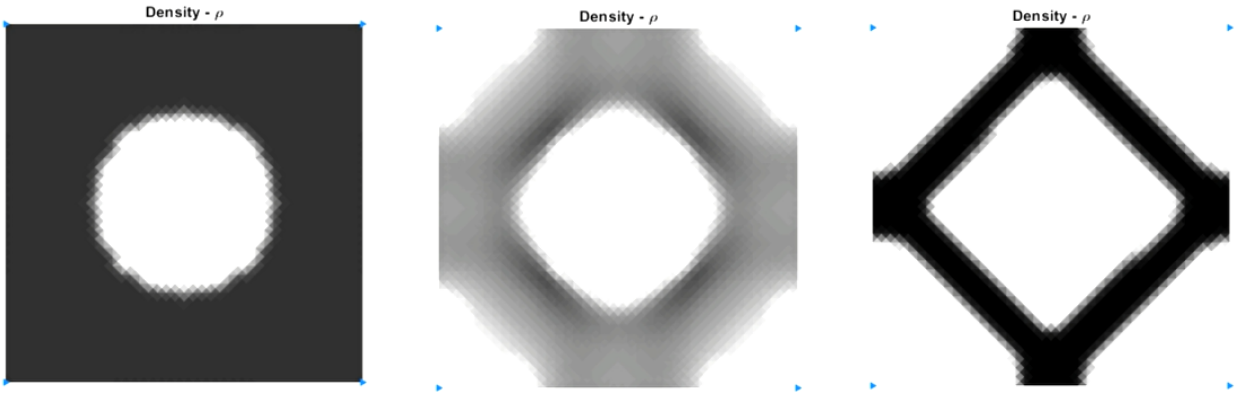
J_2 - cost = 'chomog_fraction'



(a) Step 1. (b) Step 30. (c) Step 60.

Figure 6.3: Density results from the 2D J_2 cost function.

J_3 - cost = 'enforceCh_CCstar_L2'



(a) Step 1. (b) Step 15. (c) Step 60.

Figure 6.4: Density results from the 2D J_3 cost function.

Mention that, on the J_3 simulation, instead of using the alpha and beta variables, a new variable called *selectiveC_Cstar* is introduced. This variable represents the desired $\mathbb{C}^*$ and takes the value '*IsotropyHexagon*'.

As anticipated, the three simulations resulted in significantly different material distributions, as each case optimizes a different cost function. Moreover, it is apparent that the convergence rate of the density distributions varies between the cases, despite utilizing the same optimizer.

6.4.2 3D case

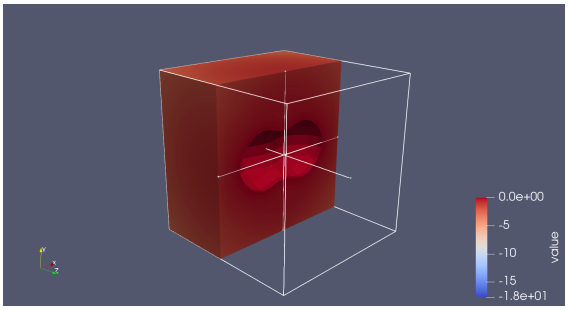
In the case of the 3D problem, both cost functions will be applied too to the same test in order to facilitate a meaningful comparison. The test case involves a cubic mesh with a spherical inclusion at its center, and the optimization process will utilize the following parameters:

SIMULATION CONDITIONS	
Initial case	'sphereInclusion'
Constraint	'volumeConstraint'
Optimizer	'NullSpace'
Optimizer unconstrained	'SLERP'
Design variable	'LevelSet'
Final volume fraction	0.7
Alpha	[1 0 0 0 0 0]'
Beta	[1 0 0 0 0 0]'

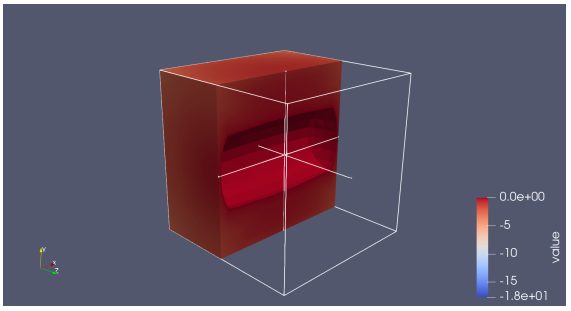
Table 6.2: Common conditions in 3D J_1 and J_2 simulations

The optimization results for both cost functions applied to the cubic mesh are displayed below.

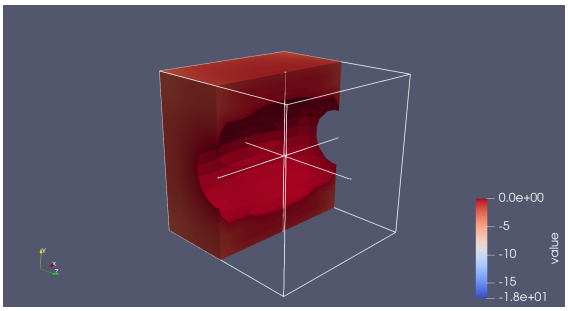
J_1 - cost = 'chomog_alphabeta'



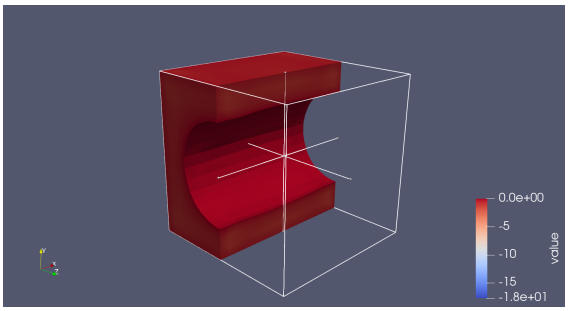
(a) Step 1.



(b) Step 5.



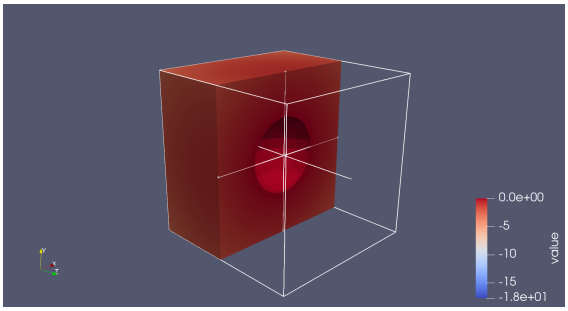
(c) Step 15.



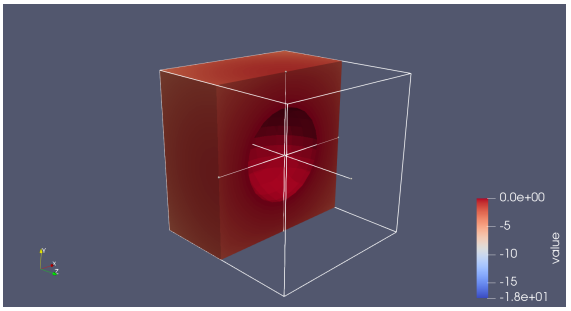
(d) Step 50

Figure 6.5: Density results from the 3D J_1 cost function.

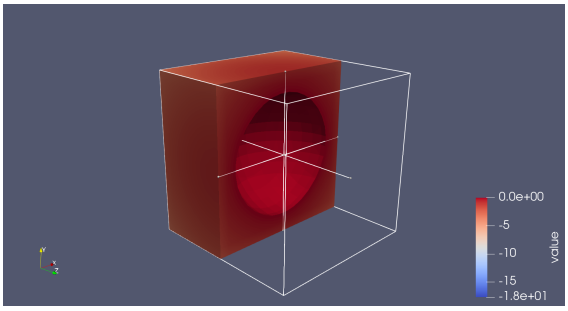
J_2 - cost = 'chomog_fraction'



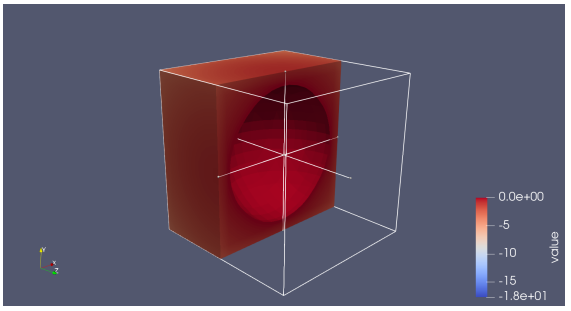
(a) Step 1.



(b) Step 3.



(c) Step 8.



(d) Step 40

Figure 6.6: Density results from the 3D J_2 cost function.

As observed, the results of both 3D simulations differ due to the utilization of distinct cost functions, similar to the 2D case. When setting $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$, it becomes apparent that the configuration minimizing J_1 corresponds to a centrally located cylinder aligned along the x-axis, whereas the configuration minimizing J_2 corresponds to a centered sphere.

It is important to note that for the J_3 cost function, only 2D simulations have been conducted due to the code's incompatibility for optimizing the J_3 function on 3D meshes. Nevertheless, the implementation of 3D optimization will be considered as a future development of the topic, aiming to address optimization in three dimensions (see Section 9.2).

Chapter 7

Methodology

This chapter provides an overview of the approach and techniques used in the research project. It encompasses three main sections: the clean code introduction, which took place during the initial months of the thesis; the SWAN software familiarization, which followed the clean code phase; and the subsequent 3D optimization simulations. These sections represent significant milestones in the project, highlighting the emphasis on improving code quality, mastering the SWAN software and conducting advanced simulations to optimize the system. Together, they contribute to the overall progression and outcomes of the study.

7.1 Clean Code Introduction

The initial step undertaken in this final thesis involved acquiring knowledge about clean code programming. This was essential since the project involved collaborative software development, necessitating the utilization of appropriate techniques to ensure the production of high-quality code for Swan.

7.1.1 GitHub

The development of the Topology Optimization code for this project involved a team of students, each working on their independent final thesis related to Topology Optimization or FEM. To facilitate collaborative software development, the team utilized GitHub, a web-based platform designed for version control and collaboration among software developers.

Version control plays a crucial role in helping developers manage and track changes to a software project's code. It enables them to work safely through branching and merging. Branching allows developers to create separate copies of the source code, allowing them to make changes without affecting the main project. Once the changes are tested and deemed successful, they can be merged back into the main source code, ensuring the integrity of the project. This entire process is carefully tracked, enabling developers to revert changes if necessary.

So, GitHub played a crucial role in promoting collaboration among developers, providing distributed version control that allows teams to seamlessly work together in a centralized Git repository. This enabled efficient tracking of changes and ensured organizational coherence throughout the development process. Also, GitHub extends beyond version control, offering a comprehensive hosting service, web interface for Git code repositories and management tools that facilitate seamless collaboration among developers. [54]

7.1.2 Object-Oriented Programming

When developing a new code, it is essential to adhere to established guidelines to maintain coherence with previous developers' work. In the case of Swan, object-oriented programming (OOP) is utilized, making it imperative to understand its principles and application.

OOP is a programming paradigm centered around "objects" that encompass both data and code. Data is represented through fields (referred to as attributes or properties), while code is expressed through procedures (known as methods). See Figure 7.1 for a visual example.

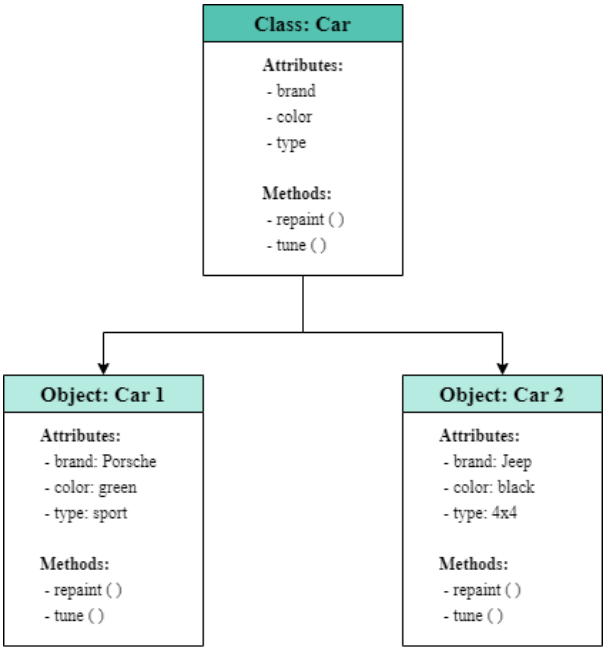


Figure 7.1: Diagram illustrating a class (car) with two object instances, depicting its attributes and methods.

OOP proves particularly advantageous for large, intricate and actively maintained programs such as the SWAN code. It simplifies the development and maintenance processes by incorporating fundamental concepts like abstraction, inheritance, polymorphism and encapsulation. These four pillars serve as guiding principles that contribute to efficient and effective software development: [55]

- Abstraction: it enables the hiding of implementation details from users while providing them with only essential information. By employing it, developers gain the flexibility to make modifications and additions to the codebase with ease and over time.
- Inheritance: it allows the acquisition of properties from a pre-existing class (parent class) into a newly created class (child class), promoting the reuse of code.
- Polymorphism: it enables the creation of methods with identical names but varying method signatures, facilitating the development of code that is clean, sensible, readable and resilient.

- Encapsulation: it serves as a mechanism for binding data and functions within an entity, safeguarding against external interference and misuse. Consequently, it contributes to enhanced security by protecting both data and functions.

Additionally, it is worth noting that properties and methods can possess access modifiers that dictate their accessibility. Three common access modifiers are:

- Public: it allows the property or method to be accessed from any location. It is the default modifier if none is specified.
- Protected: it restricts access to the property or method within the class and by classes derived from it.
- Private: it strictly limits access to the property or method exclusively within the class itself.

7.1.3 Clean Coding Practices

When coding, it is important to write code with the intention of being read by others in the future, rather than solely for personal use. Even when working on individual projects, ensuring that code is easily understandable to other developers is very is crucial.

Therefore, regardless of the programming language utilized, adopting certain practices can assist in producing clean and improved code:

- Naming convention: adhering to a proper naming convention is crucial for facilitating future edits and updates. For example, in the SWAN code, functions are written in lowercase and words are separated by capitalizing the first letter of each word (e.g., computeValue).
- Avoid poor naming: it is advisable to use meaningful variable names that strike a balance between being too short or excessively long.
- Code indention: proper indentation enhances code readability, making it easier to locate specific sections within the code.
- Concise methods: lengthy functions tend to be harder to comprehend, modify and reuse. It is recommended to aim for shorter methods, focusing on performing a single task or serving a specific purpose.
- Reduce nested conditions: excessive nesting of conditions can make code difficult to read. Simplifying nested conditions and utilizing ternary operators are recommended practices.
- Avoid code duplication: instead of copying and pasting the same code with minor modifications, strive to create reusable functions that can be called from multiple locations.

By incorporating these practices, code can become more readable, maintainable and conducive to collaboration.

7.1.4 UML Diagrams

To facilitate the analysis of code sequences using MATLAB and visually depict the logical relationships, the adoption of UML diagrams have been introduced.

Unified Modeling Language (UML) serves as a visual representation method for illustrating the architecture, design and implementation of intricate software systems. When working with extensive codebases, consisting of thousands of lines, it becomes challenging to track the intricate relationships and hierarchies within the software system. UML diagrams help in compartmentalizing the software system into components and subcomponents. [56]

In practice, UML standards define 13 different types of diagrams. However, considering the utilization of object-oriented programming on the SWAN code, the employed UML diagram type is the class diagram.

Class diagrams specifically portray the static structure of a system, encompassing classes, their attributes, behaviors and the relationships among classes. Represented by a rectangular shape, a class diagram consists of three vertically stacked compartments. The top compartment is mandatory and displays the class name, while the bottom two compartments provide details regarding the class attributes and methods.

Figure 7.2 depicts the main relationships in an UML, while Figure 7.3 provides an accompanying example that corresponds to these relationships.

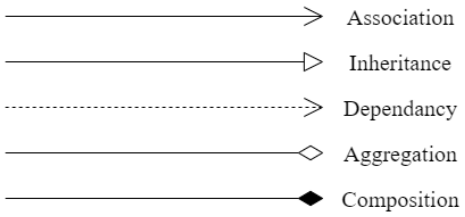


Figure 7.2: UML main relationships.

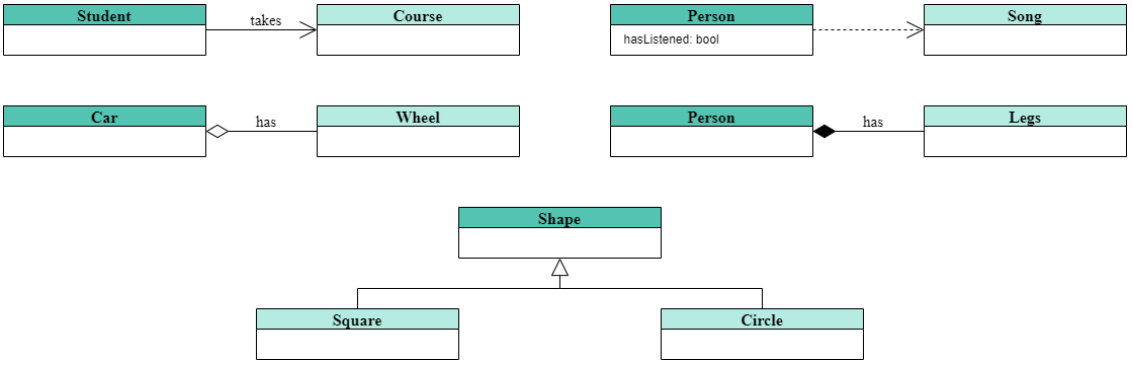


Figure 7.3: UML main relationships examples.

7.2 SWAN Software Familiarization

Once familiar with the fundamentals of Object-Oriented Programming and the functioning of an OOP code, the focus shifts towards exploring the codebase of the SWAN repository.

Swan represents a MATLAB-based Topology Optimization software with a comprehensive range of capabilities. Presently, it enables the resolution of 2D and 3D topology optimization challenges utilizing density or level-set as design variables. Swan's core objective is to provide a versatile framework, characterized by a modular design, facilitating the swift implementation of new optimization techniques or functionalities. [57]

Within Swan's existing functionality, it offers support for macro and micro scale problems, employing design variables that include Density and Level Set. In terms of optimization techniques, the software incorporates several implemented optimizers for density-based problems (such as Projected Gradient, MMA and IPOPT) and for level set-based problems (like SLERP, Projected SLERP and Hamilton-Jacobi).

To comprehend the sequence of the Finite Element Method (FEM) code within the Swan repository, rigorous debugging and analysis were performed using MATLAB. Furthermore, to enhance understanding regarding the interaction between classes and the properties and methods each class encompasses, a graphical representation in the form of a Unified Modeling Language (UML) diagram was created (as demonstrated in Appendix F.2).

In addition, specific methods were subjected to in-depth study, such as the $unfittedMesh$ method responsible for classifying mesh parts and their respective boundaries. For better comprehension, a visual example illustrating the method and an accompanying UML diagram were generated (refer to Appendix F.3).

7.3 3D Optimization

After gaining familiarity with the code in the SWAN repository and comprehending the principles of object-oriented programming and its associated clean code techniques, the next step is to initiate the adaptation of the existing micro 2D code to facilitate its compatibility with 3D functionality.

7.3.1 3D Mesh Creation - GID Software

First, the creation of the mesh was undertaken utilizing the GID software.

GID serves as an interactive graphical user interface specifically designed for defining, preparing and visualizing all the pertinent data associated with numerical simulations, combining preprocessing and postprocessing functionalities. This encompasses data pertaining to geometry, materials, conditions, solution information and various parameters. Additionally, the software possesses the capability to generate finite element analysis meshes and appropriately format the information for numerical simulation programs.

Consequently, the 3D mesh employed for the simulations exhibits the following characteristics:

- Vertices coordinates: (0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0) and (1, 1, 1)
- Element type: hexahedra
- Size of the elements: 0.0866025
- Mesh: structured
- Number of division per line: 20
- Problem type data: Swan

Also, to ensure proper data interpretation by MATLAB, it was necessary to incorporate the following conditions as part of the Problem Data:

PROBLEM DATA	
Unit System	SI
Dimensions	3D
Type of Problem	Plane Stress
Physical Type	Elastic
Macro / Micro	Micro

Table 7.1: 3D Mesh problem data

Figure 7.4 illustrates the 3D mesh generated using GID Simulation.

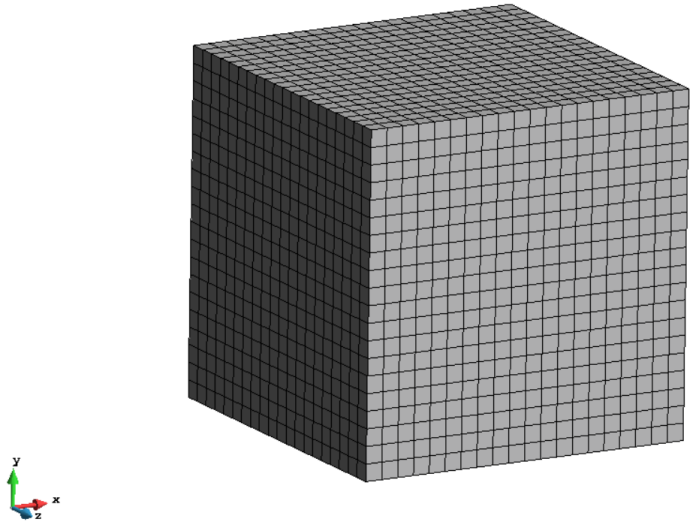


Figure 7.4: Generated 3D mesh.

7.3.2 Boundary Conditions

After the completion and calculation of the mesh, a MATLAB file containing the coordinates of all nodes and the connectivities of each element is generated. The subsequent step involves imposing the desired boundary conditions on the mesh.

In the specific case under study, the first objective is to restrict the movement of the vertex nodes (Dirichlet condition). To accomplish this, the identification of node vertices is necessary, as depicted in Code 7.1. As can be seen, the *verticesObtention* function extracts the coordinates of specific vertices from a given mesh by comparing the node coordinates with predefined vertex coordinates, and the resulting vertex coordinates are returned as an output.

```
function vertices = verticesObtention(gidcoord)
    vertexCoords = [
        0 0 0;
        0 0 1;
        0 1 0;
        1 0 0;
        0 1 1;
        1 0 1;
        1 1 0;
        1 1 1
    ];
    vertices = gidcoord(ismember(gidcoord(:,2:4), vertexCoords, 'rows'), :);
end
```

Code 7.1: Mesh node vertices determination.

Once the vertices have been identified, to ensure that they remain stationary, a prescribed displacement of 0 in all three directions have to be implemented. The developed code is presented in Code 7.2.

```
vertices = verticesObtention(gidcoord);

dirichlet_data = [ ];
for i = 1:length(vertices)
    verticeNumber = vertices(i,1);
    for j = 1:3
        dirichlet_data = [dirichlet_data;[verticeNumber j 0]];
    end
end
```

Code 7.2: Dirichlet boundary condition for the nodes establishment.

Also, the establishment of Master-Slave conditions is crucial for enforcing periodic boundary conditions and ensuring consistent topology during optimization. These conditions relate nodes on opposite sides of the domain and play a significant role in the analysis. So, the task is to generate a matrix that distinguishes between master and slave nodes. This will be achieved by assigning a value of 0 to indicate a master node and a value of 1 to indicate a slave node.

The specific function developed is shown in Code 7.3, which takes as input the *gidcoord* which represents the coordinates of nodes in the mesh and *vertices* which contains the vertex nodes. It iterates through each row of *gidcoord* and checks if the node coordinates match any of the vertices. If a node is not a vertex, it determines whether any of its coordinates are equal to 0 or 1. For each node that meets these conditions, it adds a row to the MS matrix with the node number and a corresponding value of 0 or 1. Finally, the function removes any duplicate rows from MS to ensure uniqueness.

```
function MS = get_MasterSlave(gidcoord, vertices)
    MS = [];
    for i = 1:length(gidcoord)
        nodeCoords = gidcoord(i, 2:4);
        if ismember(gidcoord(i,:), vertices, 'rows')
            continue; % Skip if node is a vertex
        end
        if any(nodeCoords == 0)
            MS = [MS; [gidcoord(i, 1), 0]];
        end
        if any(nodeCoords == 1)
            MS = [MS; [gidcoord(i, 1), 1]];
        end
    end
    MS = unique(MS, 'rows'); % Remove repeated rows
end
```

Code 7.3: Generation of the master-slave matrix.

7.3.3 Master-Slave Adaptation

To ensure the tessellation property of the generated cubic mesh, the Topology Optimization code utilizes a matrix of master-slave nodes. Each row in the matrix represents a pair of nodes, where the first column denotes the master node and the second column represents its corresponding slave node. In order to support 3D cases, it is essential to make appropriate modifications to the existing *MasterSlaveRelator* class. This adaptation enables the code to effectively handle 3D scenarios and generate the required master-slave node information.

The class first initializes itself with the coordinates of the mesh nodes, storing these coordinates internally for further computation.

Next, it computes the nodes present in different faces of the mesh. It identifies the nodes located on the minimum and maximum boundaries of each face, excluding the corner nodes, and it stores them separately for each face.

The class then proceeds to compute the master-slave relations for each face of the mesh by comparing the nodes present on the minimum and maximum boundaries of a given face. Based on specific criteria, such as proximity, it determines the appropriate master and slave nodes for each face. The resulting master-slave pairs are stored in a matrix, where the first column represents the master nodes and the second column represents their corresponding slave nodes.

To ensure the integrity of the master-slave relationships and avoid redundancy, the class includes a method to eliminate repeated edges in the master-slave matrix. It identifies common nodes between the master and slave columns and removes the corresponding rows from the matrix, resulting in a clean and non-redundant list of master-slave pairs.

The 2D version of the class can be found in Code C.1.1, while Code C.1.2 showcases the adapted version specifically designed for 3D scenarios. Despite being part of the same class, they have been separated to facilitate the comprehension of their distinct functionalities in their corresponding scenarios.

7.3.4 Hexahedral Mesh Adjustment

Another issue that had to be adressed was the incompatibility of using tetrahedral elements for mesh generation, as originally intended, due to their inherent asymmetry (as will be detailed in Section 8.4). Consequently, the decision was made to utilize hexahedral elements as an alternative.

The challenge arose from the disparity in the number of Gaussian points between tetrahedral elements (1) and hexahedral elements (4), which the existing code was not equipped to handle. Consequently, certain functions within the LHS and RHS integrators required adaptation.

For instance, consider the $assembleVector$ function within the RHS. This function's purpose is to construct a global vector by aggregating contributions from element vectors computed at each degree of freedom and integration point.

To address the inclusion of hexahedral elements, adjustments were made to the code, as demonstrated in Code 7.4. It can be seen that the method takes an object obj and a matrix F as inputs and then it computes the vector V by iterating over degrees of freedom ($dofs$) and integration points (Gaussian points), adding the computed vector $Fadd$ to V at each iteration.

```
function V = assembleVector(obj, F)
    dofsInElem  = obj.computeDofConnectivity();
    ndofPerElem = obj.dim.ndofsElem;
    ndof        = obj.dim.ndofs;
    ngaus       = size(F,2);
    V = zeros(ndof,1);
    for iDof = 1:ndofPerElem
        for igaus = 1:ngaus
            dofs = dofsInElem(iDof,:);
            c = squeeze(F(iDof,igaus,:));
            Fadd = obj.computeAddVectorBySparse(dofs, c, ndof);
            % Fadd = obj.computeAddVectorByAccumarray(dofs, c, ndof);
            V = V + Fadd;
        end
    end
end
```

Code 7.4: Vector assembly from element contributions at dofs and gauss points.

7.3.5 Simulations

In this study, a comprehensive analysis was conducted using multiple levels of comparison. The investigation began with 2D and 3D simulations, allowing for a comparison between the two-dimensional and three-dimensional optimization processes.

Table 7.2 presents the studied cases for the 2D scenarios.

2D Simulations	**Case 1**	$\alpha = \beta = [1\ 0\ 0]$
	Case 2	$\alpha = [1\ 0\ 0]$ and $\beta = [0\ 1\ 0]$
	Case 3	$\alpha = [1\ 0\ 0]$ and $\beta = [0\ 0\ 1]$
	Case 4	$\alpha = [1\ 1\ 0]$ and $\beta = [1\ 0\ 0]$
	Case 5	$\alpha = \beta = [1\ 1\ 0]$
	Case 6	$\alpha = [1\ 1\ 1]$ and $\beta = [1\ 1\ 0]$

Table 7.2: Different 2D cases simulated

In the context of 3D simulations, the study focused on two distinct categories: normal materials and metamaterials microstructures. A key aspect of the investigation involved comparing the performance and effectiveness of two optimization approaches: the MMA optimizer, which employs the density-based method, and the Null Space optimizer, which utilizes a level set-based method. This comparative analysis aimed to assess the capabilities and outcomes of each optimization approach within the context of optimizing normal materials.

Furthermore, within the context of normal materials, an additional comparison was performed by varying the final volume fractions. Specifically, the optimization process was executed for two different volume fractions: 0.7 and 0.85. The objective of this analysis was to explore the impact of altering the final volume fraction on the resulting microstructures and optimization outcomes.

Table 7.3 provides an overview of the varied parameters used to simulate each case, facilitating their visualization and comparison.

3D SIMULATIONS (NORMAL MICROSTRUCTURES)	**Optimizer**	MMA (Density-based method)	
		Null Space (Level Set method)	
	Cases	Cilinder	$\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$
		Vertical plate	$\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$
		Sphere	$\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$
		Diagonal	$\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$
		Shear	$\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$
	Final volume fraction	$Vfinal = 0.7$	
		$Vfinal = 0.85$	

Table 7.3: Different 3D cases simulated for normal microstructures

Finally, the 3D simulations for metamaterials microstructures were conducted employing the Null Space optimizer with different values of α and β, as well as varying the final volume fraction. This allowed for a comprehensive comparison of the results obtained for each case.

Table 7.4 displays the specific values that each parameter took during the simulations.

3D SIMULATIONS (METAMATERIALS MICROSTRUCTURES)	**Cases**	Case 1	$\alpha = [1\ 0\ 0\ 0\ 0\ 0]$ and $\beta = [0\ -1\ 0\ 0\ 0\ 0]$
		Case 2	$\alpha = [1\ 1\ 0\ 0\ 0\ 0]$ and $\beta = [0\ 0\ -1\ 0\ 0\ 0]$
		Case 3	$\alpha = [1\ 0\ 0\ 0\ 0\ 0]$ and $\beta = [-1\ 0\ 0\ 0\ 0\ 0]$
	Final volume fraction	$Vfinal = 0.7$	
		$Vfinal = 0.85$	

Table 7.4: Different 3D cases simulated for metamaterials microstructures

Additionally, to ensure a fair comparison across all 3D cases, the simulations were conducted with the same tolerance in the volume constraint. The tolerance was set to 10^{-3}, ensuring consistency in evaluating the fulfillment of the volume constraint for each optimization.

7.3.6 Parameters Adequation

One issue that required attention was the parameter adjustments for each optimizer in different types of simulations. Depending on the optimizer and the test conditions (such as the desired final volume, various α and β values, etc.), certain parameters needed to be manually adjusted.

For instance, in the case of the Null Space optimizer, the following parameters had to be modified depending on the specific case:

- **aJmax.** This parameter determined the initial strength of the optimizer and its ability to remove material. Larger values could result in reduced efficiency and longer computation times, while lower values could cause the optimizer to add material instead of removing it, leading to a volume fraction of 1 (representing a fully filled cube). The value of aJmax depended on each individual case, particularly the desired final volume, and typically ranged between 2 and 10.
- **aGmax.** This parameter influenced the optimizer's ability to handle constraints and adhere to them. For example, the value of λ present on the monitorings should be within the range of 10^0 to 10^1, and it was determined that a constant value of aGmax = 0.05 yielded efficient results across all tested cases.
- **eta.** This parameter determined the fraction of volume that could be removed between iterations. Considering the specific case (initially a cube with a spherical inclusion), optimal values of eta for obtaining smooth solutions can be found in Code 7.5.

```
1  ...
2
3  if obj.nIter==0
4      obj.eta = 0.025;
5  else
6      if obj.aG ≤ 0.5*obj.aGmax
7          obj.eta = 0.025;
8      else
9          obj.eta = 0.001;
10     end
11 end
12
13 ...
```

Code 7.5: Definition of eta value extracted from lines 152-160 of the *OptimizerNullSpace.m* code.

In the case of the MMA optimizer, the variable that required attention was **c**, which determines the optimizer's strength. Through experimentation, it was determined that the optimal power of 10 order for the given mesh and optimization process was:

```
1  ...
2
3  obj.c = 10*ones(obj.m,1);
4
5  ...
```

Code 7.6: Definition of c value extracted from line 169 of the *Optimizer_MMA.m* code.

Chapter 8

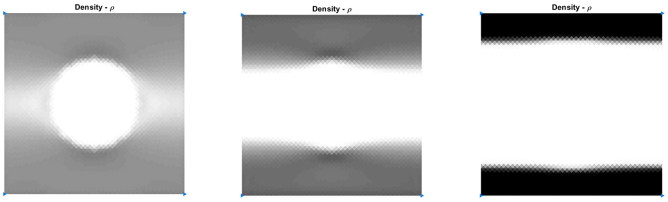
Results

In this chapter, the simulation results of the optimization processes will be presented. Firstly, the 2D simulations will be showcased, considering various values of α and β. Subsequently, the 3D simulations will be divided into three sections: the simulations conducted with the MMA and Null Space optimizers for normal materials, and the optimization of metamaterials using the Null Space optimizer. While this paper presents a selection of simulation results, additional results and animations for each simulation can be found in the associated Drive folder. [58]

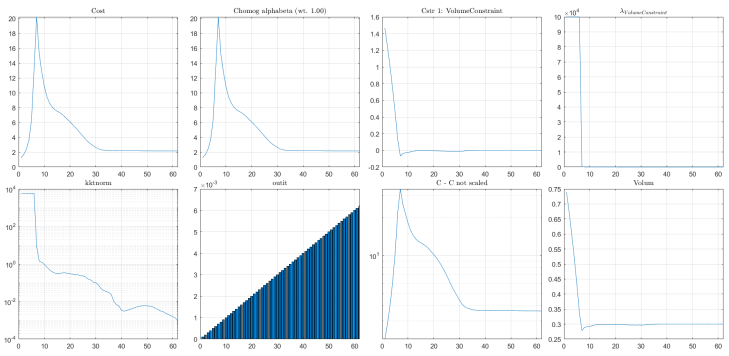
8.1 2D Simulations

In this section, the 2D simulations conducted with the MMA optimizer will be presented. The parameters utilized for these simulations are displayed in the Appendix G.1.

Case 1: $\alpha = \beta = [1\ 0\ 0]$



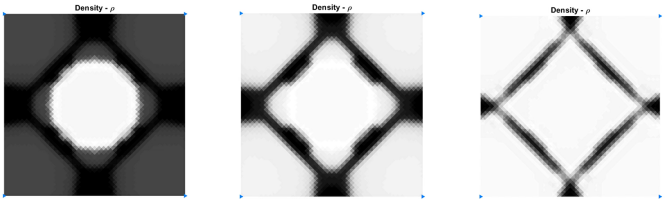
(a) Step 1. (b) Step 10. (c) Step 50.



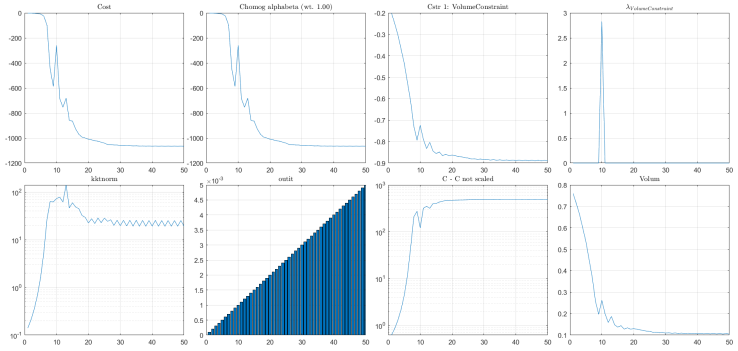
(d) Monitoring.

Figure 8.1: Density results from the 2D J_1 cost function for $\alpha = \beta = [1\ 0\ 0]$.

Case 2: α = [1 0 0] and β = [0 1 0]



(a) Step 1. (b) Step 15. (c) Step 45.

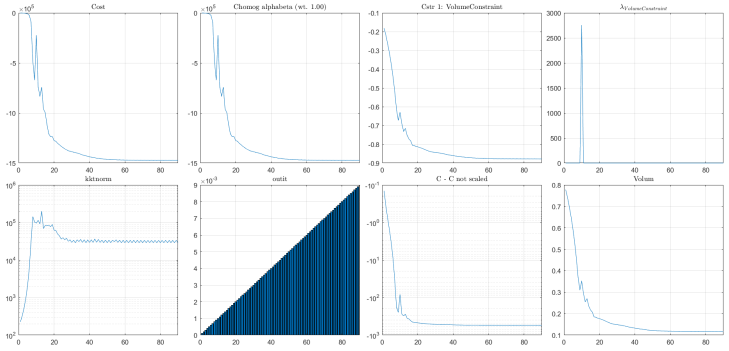


(d) Monitoring.

Figure 8.2: Density results from the 2D J_1 cost function for α = [1 0 0] and β = [0 1 0].

Case 3: α = [1 0 0] and β = [0 0 1]

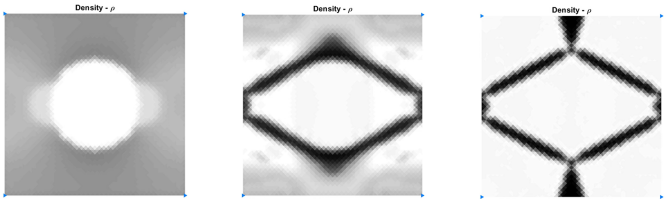
(a) Step 1. (b) Step 20. (c) Step 85.



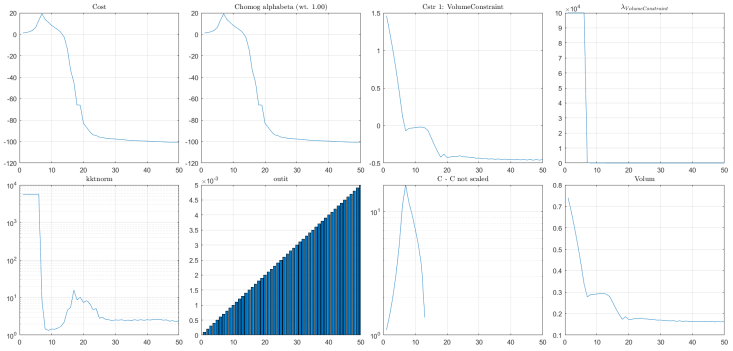
(d) Monitoring.

Figure 8.3: Density results from the 2D J_1 cost function for α = [1 0 0] and β = [0 0 1].

Case 4: α = [1 1 0] and β = [1 0 0]



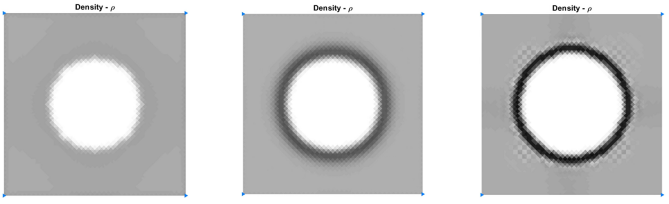
(a) Step 1. (b) Step 10. (c) Step 45.



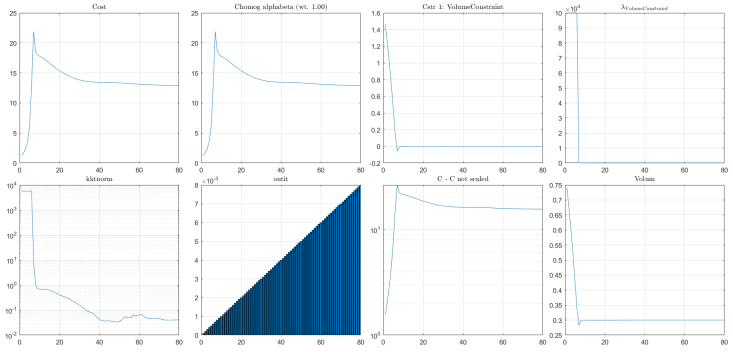
(d) Monitoring.

Figure 8.4: Density results from the 2D J_1 cost function for α = [1 1 0] and β = [1 0 0].

Case 5: $\alpha = \beta$ = [1 1 0]



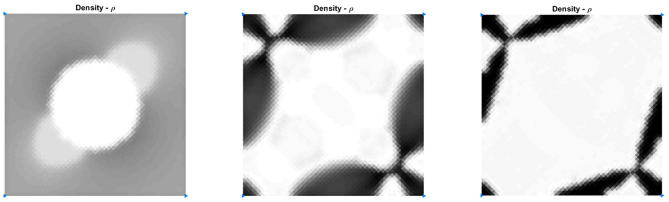
(a) Step 1. (b) Step 5. (c) Step 30.



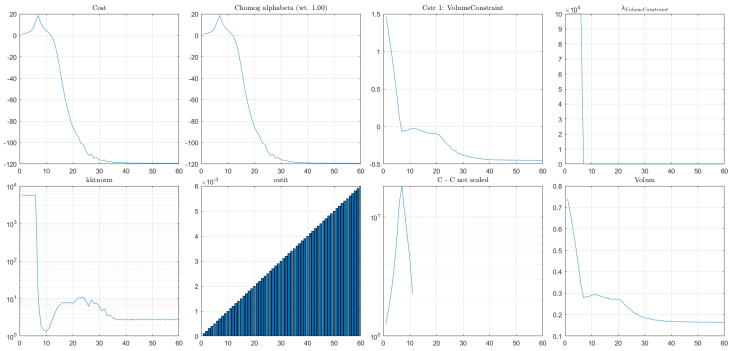
(d) Monitoring.

Figure 8.5: Density results from the 2D J_1 cost function for $\alpha = \beta$ = [1 1 0].

Case 6: α = [1 1 1] and β = [1 1 0]



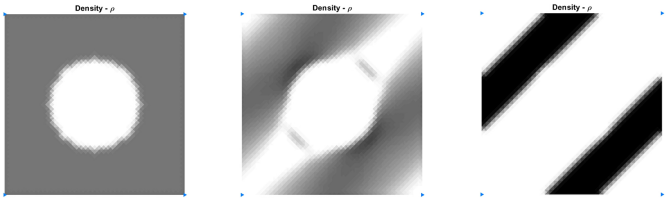
(a) Step 1. (b) Step 10. (c) Step 50.



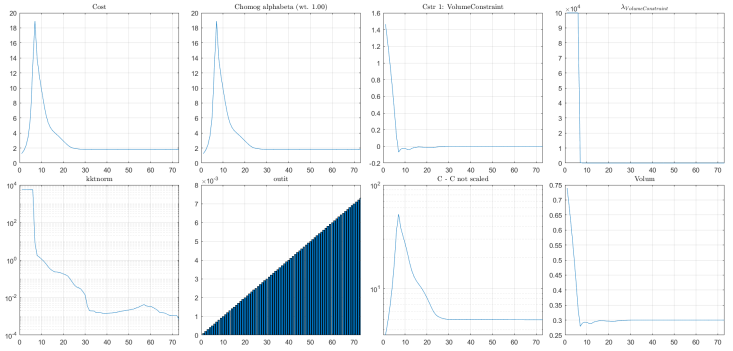
(d) Monitoring.

Figure 8.6: Density results from the 2D J_1 cost function for α = [1 1 1] and β = [1 1 0].

Case 7: $\alpha = \beta$ = [1 1 1]



(a) Step 1. (b) Step 5. (c) Step 30.



(d) Monitoring.

Figure 8.7: Density results from the 2D J_1 cost function for $\alpha = \beta$ = [1 1 1].

8.2 3D Simulations

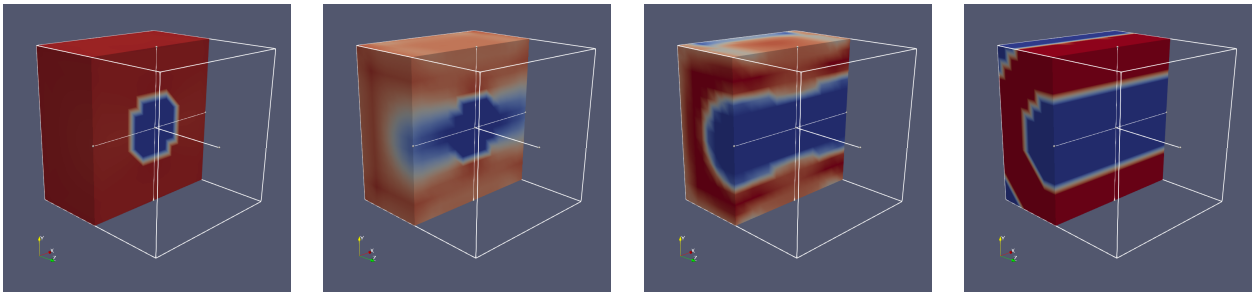
This section focuses on the 3D simulations conducted for different types of microstructures, and the parameters employed can be found in Appendix G.2.

Firstly, simulations of normal microstructures will be presented, utilizing both the MMA and Null Space optimizers with varying values of α and β to optimize the J_1 cost function, what allows for a comparison between the two types of optimizers. To provide an exhaustive analysis, the simulations will be performed with final volume fractions of 0.7 and 0.85. However, the results corresponding to $Vfinal = 0.85$ will be included in Appendix G.3.1 and Appendix G.3.2 (for the MMA and Null Space optimizers, respectively), while this section will focus on presenting the results for $Vfinal = 0.7$.

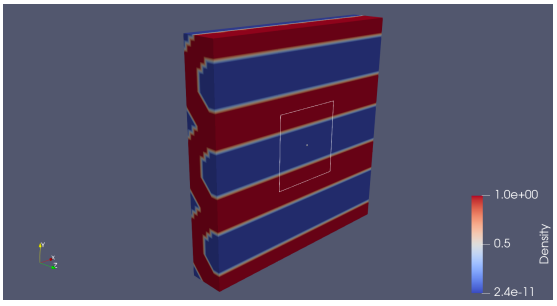
Following that, the simulations of metamaterial microstructures will be presented. These simulations were conducted using the Null Space optimizer, optimizing the J_1 function with different α and β values. To ensure a comprehensive analysis, the simulations were performed for different final volume fractions, specifically 0.7 and 0.85 too. However, adhering to the same structure as for the regular cases, this section will focus on the simulations with $Vfinal = 0.7$, while the simulations with $Vfinal = 0.85$ can be found in Appendix G.3.3.

8.2.1 Density-Based Method: MMA

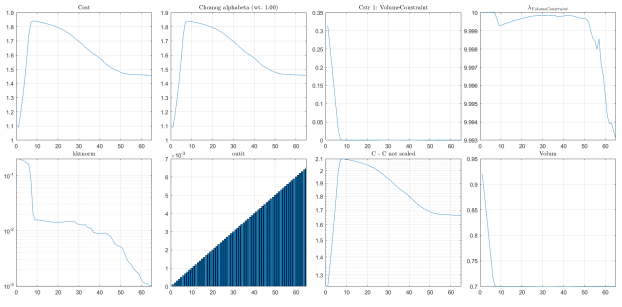
Cilinder: $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$



(a) Step 1. (b) Step 25. (c) Step 40. (d) Step 65.



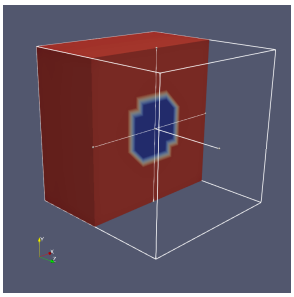
(e) Grid visualization.



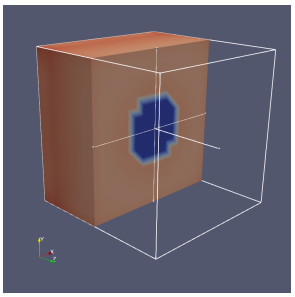
(f) Monitoring.

Figure 8.8: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.7$).

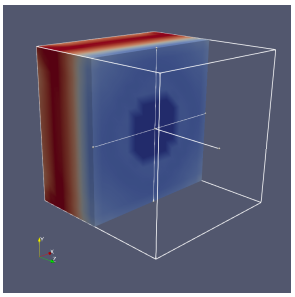
Vertical plate: $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$



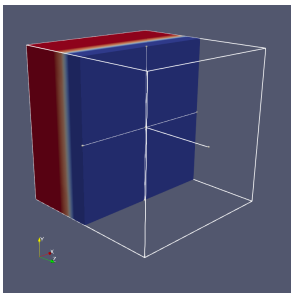
(a) Step 1.



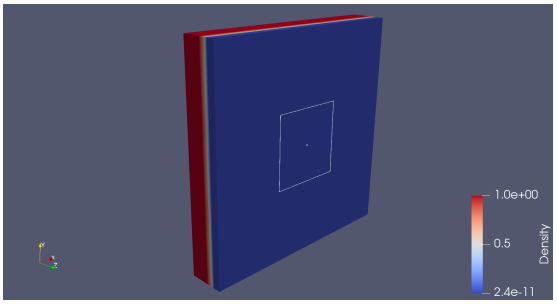
(b) Step 15.



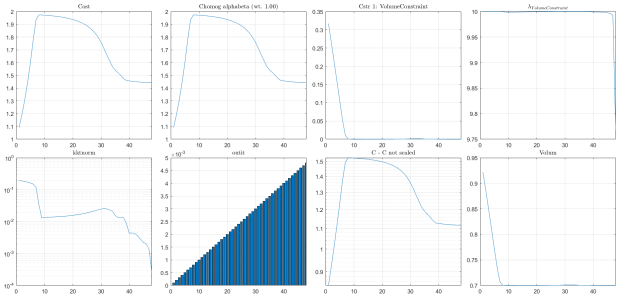
(c) Step 60.



(d) Step 47.



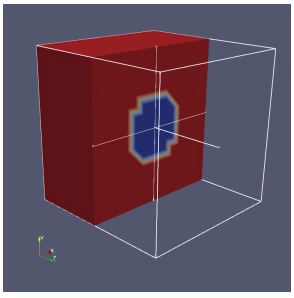
(e) Grid visualization.



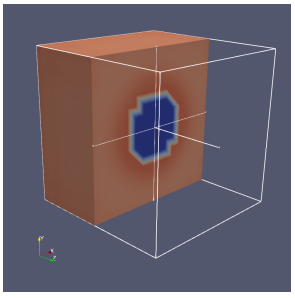
(f) Monitoring.

Figure 8.9: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.7$).

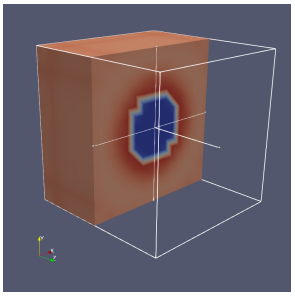
Sphere: $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$



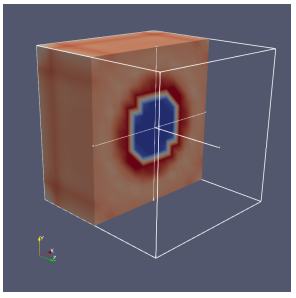
(a) Step 1.



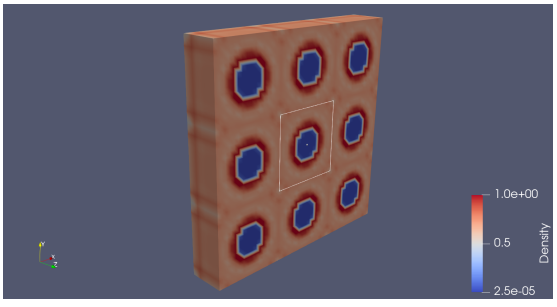
(b) Step 20.



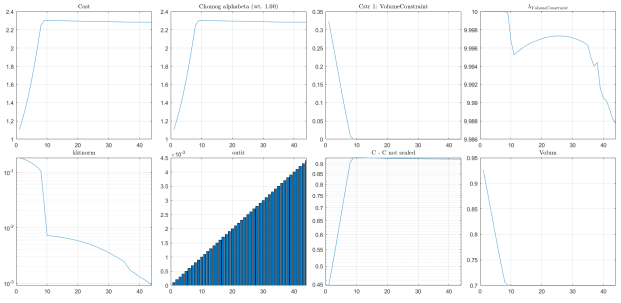
(c) Step 30.



(d) Step 44.



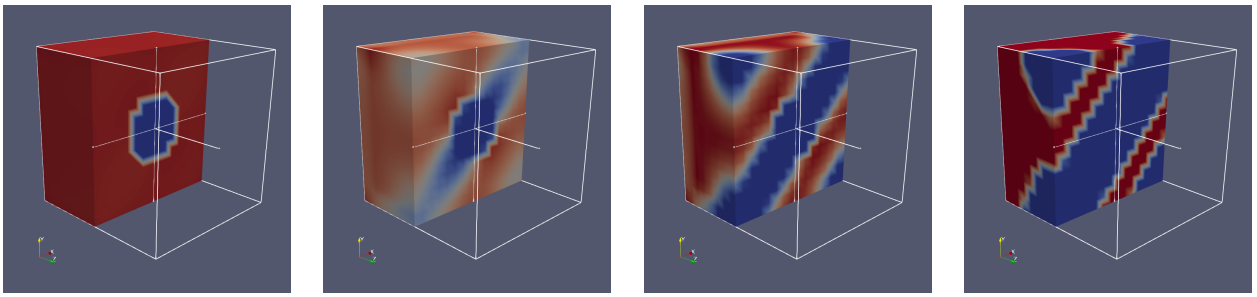
(e) Grid visualization.



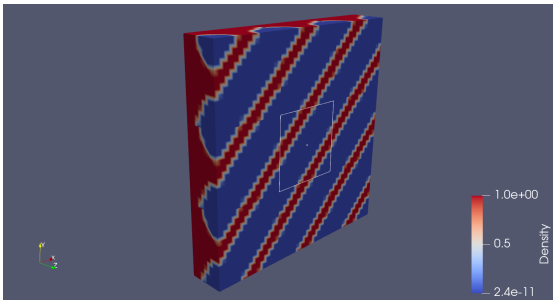
(f) Monitoring.

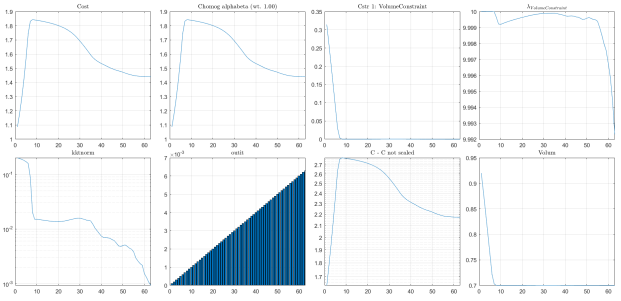
Figure 8.10: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.7$).

Diagonal: $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 15. (c) Step 60. (d) Step 47.

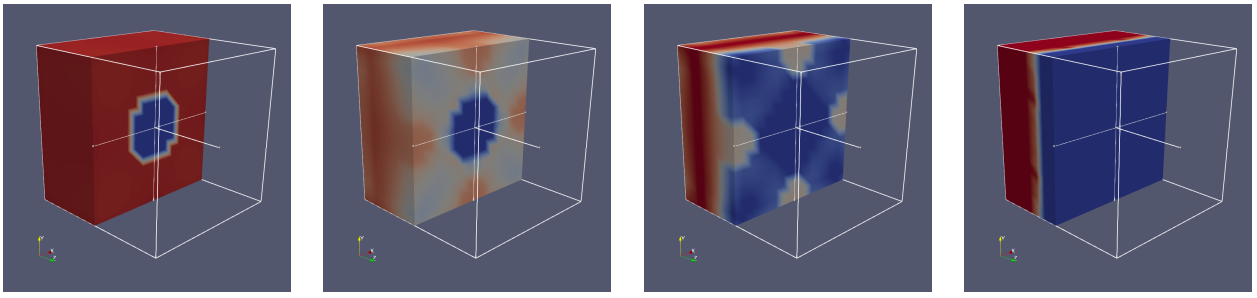




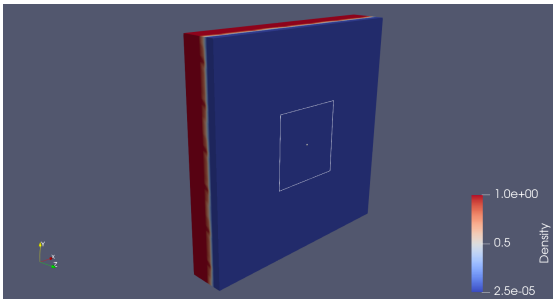
(e) Grid visualization. (f) Monitoring.

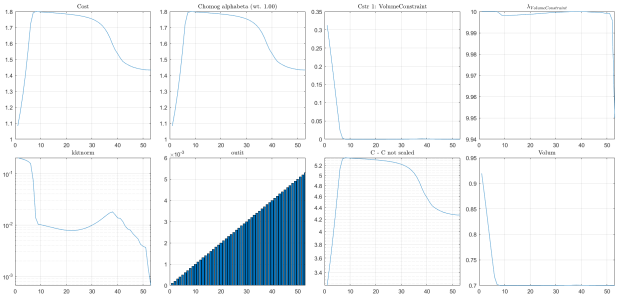
Figure 8.11: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$ (MMA - $Vfinal = 0.7$).

Shear: $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 30. (c) Step 40. (d) Step 53.



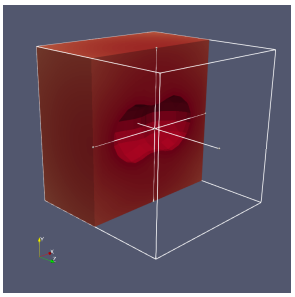


(e) Grid visualization. (f) Monitoring.

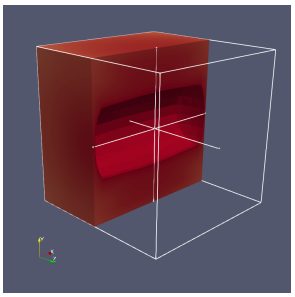
Figure 8.12: Density results from the 3D J_1 cost function for $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$ (MMA - $Vfinal = 0.7$).

8.2.2 Level Set Method: Null Space

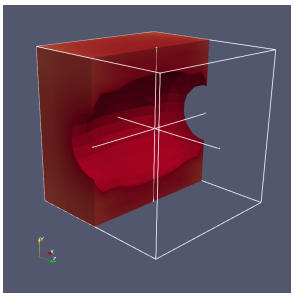
Cilinder: $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$



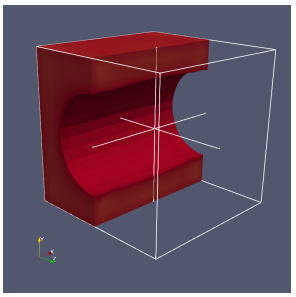
(a) Step 1.



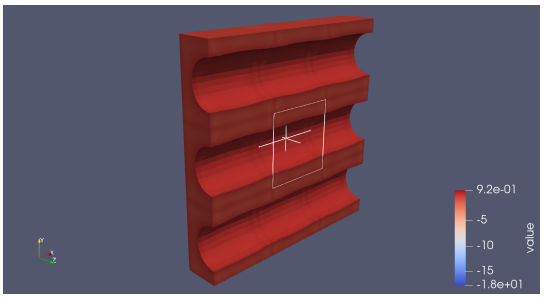
(b) Step 5.



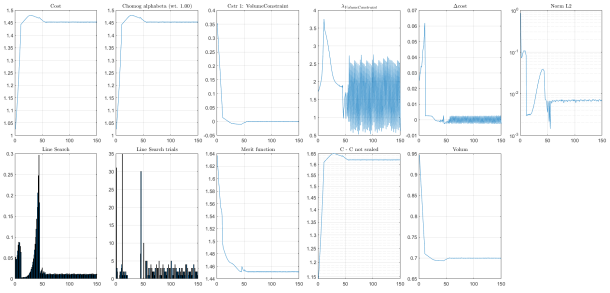
(c) Step 25.



(d) Step 70.



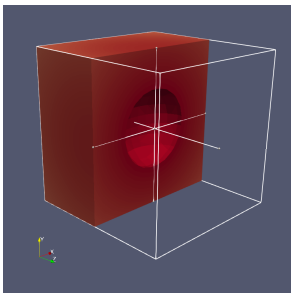
(e) Grid visualization.



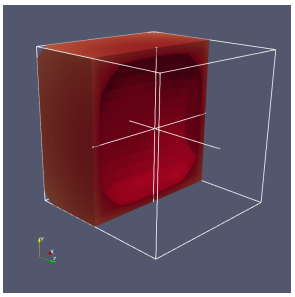
(f) Monitoring.

Figure 8.13: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.7$).

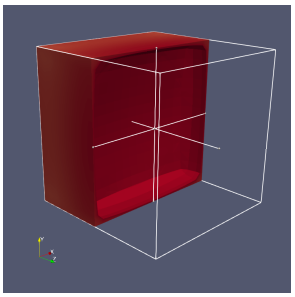
Vertical plate: $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$



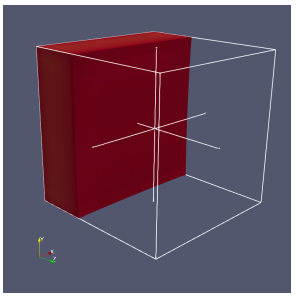
(a) Step 1.



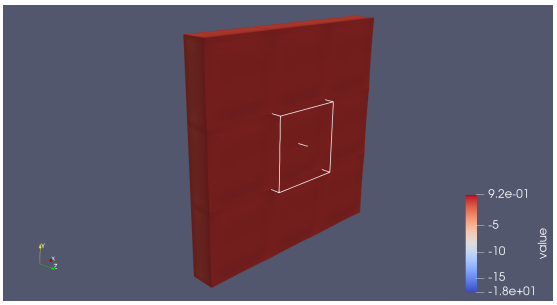
(b) Step 10.



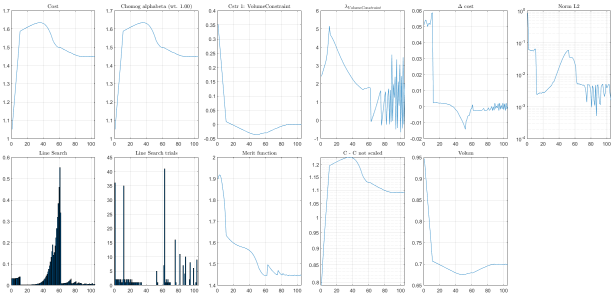
(c) Step 50.



(d) Step 70.



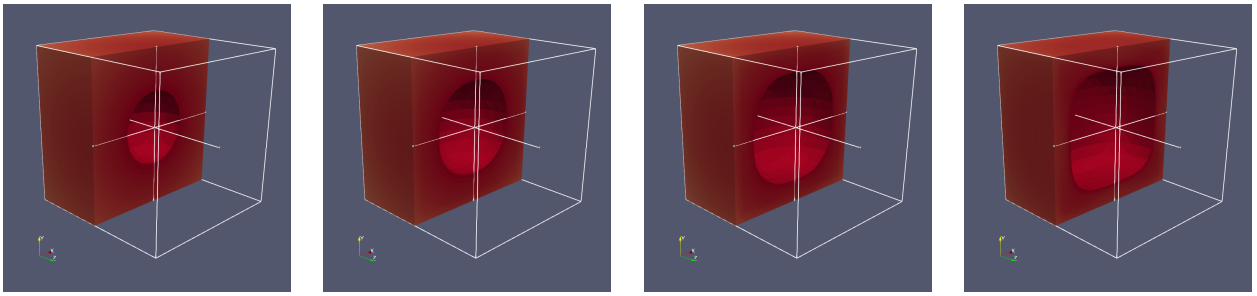
(e) Grid visualization.



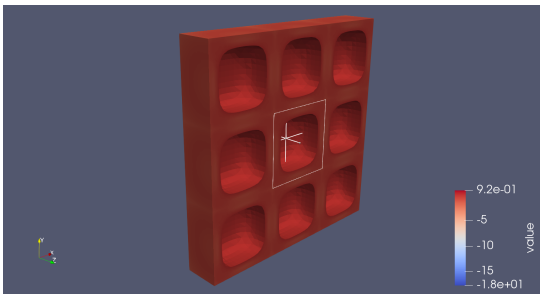
(f) Monitoring.

Figure 8.14: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.7$).

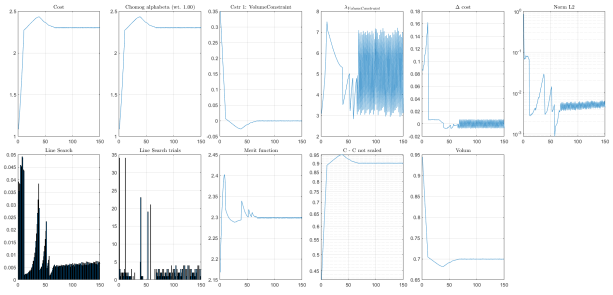
Sphere: $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$



(a) Step 1. (b) Step 5. (c) Step 25. (d) Step 70.



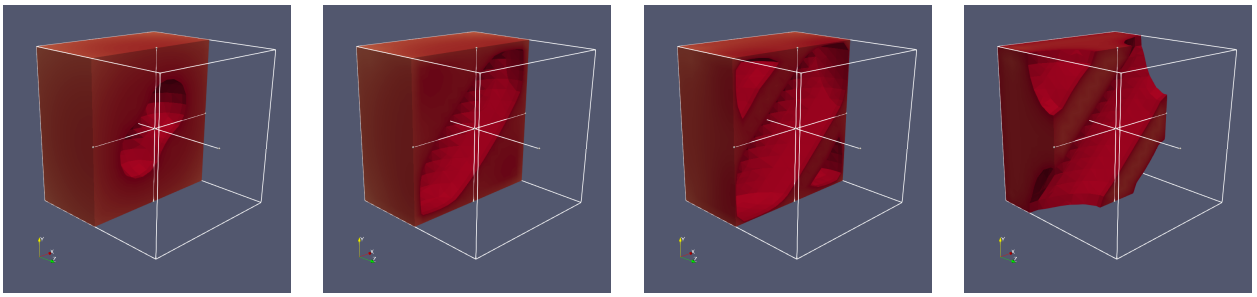
(e) Grid visualization.



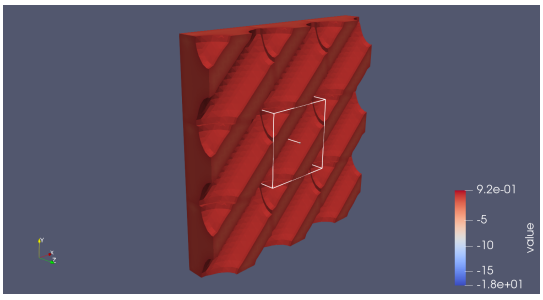
(f) Monitoring.

Figure 8.15: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.7$).

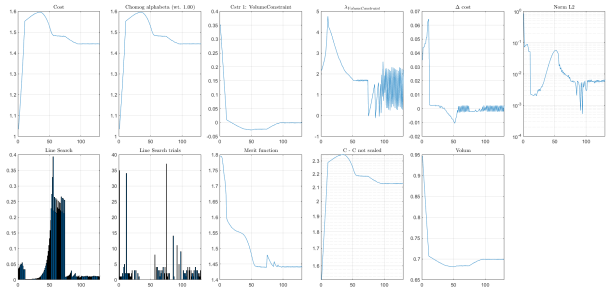
Diagonal: $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 10. (c) Step 25. (d) Step 100.



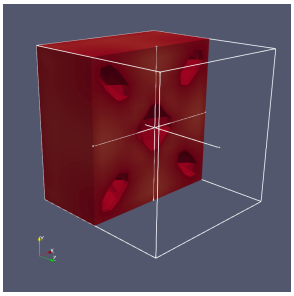
(e) Grid visualization.



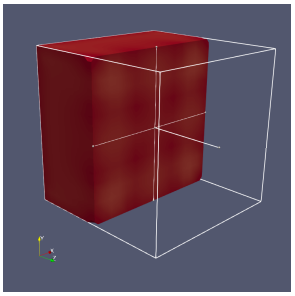
(f) Monitoring.

Figure 8.16: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$ (Null Space - $Vfinal = 0.7$).

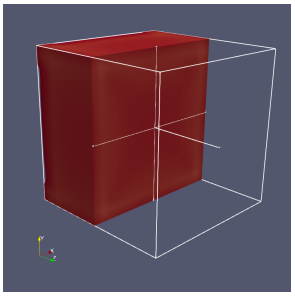
Shear: $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$



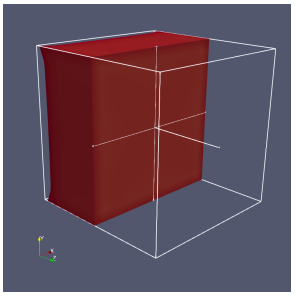
(a) Step 1.



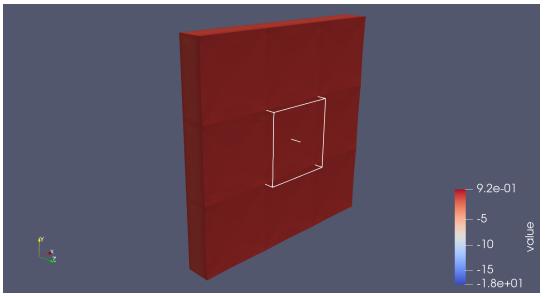
(b) Step 3.



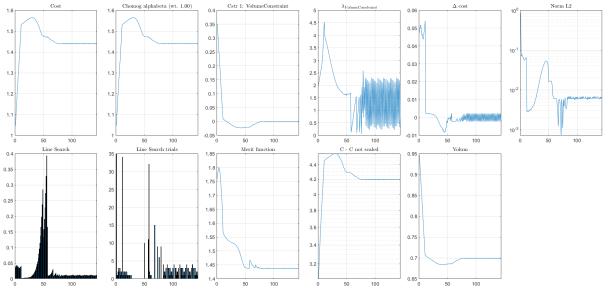
(c) Step 10.



(d) Step 50.



(e) Grid visualization.

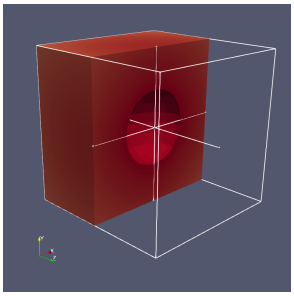


(f) Monitoring.

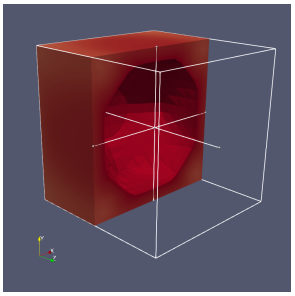
Figure 8.17: Density results from the 3D J_1 cost function for $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$ (Null Space - $Vfinal = 0.7$).

8.2.3 Metamaterials

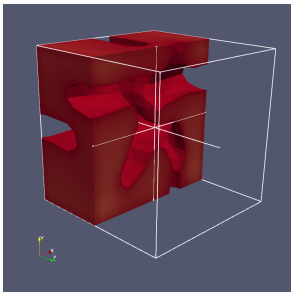
Case 1: $\alpha = [1\ 0\ 0\ 0\ 0\ 0]$ and $\beta = [0\ -1\ 0\ 0\ 0\ 0]$



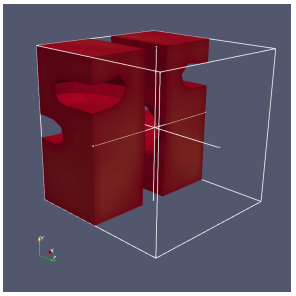
(a) Step 1.



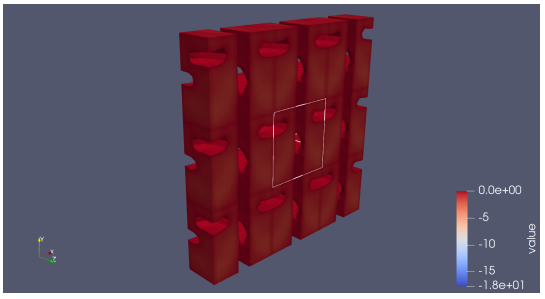
(b) Step 75.



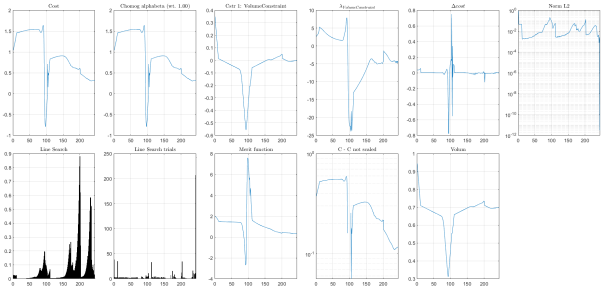
(c) Step 170.



(d) Step 240.



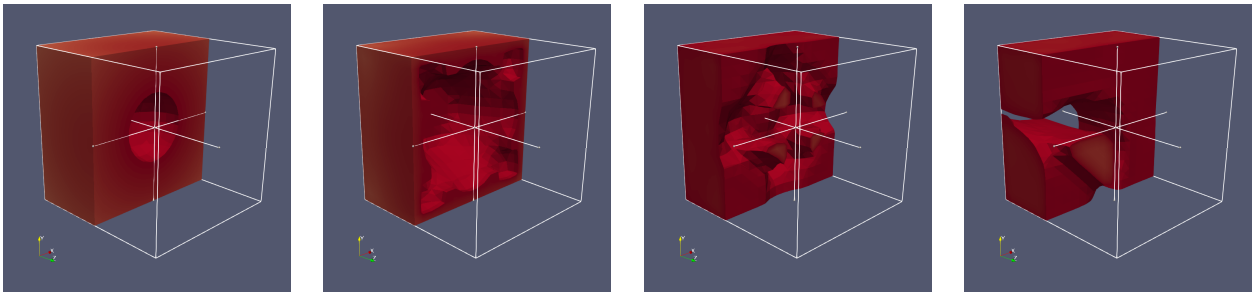
(e) Grid visualization.



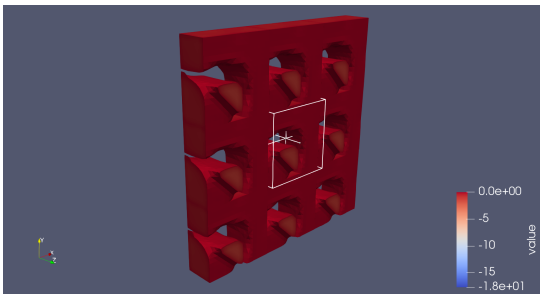
(f) Monitoring.

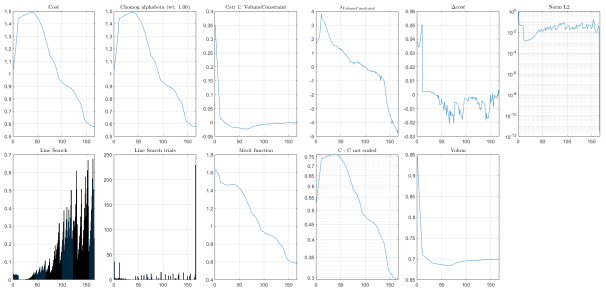
Figure 8.18: Density results from the 3D J_1 cost function for case 1 metamaterial simulation ($Vfinal = 0.7$).

Case 2: α = [1 1 0 0 0 0] and β = [0 0 −1 0 0 0]



(a) Step 1. (b) Step 50. (c) Step 100. (d) Step 167.

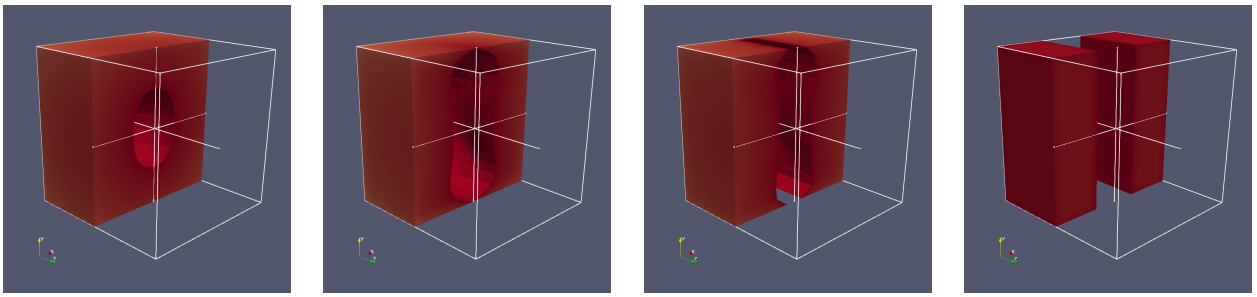




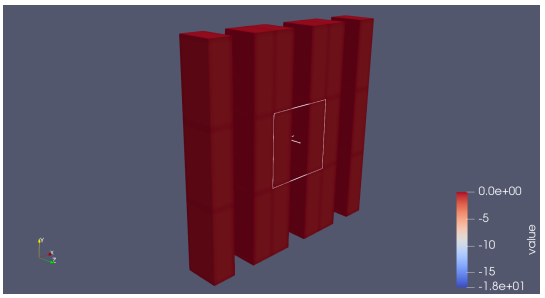
(e) Grid visualization. (f) Monitoring.

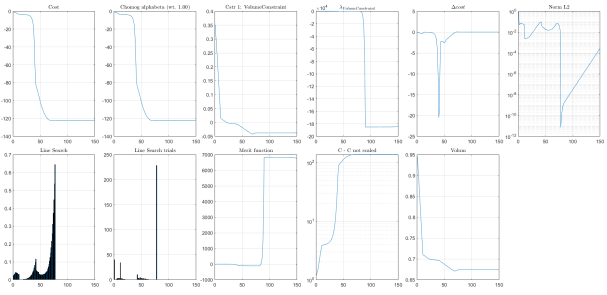
Figure 8.19: Density results from the 3D J_1 cost function for case 2 metamaterial simulation ($Vfinal = 0.7$).

Case 3: α = [1 0 0 0 0 0] and β = [−1 0 0 0 0 0]



(a) Step 1. (b) Step 20. (c) Step 40. (d) Step 90.





(e) Grid visualization. (f) Monitoring.

Figure 8.20: Density results from the 3D J_1 cost function for case 3 metamaterial simulation ($Vfinal = 0.7$).

8.3 Results Analysis

General Observations

Firstly, the focus will be on the 2D simulations, which exhibited expected behavior. For example, Figure 8.1 illustrates a scenario where the software was instructed to optimize the $\mathbb{C}_{11}$ component by setting $\alpha = \beta$ =[1 0 0]. In this case, the physical objective was to optimize the material's horizontal elasticity properties, specifically for handling horizontal loads and, as anticipated, the simulations successfully achieved this objective by forming two horizontal stripes.

Similar observations can be made regarding the 3D simulations. In Figure 8.8 and 8.13, the software was instructed to handle $\alpha = \beta$ =[1 0 0 0 0 0] for the MMA and Null Space optimizer, respectively, That physically signifies an axial force along the x-axis while maintaining symmetry in the y and z directions. As anticipated, this simulation yielded a cylindrical void within the mesh cubic structure.

In another scenario, illustrated in Figure 8.10 and 8.15, it was aimed to optimize $\alpha = \beta$ =[1 1 1 0 0 0] for both optimizers again. This directive aimed to enhance material behavior along the x, y, and z directions while disregarding the orthogonal directions. Consequently, the resulting simulation showcased a centralized sphere, aligning with the expected outcome.

Also, in general (although exceptions exist for complex final geometries), the 2D simulations demonstrated a faster convergence compared to their 3D counterparts. This outcome can be attributed to the fact that the 2D simulations involve a smaller number of optimization problems being solved, while the 3D simulations entail a larger set of optimization challenges, leading to a relatively longer convergence time. This manifests a logical relationship between the dimensionality of the simulation and the convergence speed.

Connection between 2D and 3D

Another noteworthy aspect to discuss is the relationship between the 2D and 3D simulations. As previously mentioned, the 2D simulation with $\alpha = \beta$ =[1 0 0] yielded two horizontal stripes, and transforming this simulation into a three-dimensional representation can be achieved through various approaches. One method involves continuing the optimization solely along the x-direction, replicating the 2D scenario as the 3D case with $\alpha = \beta$ =[1 0 0 0 0 0]. Consequently, the resulting 3D simulation would feature the rotated 2D plane, resembling a cylinder formed around its central axis (see Figure 8.8 and 8.13).

Alternatively, another approach involves optimizing a different principal direction while leaving the orthogonal directions unchanged. For instance, considering the case of $\alpha = \beta$ =[1 1 0 0 0 0], the y-direction would also be prioritized while disregarding the other orthogonal directions. In this scenario, the 2D result would be maintained across all values of y, leading to the transformation of the two 2D stripes into two distinct 3D plates (see Figure 8.9 and 8.14).

Cost Parameter

An essential factor to consider is the cost parameter present on the optimization process monitoring, which exhibits a consistent trend across various scenarios, including the 2D cases, 3D normal materials and 3D metamaterials.

In the simulations under study, the cost function was defined as the compliance of the structure, which is inversely proportional to its stiffness. In other words, the software's objective is to minimize the compliance to maximize the stiffness of the structure.

Typically, the cost initially increases as the software eliminates material from the cube, aiming to achieve the desired final volume fraction. Subsequently, the cost begins to decrease as the software optimizes the physical properties of the material. This observation carries significant importance, indicating that the solution is converging effectively. The diminishing cost between iterations indicates a reduction in the effort required to transition from one iteration to the next, reinforcing the progress made in reaching an optimal solution.

MMA vs. Null Space

It is interesting to conduct a comparison between the 3D simulations conducted using the MMA and Null Space optimizers. One noticeable observation is that the Null Space optimizer requires a significantly higher number of iterations, often double or more than those needed by the MMA, as illustrated in Table 8.1.

		OPTIMIZERS	
		MMA (Density)	**Null Space (Level Set)**
C	**Cilinder**	65	151
A	**Vertical Plate**	49	106
S	**Sphere**	44	152
E	**Diagonal**	63	137
S	**Shear**	53	148

Table 8.1: Number of iterations for the 3D MMA and Null Space simulations

However, it is important to emphasize that the increased number of iterations does not necessarily translate into a longer compilation time. This distinction arises due to the unique functioning of the Null Space optimizer. While it initially requires a substantial amount of time to yield acceptable results, once it obtains them, the simulation's performance rapidly accelerates.

Moreover, despite the distinct mathematical approaches employed by each optimizer, it is crucial to note that both methods ultimately yield identical simulation results. This outcome stems from the fact that, despite the different algorithms employed, both optimizers tackle the same underlying optimization problem.

Metamaterials

In the context of metamaterial simulations, it is crucial to highlight a few notorious aspects. Firstly, despite the relatively similar number of iterations compared to other 3D simulations, they require more than double the time to complete. This time discrepancy is attributed to the optimization process of the material, which operates in a manner that is not inherently intuitive, particularly due to the incorporation of negative Poisson values.

Secondly, it is intriguing to note the challenges encountered when calibrating the aJmax and aGmax parameters for the metamaterial simulations. The solver employed in the optimization process has a tendency to minimize the cost parameter. However, the presence of negative Poisson values can lead to a negative cost, resulting in the optimizer continuously attempting to further minimize it until reaching a final volume fraction of 0. To prevent this undesired outcome, the aGmax must be meticulously calibrated. By carefully adjusting this parameter, the solution to the problem adheres to the imposed volume constraint, ensuring a more suitable optimization result.

Lastly, it is worth mentioning the significant impact of the final volume fraction on the results of metamaterial simulations. As evidenced by Figure 8.18 and Figure G.11, this parameter has a significant impact on the resulting mesh geometry. Such dependency establishes a notable distinction compared to simulations involving 3D normal materials, where it didn't affect the simulation outcome in terms of its shape.

Analogy with Composite Materials' Structure

Lastly, it is worth noting another significant outcome that establishes a meaningful connection between the achieved results and the actual configuration of composite materials. Composite materials encompass a range of structural forms, with laminated and fiber-reinforced composites being the most prevalent. Laminated composites are specifically engineered to exhibit superior resistance to tensile stresses within the plane of the laminate, while fiber-reinforced composites optimize tensile strength along the axial direction of the fibers.

Remarkably, the 3D simulations conducted in the cases of the vertical plate and cylindrical configurations precisely manifested these expected characteristics, as the software was specifically instructed to prioritize the optimization of the xy plane and the x direction, respectively. This correspondence between the simulation results and the structural design of composite materials not only validates the accuracy of the code but also provides valuable insights into the rationale behind the specific arrangement of composite materials.

8.4 Challenges and Alternative Solutions

During the course of this project, several challenges were encountered that required attention. To overcome these challenges, various alternatives were proposed and carefully evaluated to find the optimal solutions for each problem. In this section, the key difficulties encountered will be highlighted and the alternatives that were pursued to address them effectively will be explained.

8.4.1 Mesh Refinement

One of the main challenges encountered was related to mesh refinement. The need for highly accurate results required a mesh composed of a large number of elements, which, in turn, gave rise to several issues:

- **GID.** First, due to limitations imposed by the employed license on the GID software, the creation of meshes was restricted to a maximum of 10,000 nodes. To address this issue, a study was conducted to determine the maximum number of divisions per line and the minimum size of the elements that could be created.
- **MATLAB Java heap memory.** Subsequently, after successfully creating the mesh, another issue arose during the simulation with the Null Space optimizer in MATLAB. An error message indicated that Java objects had exceeded the available memory, resulting in an out-of-memory error. To resolve this, the heap size allocated for Java objects had to be increased to its maximum capacity (1.958 MB). However, even with this adjustment, the issue persisted. As a result, the mesh had to be recreated with a reduced number of elements to alleviate the memory demands and ensure smooth execution of the simulation.
- **MATLAB array size.** Finally, after successfully managing the Java objects, another error occurred in MATLAB indicating that the arrays being processed exceeded the maximum array size limit (which was 7.7 GB). As a result, the program might become unresponsive. To address this issue, the mesh had to be reconstructed once again to ensure it could be properly handled within the limitations of MATLAB.

After careful analysis, it was determined that the optimal values were 20 divisions per line and an element size of 0.0866025. These findings are detailed in Section 7.3.1 of the document.

8.4.2 Mesh Element Type

Another issue that needed to be addressed was the choice of element type in the mesh. Initially, the mesh consisted of tetrahedral elements, which were the default element type. However, these elements resulted in a slightly asymmetric mesh, leading to non-symmetric simulation results. This was problematic as the simulation aimed to eliminate material in a preferred direction. To resolve this issue, the mesh had to be redesigned using hexahedral elements, which ensured full symmetry and eliminated the asymmetry-related problems in the results.

However, changing the element type required code adaptation. The micro problem was originally programmed to handle only one Gaussian point (as required by tetrahedral elements), but with hexahedral elements, it needed to accommodate four Gaussian points. As a result, certain functions had to be modified to support the new element type. An example of this adaptation can be found in Section 7.3.4.

8.4.3 Simulation Performance

The simulation performance posed another challenge, as certain issues were observed during the monitoring process. The most common problems encountered were:

- The optimizer erroneously added material instead of removing it, resulting in the formation of a completely filled cube.
- The final volume did not meet the specified requirements.
- The volume constraint was not being satisfied.
- The obtained lambda value deviated from the expected order.

To address these challenges, the parameters of each optimizer had to be carefully adjusted based on the specific case. Further details regarding the modified variables and their respective solutions can be found in the preceding Section 7.3.6.

Chapter 9

Conclusions

9.1 Discussion

In this study, a comprehensive analysis of the simulation results was conducted, leading to several notable conclusions. Through an examination of the obtained data and observations made during the simulations, the following key findings were identified:

- The simulations in the 2D domain exhibited expected behavior and demonstrated a faster convergence compared to their 3D counterparts, proving a relationship between the dimensionality of the simulations and its convergence speed.
- The cost parameter consistently exhibited a specific trend across different scenarios, indicating effective convergence and progress towards achieving an optimal solution. The diminishing cost between iterations suggests a reduction in the effort required to transition from one iteration to the next.
- A comparison between the MMA and Null Space optimizers revealed that the latter required a significantly higher number of iterations, although it did not necessarily translate into longer compilation times. Both optimizers ultimately yielded identical simulation results, despite utilizing different mathematical approaches.
- Simulations involving metamaterials presented unique challenges, including longer simulation times and the need for meticulous calibration of parameters to avoid undesired outcomes. The final volume fraction significantly influenced the resulting mesh geometry, distinguishing it from simulations involving 3D normal materials.
- The observed outcomes in the simulations established a meaningful connection between the achieved results and the structural design of composite materials. The correspondence between the simulation results and the specific arrangements of composite materials validated the accuracy of the employed code and provided insights into their underlying rationale.

In summary, this research not only validated standard solutions but also introduced new geometries. The simulations performed, along with the accompanying analysis, have significantly contributed to the understanding of the optimization

process and the generation of innovative configurations. These findings have broad implications for various fields, including materials science and engineering, where the development of advanced and tailored structures holds significant importance.

9.2 Future Development

Several potential areas for further investigation and improvements arise from this study:

- **Exploration of 3D Meshes.** Creating 3D meshes with different element types would be valuable to assess the adaptability of the code to diverse mesh structures. This investigation would provide insights into the code's versatility and its ability to handle varying mesh configurations effectively.
- **Mesh-Independence Study.** Conducting a thorough mesh-independence study is crucial to assess the influence of mesh resolution on simulation results. By systematically varying the mesh density and analyzing the corresponding outputs, the level of mesh refinement required for accurate and reliable simulations can be determined.
- **Comparative Analysis of Additional Optimizers.** Simulating the optimization process using various supplementary optimization algorithms would allow for a comprehensive comparison of their performance. By analyzing the results obtained from different optimizers, it can be procured the insights into the strengths and weaknesses of each approach and the most suitable optimizer for specific scenarios can be specified.
- **Development of J_3 Function for 3D Cases.** Extending the existing optimization framework to incorporate the J_3 function specifically designed for 3D cases would enable a broader range of cost functions to be utilized. This expansion would provide additional tools to tailor the optimization process to specific requirements.
- **Simulation of metamaterial cases for smaller final volume fractions**. Expanding the analysis to include lower final volume fractions (such as 0.5 or 0.3) can provide insights into the behavior and optimization outcomes of metamaterial microstructures at reduced densities. This investigation would explore the impact of decreasing the volume fraction on the resulting microstructures and their performance, which would be very useful taking into account the applications of these materials.
- **Iterative Solver.** The development and implementation of a new iterative solver aims to reduce the computing time of the topology optimization problem, which is a crucial concern due to the complexity and scale of the simulations involved.

References

[1] Martin P. Bendsøe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. [Last visited: 24/03/2022]. 2003. URL: https://vdoc.pub/documents/topology-optimization-theory-methods-and-applications-3eun5i6gatdg.

[2] G. I. N. Rozvany. "A critical review of established methods of structural topology optimization". In: *Structural and Multidisciplinary Optimization* 37.3 (2009), pp. 217–237. ISSN: 1615-1488.

[3] Jikai Liu and Yongsheng Ma. "A survey of manufacturing oriented topology optimization methods". In: *Advances in Engineering Software* (2016). [Last visited: 24/02/2022]. URL: https://www.sciencedirect.com/science/article/abs/pii/S0965997816302174.

[4] S.B. Guenneau et al. "Acoustic metamaterials for sound focusing and confinement". In: *New Journal of Physics* (2007).

[5] M. Brun, S. Guenneau, and A.B. Movchan. "Achieving control of in-plane elastic waves". In: *Applied Physics Letters* (2009).

[6] Tiantian He et al. "Guided mode meta-optics: metasurface-dressed waveguides for arbitrary mode couplers and on-chip OAM emitters with a configurable topological charge". In: *Optics Express* (2021).

[7] J. Berger and H. Wadley. "Mechanical metamaterials at the theoretical limit of isotropic elastic stiffness". In: *Nature* (2017).

[8] Sattar Mohammadi Esfarjani, Ali Dadashi, and Mohammad Azadi. "Topology optimization of additive-manufactured metamaterial structures: A review focused on multi-material types". In: *Forces in Mechanics* 7 (2022). [Last visited: 24/02/2022]. ISSN: 2666-3597. URL: https://www.sciencedirect.com/science/article/pii/S2666359722000300.

[9] Martin P Bendsøe and Ole Sigmund. *Topology optimization: theory, methods and applications*. Springer Science & Business Media, 2003.

[10] Martin P Bendsøe and Noboru Kikuchi. "Optimal shape design as a material distribution problem". In: *Structural Optimization* 1.4 (1989), pp. 193–202.

[11] Engineering Product Design. *Introduction to Topology Optimization*. Last accessed: 19/04/2023. 2023. URL: https://engineeringproductdesign.com/knowledge-base/topology-optimization/.

[12] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer Science & Business Media, 2006.

[13] Katia Bertoldi et al. "Flexible mechanical metamaterials". In: *Nature Reviews Materials* 2.11 (2017), pp. 1–11.

[14] Christa P. de Jonge, Helena M. A. Kolken, and Amir A. Zadpoor. "Non-Auxetic Mechanical Metamaterials". In: *Materials* 12.4 (2019). ISSN: 1996-1944. URL: https://www.mdpi.com/1996-1944/12/4/635.

[15] Lorenzo Valdevit et al. "Optimal lattice-structured materials". In: *Journal of Materials Research* 29.1 (2014), pp. 152–161.

[16] Roderic Lakes. "Materials with structural hierarchy". In: *Nature* 361.6412 (1993), pp. 511–515.

[17] Jesse L Silverberg et al. "Using origami design principles to fold reprogrammable mechanical metamaterials". In: *Science* 345.6197 (2015), pp. 647–650.

[18] Ahmad Rafsanjani et al. "Kirigami skins make a simple soft actuator crawl". In: *Science Robotics* 3.15 (2018), eaar7555.

[19] Ian Gibson, David W Rosen, and Brent Stucker. *Additive manufacturing technologies: rapid prototyping to direct digital manufacturing*. Springer, 2010.

[20] Marc J Madou. *Fundamentals of microfabrication: the science of miniaturization*. CRC press, 2002.

[21] Cat McClintock. *Why Is Additive Manufacturing Important?* Last accessed: 19/04/2023. 2019. URL: https://www.ptc.com/en/blogs/cad/why-additive-manufacturing-important.

[22] Carmel Majidi. "Soft Robotics: A Perspective—Current Trends and Prospects for the Future". In: *Soft Robotics* 1.1 (2017), pp. 5–11.

[23] Jordan S Miller, Can Liu, and Katia Bertoldi. "Biomedical applications of mechanical metamaterials". In: *Nature Reviews Materials* 4.7 (2019), pp. 441–456.

[24] Guancong Ma and Ping Sheng. "Acoustic metamaterials: From local resonances to broad horizons". In: *Science Advances* 2.2 (2016), e1501595.

[25] Ton Creus Costa. "Study of numerical methods for the design of large lightweight structures". In: *UPC Commons* (2022).

[26] Jing Zheng, Hao Li, and Chao Jiang. "Robust topology optimization for cellular composites with hybrid uncertainties". In: *International Journal for Numerical Methods in Engineering* 115 (May 2017).

[27] Raghavendra Sivapuram, Peter D. Dunning, and H. Alicia Kim. "Simultaneous material and structural optimization by multiscale topology optimization". In: *Structural and Multidisciplinary Optimization* 54.5 (Nov. 1, 2016), pp. 1267–1281. ISSN: 1615-1488. URL: https://doi.org/10.1007/s00158-016-1519-x.

[28] Tristan Djourachkovitch et al. "Multiscale topology optimization of 3D structures: A micro-architectured materials database assisted strategy". In: *Computers & Structures* 255 (2021), p. 106574. ISSN: 0045-7949.

[29] Àlex Ferrer Ferré. "Multi-scale topological design of structural materials: an integrated approach". In: (2017).

[30] Erik Andreassen and Casper Schousboe Andreasen. "How to determine composite material properties using numerical homogenization". In: *Computational Materials Science* 83 (2014), pp. 488–495. ISSN: 0927-0256.

[31] "Topology optimization for microstructural design under stress constraints". In: *Structural and Multidisciplinary Optimization* 58.6 (2018), pp. 2677–2695.

[32] Daniel Yago et al. "Topology Optimization Methods for 3D Structural Problems: A Comparative Study". In: *Archives of Computational Methods in Engineering* 29.3 (May 1, 2022), pp. 1525–1567. ISSN: 1886-1784.

[33] Matteo Bruggi and Pierre Duysinx. "Topology optimization for minimum weight with compliance and stress constraints". In: *Structural and Multidisciplinary Optimization* 46 (2012), pp. 369–384.

[34] James K Guest, Jean H Prévost, and Ted Belytschko. "Achieving minimum length scale in topology optimization using nodal design variables and projection functions". In: *International journal for numerical methods in engineering* 61.2 (2004), pp. 238–254.

[35] Miche Jansen. "Explicit level set and density methods for topology optimization with equivalent minimum length scale constraints". In: *Structural and Multidisciplinary Optimization* 59.5 (May 1, 2019), pp. 1775–1788. ISSN: 1615-1488.

[36] Joe Alexandersen. "A detailed introduction to density-based topology optimisation of fluid flow problems with implementation in MATLAB". In: *Structural and Multidisciplinary Optimization* 66.1 (Dec. 27, 2022), p. 12. ISSN: 1615-1488.

[37] George IN Rozvany. "Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics". In: *Structural and Multidisciplinary optimization* 21 (2001), pp. 90–108.

[38] Mathias Stolpe and Krister Svanberg. "On the trajectories of penalization methods for topology optimization". In: *Structural and Multidisciplinary Optimization* 21 (2001), pp. 128–139.

[39] Dmitri Tcherniak. "Topology optimization of resonating structures using SIMP method". In: *International Journal for Numerical Methods in Engineering* 54.11 (2002), pp. 1605–1622.

[40] A. Ferrer. "SIMP-ALL: A generalized SIMP method based on the topological derivative concept". In: *International Journal for Numerical Methods in Engineering* 120.3 (2019), pp. 361–381.

[41] Zhen Luo et al. "Compliant mechanism design using multi-objective topology optimization scheme of continuum structures". In: *Structural and Multidisciplinary Optimization* 30 (2005), pp. 142–154.

[42] *Material Interpolation*. Last accessed: 04/05/2023. 2018. URL: https://docs.software.vt.edu/abaqusv2022/English/TsoUserMap/tso-c-user-TopOpt-Sett-Sensi-MatInt.htm.

[43] N. P. van Dijk et al. "A level-set based topology optimization using the element connectivity parameterization method". In: *Structural and Multidisciplinary Optimization* 42.2 (Aug. 1, 2010), pp. 269–282. ISSN: 1615-1488. DOI: 10.1007/s00158-010-0485-y.

[44] N. P. van Dijk et al. "Level-set methods for structural topology optimization: a review". In: *Structural and Multidisciplinary Optimization* 48.3 (Sept. 1, 2013), pp. 437–472. ISSN: 1615-1488.

[45] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. "A level set method for structural topology optimization". In: *Computer methods in applied mechanics and engineering* 192.1-2 (2003), pp. 227–246.

[46] Joshua D Deaton and Ramana V Grandhi. "A survey of structural and multidisciplinary continuum topology optimization: post 2000". In: *Structural and Multidisciplinary Optimization* 49 (2014), pp. 1–38.

[47] Thuan Ho-Nguyen-Tan and Hyun-Gyu Kim. "Level set-based topology optimization for compliance and stress minimization of shell structures using trimmed quadrilateral shell meshes". In: *Computers & Structures* 259 (2022), p. 106695. ISSN: 0045-7949.

[48] Samuel Amstutz and Heiko Andrä. "A new algorithm for topology optimization using a level-set method". In: *Journal of computational physics* 216.2 (2006), pp. 573–588.

[49] S Amstutz et al. "Topological derivative for multi-scale linear elasticity models applied to the synthesis of microstructures". In: *International Journal for Numerical Methods in Engineering* 84.6 (2010), pp. 733–756.

[50] Ole Sigmund. "Materials with prescribed constitutive parameters: An inverse homogenization problem". In: *International Journal of Solids and Structures* 31.17 (1994), pp. 2313–2329. ISSN: 0020-7683.

[51] Nestor Rossi et al. "A Microarchitecture Design Methodology to Achieve Extreme Isotropic Elastic Properties of Composites Based on Crystal Symmetries". In: *Structural and Multidisciplinary Optimization* (May 2021).

[52] P.G. Coelho et al. "Scale-size effects analysis of optimal periodic material microstructures designed by the inverse homogenization method". In: *Computers & Structures* 174 (2016). CIVIL-COMP, pp. 21–32. ISSN: 0045-7949.

[53] Ferran De la Fuente. "Study of 3D printing in optimal design of structures and materials". MA thesis. Universitat Politècnica de Catalunya, 2017.

[54] github. *GitHub*. 2023. URL: https://github.com/.

[55] freeCodeCamp. *OOP Meaning – What is Object-Oriented Programming?* 2022. URL: https://www.freecodecamp.org/news/what-is-object-oriented-programming/.

[56] Smart Draw. *UML Diagram*. 2023. URL: https://www.smartdraw.com/uml-diagram/#whatIsUML.

[57] Ferrer et al. *Swan - Topology Optimization Laboratory*. https://https://github.com/SwanLab/Swan. 2023.

[58] Ariadna Sorribas Bono. *Google Drive - Simulation Results*. 2023. URL: https://drive.google.com/drive/folders/1sXrpuYpx7_YZXBkDvMuzVFtISpAzkWjj?usp=sharing.

[59] Daryl L Logan. *First Course in the Finite Element Method, Enhanced Edition, SI Version*. Cengage Learning, 2022.

[60] Joaquín Hernández Ortega. *Enginyeria Aeroespacial Computacional, Assignment 1*. UPC, 2022.

[61] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

Appendix A

Tasks Calendar

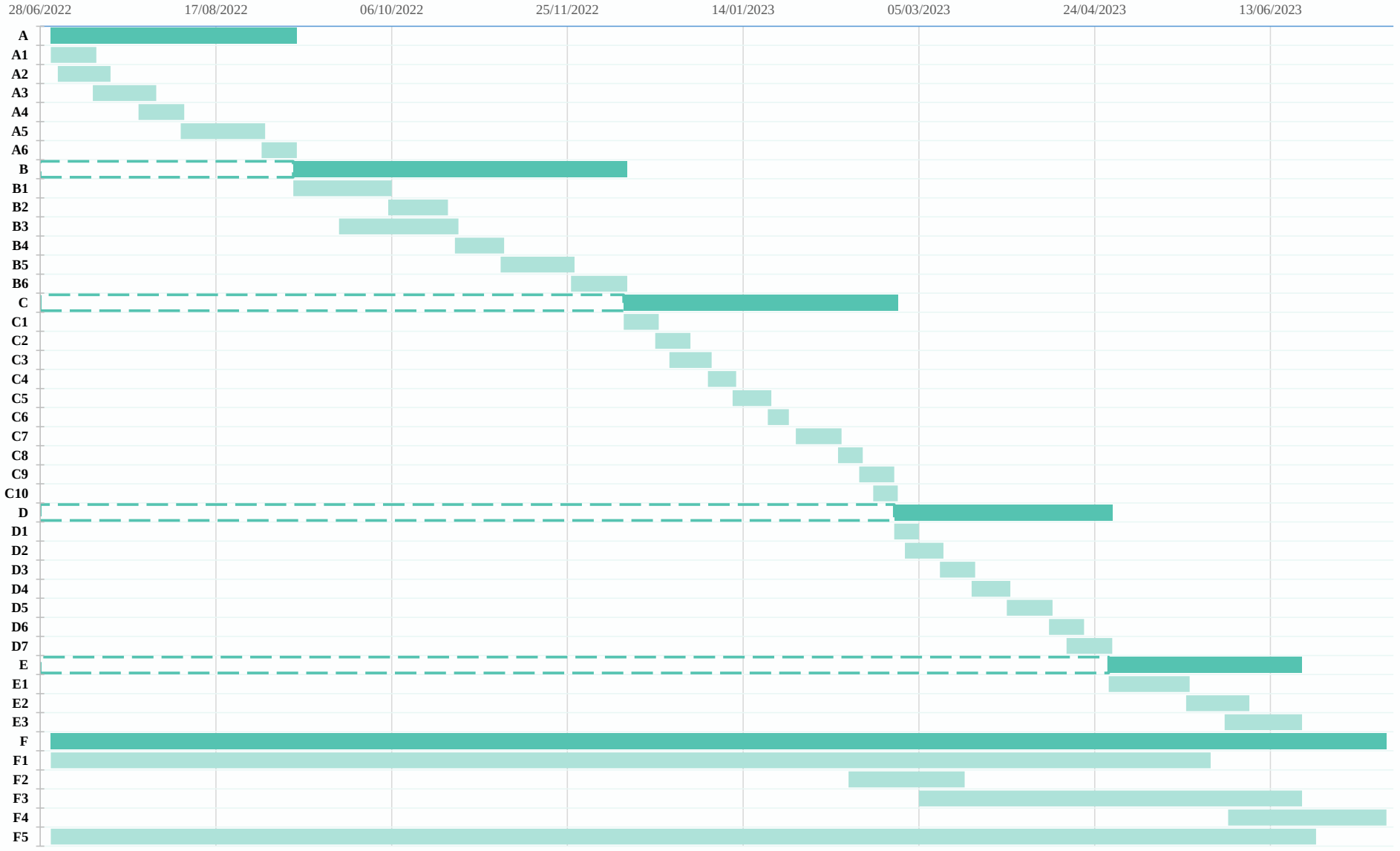
Task code	Name	Description	Preceding Tasks
A	**Clean Code Introduction**		
A1	Object-oriented programming	Research about object-oriented programming in MATLAB (classes, composition, inheritance, types of data access...)	-
A2	Clean coding practices	Learn about the clean code practices in MATLAB	-
A3	Refactor FEM code	Modify an existing FEM code implementing OOP (solver class) and clean code practices	A1, A2
A4	Test-driven development	Internet research about unit testing, creation of different tests for the FEM code and run a code coverage analysis	-
A5	Final FEM code refactoring	Refactor the whole FEM code and the tests so they follow an OOP approach while applying clean code practices	A3, A4
A6	UML diagram	Acquire knowledge on how to represent graphically any code, get introduced to UML modeling and illustrate the OOP code with an UML diagram	A5
B	**Finite Element Method Code in Swan Repository**		
B1	FEM Macro - Analysis	Understanding of the FEM Macro code and how it works	A6
B2	FEM Macro - UML	Illustration of the relation between FEM Macro classes using an UML diagram	B1
B3	FEM Micro - Analysis	Understanding of the FEM Micro code and how it works	B1
B4	FEM Micro - UML	Illustration of the relation between FEM Micro classes using an UML diagram	B3
B5	GID	Visualization of the results using GID Postprocess	B2, B4
B6	Results comparison	Comparation of the obtained results with the expected ones	B5

Task code	Name	Description	Preceding Tasks
C	**Topology Optimization Code in Swan Repository**		
C1	Introduction to TO	Visualization of explanatory videos on topological optimization and creation of a report summarizing their content	B6
C2	Density-based methods	Comprehension of how density-based methods work: SIMP, RAMP, SINH...	C1
C3	Level set method	Comprehension of how boundary-based methods work, specially the level set method	C1
C4	Theorical aspects	Deepening of theoretical aspects of TO: formulation of the problem, regularization of the problem, filters, shape functionals...	C1, C2, C3
C5	TO Macro - Analysis	Familiarization with the TO Macro code and comprehension of how the sequence of code is analysed with MATLAB	C4
C6	TO Macro - UML	Graphical representation of the relation between TO Macro classes using an UML diagram	C5
C7	TO Micro - Analysis	Familiarization with the TO Micro code and comprehension of how the sequence of code is analysed with MATLAB	C5
C8	TO Micro - UML	Graphical representation of the relation between TO Micro classes using an UML diagram	C7
C9	Micro-2D examples	Representation and comparison of plots for different α and β for MMA optimizer	C8
C10	Optimizers comparison	Comparison of different optimizers: Augmented Lagrangian, Null-Space, Bisection, Projected Gradient, SLERP, Hamilton-Jacobi...	C8

Task code	Name	Description	Preceding Tasks
D	**Material Design**		
D1	Shape functionals	Analysis of different existing shape functionals	C10
D2	Theoretical background	Introduction to the h_1, h_2 and h_3 cases	C10
D3	Case h_1	Simulations of the case h_1 to understand it (varying the parameters)	D2
D4	Case h_2	Simulations of the case h_2 to understand it (varying the parameters)	D3
D5	Case h_3	Simulations of the case h_3 to understand it (varying the parameters)	D4
D6	3D design - density method	3D material design applying density-based methods	D5
D7	3D design - level set method	3D material design applying level set method	D5
E	**Inverse Homogenization**		
E1	Inverse homog. problem	Research about the elastic inverse homogenization problem	D6, D7
E2	Inverse homog. - density method	Study of the inverse homogenization applying density-based methods	E1
E3	Inverse homog. - level set method	Study of the inverse homogenization applying level set method	E1
F	**Project Planning**		
F1	Progress reports	Report writing in Overleaf of the progress made (including diagrams, summary of topics...)	-
F2	Project Charter	Preparation of the Project Charter (justification, requirements, tasks calendar, ...)	-
F3	Final Thesis report	Development of the Final Thesis documents: memory, budget, self-assessment, annex...	-
F4	Final Thesis defense	Planning and carry out the Final Thesis defense	-
F5	Weekly meetings	Periodic weekly meetings with the Final Thesis director to monitor the progress done	-

Table A.1: Calendar of the Final Thesis tasks.

FINAL THESIS - GANTT DIAGRAM



Appendix B

Micro TO Problem Formulation

This section presents the equations that constitute the microscopic optimization problem, along with the derivation of the corresponding Lagrangian function and the expression for the Lagrange multiplier that solves it. To enhance the understanding of the role of each variable, important parameters have been assigned different colors for clarity and ease of comprehension.

B.1 Homogenization Theory

Defining:

$$\begin{cases} \sigma_\mu\left(\varepsilon_{ij}\right) = \mathbb{C}[\varepsilon_{ij} + \varepsilon(\omega(\varepsilon_{ij}))] \\ \sigma^h(\varepsilon_{ij}) = \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \sigma_\mu\left(\varepsilon_{ij}\right) = \dfrac{1}{|\Omega|}\int_\Omega \mathbb{C}[\varepsilon_{ij} + \varepsilon(\omega_{ij}(\varepsilon_{ij}))] \end{cases}$$

with:

$$\begin{aligned} \omega_{ij}(\varepsilon_{ij}) \;=\; arg\ \min_{\omega_{ij}} \quad & \frac{1}{|\Omega|}\int_\Omega \frac{1}{2}[\varepsilon(\omega_{ij}) + \varepsilon_{ij}]\ \mathbb{C}\ [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \\ s.t. \quad & \frac{1}{|\Omega|}\int_\Omega \varepsilon(\omega_{ij}) \;=\; 0\ (\lambda) \end{aligned}$$

$$\Updownarrow$$

$$\begin{pmatrix} \min\limits_{\omega_{ij}} & \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \frac{1}{2}\sigma(\varepsilon_{ij}, \omega_{ij})\ [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \\ s.t. & \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \varepsilon(\omega_{ij}) \;=\; 0\ (\lambda) \end{pmatrix}$$

$$L(\omega_{ij}, \lambda) = \frac{1}{2}\frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}\, [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] + \lambda \frac{1}{|\Omega|}\int_\Omega \varepsilon(\omega_{ij})$$

$$\begin{cases} (1)\ \ \nabla_\omega\, L(\omega_{ij}, \lambda)_v = & \frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}\, \varepsilon(v) + \lambda \frac{1}{|\Omega|}\int_\Omega \varepsilon(v) & = 0 \\ (2)\ \ \nabla_\lambda\, L(\omega_{ij}, \lambda)_\mu = & \mu \frac{1}{|\Omega|}\int_\Omega \varepsilon(\omega_{ij}) & = 0 \end{cases}$$

$$\Updownarrow$$

$$\begin{bmatrix} K & \phi \\ \phi^T & 0 \end{bmatrix} \begin{bmatrix} \omega_{ij} \\ \lambda \end{bmatrix} = \begin{bmatrix} -D\,\mathbb{C}\,\varepsilon_{ij} \\ 0 \end{bmatrix}$$

In $\nabla_\omega\, L(\omega_{ij}, \lambda)_v = 0$; take v, $\varepsilon(v) = \varepsilon_{ij} + \varepsilon(\omega_{ij}(\varepsilon_{ij}))$ with $\omega_{ij}(\varepsilon_{ij}) = \omega_{ij}$ solution of (1) and (2).

$$\frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}\, [\varepsilon_{ij} + \varepsilon(\omega_{ij})] + \lambda \frac{1}{|\Omega|}\int_\Omega [\varepsilon_{ij} + \varepsilon(\omega_{ij})] = 0$$

$$\frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}\, [\varepsilon_{ij}] + \underbrace{\frac{1}{|\Omega|}\int_\Omega [\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}\, [\varepsilon(\omega_{ij})] + \lambda \frac{1}{|\Omega|}\int_\Omega \varepsilon(\omega_{ij})}_{=0 \quad (\text{take } v=\omega_{ij} \quad \text{in (1)})} + \lambda \frac{1}{|\Omega|}\int_\Omega \varepsilon_{ij} = 0$$

Since ε_{ij} is constant:

$$\left[\frac{1}{|\Omega|}\int_\Omega \underbrace{[\varepsilon(\omega_{ij}) + \varepsilon_{ij}] \,\mathbb{C}}_{\sigma_\mu(\varepsilon_{ij})}\right] [\varepsilon_{ij}] + \lambda\, \varepsilon_{ij} \frac{1}{|\Omega|}\int_\Omega 1 = 0$$

$$\Updownarrow$$

$$\boxed{\lambda = -\frac{1}{|\Omega|}\int_\Omega \sigma_\mu(\varepsilon_{ij}) = -\sigma^h(\varepsilon_{ij})}$$

B.2 Change of Variable

$$\varepsilon(u) = \varepsilon(\omega_{ij}) + \varepsilon_{ij} \qquad \varepsilon_{ij} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Defining :

$$\begin{cases} \sigma_\mu\left(\varepsilon_{ij}\right) = \mathbb{C}\ \varepsilon(u(\varepsilon_{ij})) \\ \sigma^h(\varepsilon_{ij}) = \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \sigma_\mu\left(\varepsilon_{ij}\right) = \dfrac{1}{|\Omega|}\int_\Omega \mathbb{C}\ \varepsilon(u(\varepsilon_{ij})) = \mathbb{C}^h : \varepsilon_{ij} \end{cases}$$

with:

$$\begin{aligned} u(\varepsilon_{ij}) \;=\; arg\ \ \underset{u}{min} \quad & \frac{1}{|\Omega|}\int_\Omega \frac{1}{2}\varepsilon(u)\ \mathbb{C}\ \varepsilon(u) \\ s.t. \quad & \frac{1}{|\Omega|}\int_\Omega \varepsilon(u) \;=\; \varepsilon_{ij}\ (\lambda) \end{aligned}$$

$$\Updownarrow$$

$$\begin{pmatrix} \underset{u}{min} & \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \frac{1}{2}\,\sigma(u)\ \varepsilon(u) \\ s.t. & \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \varepsilon(u) \;=\; \varepsilon_{ij}\ (\lambda) \end{pmatrix}$$

$$L(u,\lambda) = \frac{1}{|\Omega|}\int_\Omega \frac{1}{2}\ \varepsilon(u)\ \mathbb{C}\ \varepsilon(u) + \lambda\left(\frac{1}{|\Omega|}\int_\Omega \varepsilon(u) - \varepsilon_{ij}\right)$$

$$\begin{cases} (1)\ \ \nabla_u\ L(u,\lambda)_v = & \dfrac{1}{|\Omega|}\displaystyle\int_\Omega \varepsilon(u)\ \mathbb{C}\ \varepsilon(v) + \lambda\frac{1}{|\Omega|}\int_\Omega \varepsilon(v) & = 0 \\ (2)\ \ \nabla_\lambda\ L(u,\lambda)_\mu = & \mu\left[\dfrac{1}{|\Omega|}\displaystyle\int_\Omega \varepsilon(u) - \varepsilon_{ij}\right] & = 0 \end{cases}$$

$$\Updownarrow \text{Take } v = Ni$$

$$\begin{bmatrix} K & \phi \\ \phi^T & 0 \end{bmatrix}\begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \varepsilon_{ij} \end{bmatrix}$$

In $\nabla_u \, L(u, \lambda)_v = 0$; take v, $\varepsilon(v) = 1$ and $u(\varepsilon_{ij})$ solution of (1) and (2).

$$\frac{1}{|\Omega|} \int_\Omega [\varepsilon(u(\varepsilon_{ij})) \, \mathbb{C} + \lambda] \varepsilon(v) = 0$$

$$\Updownarrow$$

$$\boxed{\lambda = -\frac{1}{|\Omega|} \int_\Omega \mathbb{C} \, \varepsilon(u(\varepsilon_{ij})) = -\sigma^h(\varepsilon_{ij})}$$

Appendix C

Code

C.1 Master-Slave Computation

This section presents the code implementation used to establish the relationship between master and slave nodes, which is necessary for both 2D and 3D cases in order to apply the topology optimization code. Although they are part of the same class, they have been separated to facilitate the understanding of their individual roles in their respective cases.

C.1.1 2D case

```
classdef MasterSlaveRelator < handle

    properties (Access = private)
        x
        y
        z
        nodesInXmin
        nodesInXmax
        nodesInYmin
        nodesInYmax
        nodesInZmin
        nodesInZmax
        cellDescriptor
        msRelation
        allNodes
        cornerNodes
    end

    methods (Access = public)

        function obj = MasterSlaveRelator(coord)
            switch size(coord,2)
```

```
                case 2
                    obj.init2(coord);
                    obj.computeNodesInFaces(coord);
                    obj.computeMasterSlaveRelation();
                case 3
                    obj.init3(coord);
                    obj.computeNodesInFaces3D(coord);
                    obj.computeMasterSlaveRelation3D();
            end
        end

        function r = getRelation(obj)
            r = obj.msRelation;
        end

    end

    methods (Access = private)

        function init2(obj,coord)
            obj.x = coord(:,1);
            obj.y = coord(:,2);
            obj.allNodes(:,1) = 1:size(obj.x,1);
        end

        function computeNodesInFaces(obj,coord)
            cD = CellNodesDescriptor(coord);
            corners = cD.cornerNodes;
            obj.nodesInXmin = setdiff(cD.nodesInXmin,corners);
            obj.nodesInXmax = setdiff(cD.nodesInXmax,corners);
            obj.nodesInYmin = setdiff(cD.nodesInYmin,corners);
            obj.nodesInYmax = setdiff(cD.nodesInYmax,corners);
        end

        function computeMasterSlaveRelation(obj)
            [masterFaceX,slaveFaceX] = obj.computeMasterSlaveNodesInFaceX();
            [masterFaceY,slaveFaceY] = obj.computeMasterSlaveNodesInFaceY();
            obj.msRelation = [masterFaceX slaveFaceX;
                masterFaceY slaveFaceY];
        end

        function [master,slave] = computeMasterSlaveNodesInFaceX(obj)
            [master,slave] = ...
                obj.computeMasterSlaveNode(obj.nodesInXmin,obj.nodesInXmax,obj.y);
```

```
66          end
67
68          function [master,slave] = computeMasterSlaveNodesInFaceY(obj)
69              [master,slave] = ...
                    obj.computeMasterSlaveNode(obj.nodesInYmin,obj.nodesInYmax,obj.x);
70          end
71
72          function [master,slave] = ...
                computeMasterSlaveNode(obj,isLowerFace,isUpperFace,dir2compare)
73              nLF = obj.allNodes(isLowerFace,1);
74              nUF = obj.allNodes(isUpperFace,1);
75              master = nLF;
76              slave  = obj.obtainClosestNodeInOppositeFace(nLF,nUF,dir2compare);
77          end
78
79          function closestNodesUF = obtainClosestNodeInOppositeFace(obj,nodesLF,nodesUF,pos)
80              nNodes         = length(nodesLF);
81              closestNodesUF = zeros(nNodes,1);
82              for inode = 1:nNodes
83                  nodeLF = nodesLF(inode);
84                  closestNodesUF(inode) = obj.obtainClosestNode(nodeLF,nodesUF,pos);
85              end
86          end
87      end
88
89      methods (Access = private, Static)
90
91          function closestNodeB = obtainClosestNode(nodeA,nodeB,pos)
92              xA = pos(nodeA,1);
93              xB = pos(nodeB,1);
94              distAB = abs(xA - xB);
95              [distMin,isClosest] = min(distAB);
96              if distMin > 1e-10
97                  error('non slave node')
98              end
99              closestNodeB = nodeB(isClosest);
100         end
```

Code C.1: 2D case Master-Slave matrix computation. Extracted from [57]

C.1.2 3D case

```
...

    methods (Access = private)

        function init3(obj,coord)
            obj.x = coord(:,1);
            obj.y = coord(:,2);
            obj.z = coord(:,3);
            obj.allNodes(:,1) = 1:size(obj.x,1);
        end

        function computeNodesInFaces3D(obj,coord)
            cD = CellNodesDescriptor(coord);
            corners = cD.cornerNodes;
            obj.nodesInXmin = setdiff(cD.nodesInXmin,corners);
            obj.nodesInXmax = setdiff(cD.nodesInXmax,corners);
            obj.nodesInYmin = setdiff(cD.nodesInYmin,corners);
            obj.nodesInYmax = setdiff(cD.nodesInYmax,corners);
            obj.nodesInZmin = setdiff(cD.nodesInZmin,corners);
            obj.nodesInZmax = setdiff(cD.nodesInZmax,corners);
        end

        function computeMasterSlaveRelation3D(obj)
            [masterFaceX,slaveFaceX] = obj.computeMasterSlaveNodesInFaceX3D();
            [masterFaceY,slaveFaceY] = obj.computeMasterSlaveNodesInFaceY3D();
            [masterFaceZ,slaveFaceZ] = obj.computeMasterSlaveNodesInFaceZ3D();
            obj.msRelation = [masterFaceX slaveFaceX;
                masterFaceY slaveFaceY;
                masterFaceZ slaveFaceZ];
        end

        function [master,slave] = computeMasterSlaveNodesInFaceX3D(obj)
            [master,slave] = ...
                obj.computeMasterSlaveNode3D(obj.nodesInXmin,obj.nodesInXmax,obj.y,obj.z);
        end

        function [master,slave] = computeMasterSlaveNodesInFaceY3D(obj)
            [master,slave] = ...
                obj.computeMasterSlaveNode3D(obj.nodesInYmin,obj.nodesInYmax,obj.x,obj.z);
        end

```

```
40          function [master,slave] = computeMasterSlaveNodesInFaceZ3D(obj)
41              [master,slave] = ...
                    obj.computeMasterSlaveNode3D(obj.nodesInZmin,obj.nodesInZmax,obj.x,obj.y);
42          end
43
44          function [master,slave] = ...
                computeMasterSlaveNode3D(obj,isLowerFace,isUpperFace,dir2compare1,dir2compare2)
45              nLF = obj.allNodes(isLowerFace,1);
46              nUF = obj.allNodes(isUpperFace,1);
47              master = nLF;
48              slave  = obj.obtainClosestNodeInOppositeFace3D(nLF,nUF,dir2compare1,dir2compare2);
49          end
50
51          function closestNodesUF = ...
                obtainClosestNodeInOppositeFace3D(obj,nodesLF,nodesUF,pos1,pos2)
52              nNodes         = length(nodesLF);
53              closestNodesUF = zeros(nNodes,1);
54              for inode = 1:nNodes
55                  nodeLF = nodesLF(inode);
56                  closestNodesUF(inode) = obj.obtainClosestNode3D(nodeLF,nodesUF,pos1,pos2);
57              end
58          end
59
60          function avoidRepeatedEdges(obj)
61              MS = obj.msRelation;
62              uniqueCol1 = unique(MS(:,1));
63              uniqueCol2 = unique(MS(:,2));
64              commonNodes = intersect(uniqueCol1, uniqueCol2);
65              rowsRemoved = ismember(MS(:,1), commonNodes) | ismember(MS(:,2), commonNodes);
66              MS(rowsRemoved, :) = [];
67              obj.msRelation = MS;
68          end
69      end
70
71      methods (Access = private, Static)
72
73          function closestNodeB = obtainClosestNode3D(nodeA, nodeB, pos1,pos2)
74              xA = pos1(nodeA);
75              yA = pos2(nodeA);
76
77              xB = pos1(nodeB);
78              yB = pos2(nodeB);
79
80              distAB = sqrt((xA - xB).^2 + (yA - yB).^2);
```

```
81              [distMin, isClosest] = min(distAB);
82
83              if distMin > 1e-10
84                  error('non slave node')
85              end
86              closestNodeB = nodeB(isClosest);
87          end
88
89      end
```

Code C.2: 3D case Master-Slave matrix computation

C.2 Micro Elastic Problem

This section presents the code implementation used to generate a 3D cubic mesh with a hole inclusion and solve the micro elastic problem. Then, it calculates and outputs the strain, stress and displacement values throughout the structure.

```
1  clc; clear; close all
2
3  % Create the data container for the FEM problem
4  a.fileName = 'test3d_micro_cube';
5  m = FemDataContainer(a);
6
7  % Create the characteristic function (1 inside circle, 0 outside)
8  s.mesh    = m.mesh;
9  s.fxy     = @(x,y,z) (x-0.5).^2+(y-0.5).^2+(z-0.5).^2 -0.3.^2;
10 circleFun = CharacteristicFunction(s);
11
12 % Project the function to P0. Useful later on
13 x.mesh   = m.mesh;
14 x.connec = m.mesh.connec;
15 projP0 = Projector_toP0(x);
16 p0c = projP0.project(circleFun);
17
18 % Generate the hole in the material using the values we just found
19 fV = squeeze(p0c.fValues);
20 holeNodes = find(fV==1);
21 m.material.C(:,:,holeNodes) = m.material.C(:,:, holeNodes)*1e-3;
22
23 % Solve the elastic problem
24 fem = ElasticProblemMicro(m);
25 fem.computeChomog();
```

```
26 sss.filename = 'fluctHoleMaterial';
27 fem.uFun{1}.print(sss);
```

Code C.3: FEM computation of a 3D cubic mesh with a hole inclusion.

C.3 Paraview Transform Operation

To achieve a more efficient generation of the 3x3 grid mesh, the *Python Shell* tool in Paraview has been employed to create a code that translates the cube and visualizes the output.

```
1  from paraview.simple import *
2  import glob
3
4  # Set the file path
5  file_path = 'C:/Users/Usuari/Documents/...'
6  file_name = glob.glob(file_path)[0]
7
8  # Load the file and display the density values
9  reader = XMLUnstructuredGridReader(FileName=file_name)
10 display = Show(reader)
11 ColorBy(display, ('POINTS', 'Density'))
12
13 # Perform the Translate operation
14 translate_filter1 = Transform(Input=reader)
15 translate_filter1.Transform.Translate = [1, 0, 0]
16 Show(translate_filter1)
17
18 translate_filter2 = Transform(Input=reader)
19 translate_filter2.Transform.Translate = [-1, 0, 0]
20 Show(translate_filter2)
21
22 translate_filter3 = Transform(Input=reader)
23 translate_filter3.Transform.Translate = [1, 1, 0]
24 Show(translate_filter3)
25
26 translate_filter4 = Transform(Input=reader)
27 translate_filter4.Transform.Translate = [0, 1, 0]
28 Show(translate_filter4)
29
30 translate_filter5 = Transform(Input=reader)
31 translate_filter5.Transform.Translate = [-1, 1, 0]
32 Show(translate_filter5)
```

```
33
34 translate_filter6 = Transform(Input=reader)
35 translate_filter6.Transform.Translate = [1, -1, 0]
36 Show(translate_filter6)
37
38 translate_filter7 = Transform(Input=reader)
39 translate_filter7.Transform.Translate = [0, -1, 0]
40 Show(translate_filter7)
41
42 translate_filter8 = Transform(Input=reader)
43 translate_filter8.Transform.Translate = [-1, -1, 0]
44 Show(translate_filter8)
```

Code C.4: Paraview mesh grid obtention

Appendix D

Finite Element Method

D.1 Introduction

The Finite Element Method (FEM) is a numerical approximation technique that partitions a complex domain into a finite number of smaller elements, enabling the behavior of each element to be described by simplified equations. [59]

Derived from fundamental principles such as Newton's laws of motion, conservation of mass and energy, and thermodynamic laws, FEM enables the determination of structural mechanics, heat flow and electromagnetic radiation distribution in various scenarios. For instance, it can be employed to analyze the structural integrity of different car components under diverse loads, investigate heat transfer in engine parts, or study the propagation of electromagnetic waves from an antenna.

The process of dividing a comprehensive domain into simpler parts offers several benefits: a precise representation of intricate geometries, the incorporation of varying material properties, an efficient representation of the overall solution and the inclusion of localized effects.

For a better comprehension of the method, a 1D bar element case will be studied. This scenario will serve as a clear illustration of the equation derivation process.

D.2 Problem Formulation (Strong form)

Consider a bar with a length L and a constant cross-sectional area A, characterized by Young's Modulus E. The bar is subjected to prescribed displacement $u(0) = -g$ at the left end ($x = 0$) and an axial force F at the right end ($x = L$). Additionally, a distributed axial force per unit area $q(x) = E(\rho u(x) - sx^2)$ acts on the bar. Here, $u = u(x)$ represents the displacement field, g is a constant and further details are provided below. [60]

$$\rho = \frac{\pi^2}{L^2} \qquad s = g\rho^2 \qquad \frac{F}{AE} = \frac{g\pi^2}{L} \tag{D.1}$$

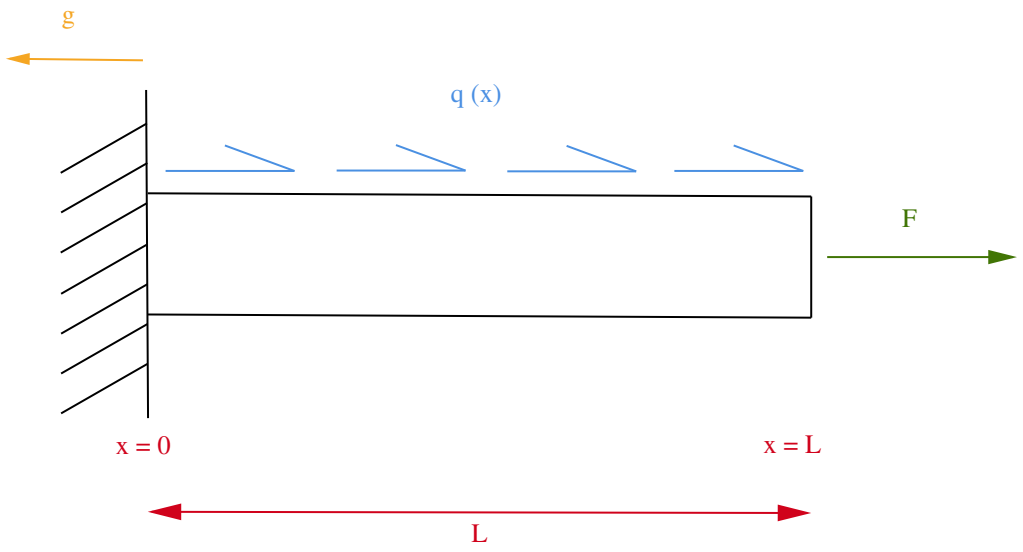


Figure D.1: FEM 1D bar element diagram

The strong form of the problem corresponds to the standard differential equation that defines it, which establishes certain conditions of continuity and differentiability for the potential solutions of the equation. [61]

In the studied case, the strong form of the problem is obtained by applying the axial equilibrium equation for 1D problems, given as:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}x} + q(x) = 0 \tag{D.2}$$

where $\sigma : \overline{\Omega} \to \mathbb{R}$.

By applying Hooke's law ($\sigma = E\frac{\mathrm{d}u}{\mathrm{d}x}$), the equation becomes:

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{q(x)}{E} = 0 \tag{D.3}$$

Substituting this into Equation (D.2) and expanding the expression:

$$u'' + \frac{E(\rho u - sx^2)}{E} = 0 \tag{D.4}$$

$$u'' + \rho u - sx^2 = 0 \tag{D.5}$$

$$u'' + \frac{\pi^2}{L^2}u - g\left(\frac{\pi^2}{L^2}\right)^2 x^2 = 0 \tag{D.6}$$

Regarding the boundary conditions, it is mentioned in the statement that the initial displacement of the bar is given by

$$u(0) = -g \tag{D.7}$$

Additionally, to determine the slope of the bar at its end, Hooke's law can be multiplied by the cross-sectional area:

$$A\sigma = AEu' \Longrightarrow u'(L) = \frac{F}{AE} \xrightarrow{\text{Eq. B.1}} u'(L) = \frac{g\pi^2}{L} \tag{D.8}$$

Hence, the strong form of the problem is expressed as follows:

$$\boxed{u'' + \frac{\pi^2}{L^2}u - g\left(\frac{\pi^2}{L^2}\right)^2 x^2 = 0 \qquad with \qquad \begin{cases} u(0) = -g \\ u'(L) = \dfrac{g\pi^2}{L} \end{cases}} \tag{D.9}$$

D.3 Problem Formulation (Weak form)

The weak form of the problem is an alternative formulation of the differential equation, expressed as an integral equation that implicitly encompasses the underlying differential equation. [61]

To simplify the Boundary Value Problem and derive its weak form, the constants from Equation (D.9) have been rearranged.

$$u'' + \frac{\pi^2}{L^2}u - g\left(\frac{\pi^2}{L^2}\right)^2 x^2 = 0 \xrightarrow{\begin{cases} a = \frac{\pi^2}{L^2} \\ f(x) = -g\left(\frac{\pi^2}{L^2}\right)^2 x^2 \end{cases}} u'' + au + f(x) = 0 \tag{D.10}$$

Then, an arbitrary sufficiently smooth function $v(x)$ is introduced, satisfying the condition $v(0) = 0$.

By relocating the terms independent of $u(x)$ to the right-hand side, multiplying both sides of the equation by $v(x)$, and integrating over the interval $[0, L]$, the following expression is obtained:

$$u'' + au = -f(x) \tag{D.11}$$

$$\int_0^L (u'' + au)\, v\mathrm{d}x = -\int_0^L f(x)v\mathrm{d}x \tag{D.12}$$

$$\int_0^L u''v\mathrm{d}x + \int_0^L au'v'\mathrm{d}x = -\int_0^L f(x)v\mathrm{d}x \tag{D.13}$$

Applying the chain rule to the derivative of uv: $(vu')' = v'u' + vu'' \rightarrow vu'' = (vu')' - v'u'$

$$[vu']_0^L - \int_0^L v'u'\mathrm{d}x + \int_0^L auv\mathrm{d}x = -\int_0^L fv\mathrm{d}x \tag{D.14}$$

$$v(L)\underbrace{u'(L)}_{=\frac{g\pi^2}{L}} - \underbrace{v(0)}_{=0}u'(0) - \int_0^L v'u'\mathrm{d}x + \int_0^L auv\mathrm{d}x = -\int_0^L fv\mathrm{d}x \tag{D.15}$$

Substituting the expressions of a and $f(x)$ presented above, the Variational form of the Boundary Value Problem is obtained:

$$\boxed{v(L)\frac{g\pi^2}{L} - \int_0^L v'u'\mathrm{d}x + \frac{\pi^2}{L^2}\int_0^L uv\mathrm{d}x = g\left(\frac{\pi^2}{L^2}\right)^2\int_0^L x^2 v\mathrm{d}x} \tag{D.16}$$

D.4 Generic Matrix Equation

By considering Equation (D.16) and applying the renaming of constant terms as a, b and $f(x)$, as previously done, the resulting equation is expressed as follows:

$$bv(L) - \int_0^L u'v'\mathrm{d}x + a\int_0^L uv\mathrm{d}x = -\int_0^L fv\mathrm{d}x \tag{D.17}$$

$$\begin{cases} a = \dfrac{\pi^2}{L^2} \\ b = \dfrac{g\pi^2}{L} \\ f(x) = -g\left(\dfrac{\pi^2}{L^2}\right)^2 x^2 \end{cases} \tag{D.18}$$

By assuming that both u and v can be represented as linear combinations of functions, specifically as $u(x) = N(x)\cdot d$ and $v(x) = N(x)\cdot c$, the resulting expression is obtained as follows:

$$\begin{cases} b\cdot v(L) = b\cdot N(L)\cdot c \\ u'\cdot v' = \dfrac{\mathrm{d}u}{\mathrm{d}x}\cdot\dfrac{\mathrm{d}v}{\mathrm{d}x} = \dfrac{\mathrm{d}N}{\mathrm{d}x}\cdot d\cdot\dfrac{\mathrm{d}N}{\mathrm{d}x}\cdot c = B\cdot d\cdot B\cdot c \\ u\cdot v = N\cdot d\cdot N\cdot c \\ f\cdot v = f\cdot N\cdot c \end{cases} \tag{D.19}$$

$$b\cdot N(L)\cdot c - \int_0^L B\cdot d\cdot B\cdot c\ \mathrm{d}x + a\int_0^L N\cdot d\cdot N\cdot c\ \mathrm{d}x = -\int_0^L f\cdot N\cdot c\ \mathrm{d}x \tag{D.20}$$

$$c^T\cdot N^T(L)\cdot b - \int_0^L c^T\cdot B^T\cdot B\ \mathrm{d}x\cdot d + a\int_0^L c^T\cdot N^T\cdot N\ \mathrm{d}x\cdot d = -\int_0^L c^T\cdot N^T\cdot f\ \mathrm{d}x \tag{D.21}$$

By rearranging the terms, the equation can be expressed in the following form, with all the terms combined on one side:

$$c^T\left[-N^T(L)\cdot b + \int_0^L B^T\cdot B\ \mathrm{d}x\cdot d - a\int_0^L N^T\cdot N\ \mathrm{d}x\cdot d - \int_0^L N^T\cdot f\ \mathrm{d}x\right] = 0 \tag{D.22}$$

By reorganizing the equation presented in Equation (D.22), the generic matrix equation that represents the Boundary Value Problem can be derived.

$$\boxed{c^T (Kd - F) = 0} \tag{D.23}$$

With the F and K matrices obtained below:

$$\boxed{\begin{cases} K = \int_0^L B^T \cdot B \, \mathrm{d}x - a \int_0^L N^T \cdot N \, \mathrm{d}x \\ F = N^T(L) \cdot b + \int_0^L N^T \cdot f \, \mathrm{d}x \end{cases}} \tag{D.24}$$

D.5 Assembly of the Global Stiffness Matrix

By recalling the general form of the matrix K, as indicated in Equation (D.24), and applying it to each element constituting the domain, the local expression of the matrix K results:

$$K^e = \int_{\Omega^e} (B^e)^T \cdot B^e \, \mathrm{d}x - a \int_{\Omega^e} (N^e)^T \cdot N^e \, \mathrm{d}x \tag{D.25}$$

In order to derive the left term of the expression, it is necessary to consider the definition of matrix B as $B = \frac{1}{h}[-1, 1]$. Applying this definition, the resulting expression is obtained as follows:

$$\int_{\Omega^e} (B^e)^T \cdot B^e \, \mathrm{d}x = \int_{\Omega^e} \left(\frac{1}{h^e}\begin{bmatrix}-1 & 1\end{bmatrix}\right)^T \cdot \left(\frac{1}{h^e}\begin{bmatrix}-1 & 1\end{bmatrix}\right) \mathrm{d}x = \frac{1}{(h^e)^2}\begin{bmatrix}-1 \\ 1\end{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix} \int_{\Omega^e} \mathrm{d}x = \frac{1}{h^e}\begin{bmatrix}1 & -1 \\ -1 & 1\end{bmatrix} \tag{D.26}$$

Regarding the right term of the expression, it can be redefined by applying the conversion from the physical domain to the reference domain. This transformation allows for the rewriting of the term as follows:

$$\begin{cases} N = \frac{1}{2}[1 - \xi \ , \quad 1 + \xi] \\ \mathrm{d}x = \frac{\mathrm{d}x}{\mathrm{d}\xi}\mathrm{d}\xi \\ x^e(\xi) = N^e(\xi) \cdot x^e \\ \frac{\mathrm{d}\xi}{\mathrm{d}x} = \frac{2}{h^e} \rightarrow \frac{\mathrm{d}x}{\mathrm{d}\xi} = \frac{h^e}{2} \end{cases} \tag{D.27}$$

Subsequently, by manually integrating the expression, it can be derived the following result:

$$\int_{\Omega^e} (N^e)^T \cdot N^e \, \mathrm{d}x = \int_{\Omega^e} \left((N^e)^T \cdot N^e \, \mathrm{d}\xi\right) \frac{h^e}{2} = \left(\int_{-1}^{+1} \frac{1}{2}\left[(1-\xi)\,,\ (1+\xi)\right]^T \cdot \frac{1}{2}\left[(1-\xi)\,,\ (1+\xi)\right] \mathrm{d}\xi\right) \frac{h^e}{2} =$$

$$= \frac{h^e}{8} \int_{-1}^{+1} \begin{bmatrix} 1-2\xi+\xi^2 & 1-\xi^2 \\ 1-\xi^2 & 1+2\xi+\xi^2 \end{bmatrix} = \frac{h^e}{8} \begin{bmatrix} 8/3 & 4/3 \\ 4/3 & 8/3 \end{bmatrix} = \frac{h^e}{8}\frac{4}{3} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \frac{h^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{D.28}$$

Hence, the matrix denoted as K for each element can be computed as follows:

$$K^e = \int_{\Omega^e} (B^e)^T \cdot B^e \, \mathrm{d}x - a \int_{\Omega^e} (N^e)^T \cdot N^e \, \mathrm{d}x = \frac{1}{h^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} - a \cdot \frac{h^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{D.29}$$

D.6 Assembly of the Global Forces Matrix

In the process of assembling the right-hand side vector (F), the following statement is considered:

$$F = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ b \end{bmatrix} + \overline{F} \tag{D.30}$$

To compute the vector F, the element force vectors need to be obtained individually in order to combine them during the assembly process.

The definition of the element vector (F^e) is as follows:

$$\overline{F}^e = \int_{\Omega^e} (N^e)^T f \mathrm{d}x \tag{D.31}$$

Which the assembly process can be written as:

$$\overline{F}^e = \sum_{e=1}^{n_{el}} L^{eT} \cdot \overline{F}^e \tag{D.32}$$

To transition from the physical domain to the reference domain, the quantity q is introduced, which is defined as the product of the transpose of the shape function matrix N^e and the distributed load vector f (so $q = N^{eT} f$). This allows for the rewriting of the element vector by utilizing Equation (D.31).

$$\overline{F}^e = \int_{-1}^{1} q(\xi)\xi \tag{D.33}$$

To compute the integral, the 1D Gauss Quadrature Method is employed. Consequently, the given equation can be expressed as follows:

$$\overline{F}^e = \frac{h^e}{2} \sum_{m}^{g=1} \omega_g \xi_g \tag{D.34}$$

Since f is a quadratic polynomial, the value of m is 2. Referring to Table D.1, the corresponding Gauss points yield the positions ξ_g and weights ω_g.

ξ_g	ω_g
$-1/\sqrt{3}$	1
$1/\sqrt{3}$	1

Table D.1: Gaussian quadrature location and weight for $m = 2$

As it has been defined before,

$$N^e(\xi_g) = \frac{1}{2}\left[(1-\xi_g)\,,\ \ (1+\xi_g)\right] \tag{D.35}$$

The values provided in Table D.1 are utilized to compute Equation (D.35).

Subsequently, the values from Table D.1 are also employed to evaluate the expression involving the transformation of the domain in the element coordinates:

$$x^e(\xi) = N^e(\xi) \cdot x^e \tag{D.36}$$

With the obtained values, they can be substituted into the expression for $f(\xi_g)$.

$$f(\xi_g) = f(x^e(\xi_g)) \tag{D.37}$$

So, the force contribution of each element is:

$$\overline{F}^e = \frac{h^e}{2}\left[(N^e)^T\left(-1/\sqrt{3}\right)\cdot f\left(-1/\sqrt{3}\right) + (N^e)^T\left(1/\sqrt{3}\right)\cdot f\left(1/\sqrt{3}\right)\right] \tag{D.38}$$

Appendix E

Material Design Functions Analysis

This section provides a detailed explanation of the calculation process for the derivative of the J_1, J_2 and J_3 functions, along with their corresponding code implementation.

To enhance the comprehension of the equations, it has been chosen that the characteristic function χ is represented by the density function ρ. This selection aims to facilitate the understanding of the equations and their interpretation.

E.1 Mathematical Background

This subsection provides a background on an important property of the derivative of an inverse matrix, which is relevant to the calculation discussed in the chapter.

As it is known, the identity matrix can be stated as:

$$A(x) \cdot A^{-1}(x) = I \tag{E.1}$$

Differenciating both sides of the equation, and applying the product rule, it results:

$$\frac{dA(x)}{dx} \cdot A^{-1} + A \cdot \frac{d(A^{-1})}{dx} = 0 \longrightarrow A^{-1} \cdot \frac{dA(x)}{dx} \cdot A^{-1} + \frac{d(A^{-1})}{dx} = 0 \tag{E.2}$$

By rearranging the terms, the derivative of the inverse of matrix A can be obtained:

$$\frac{d(A^{-1})}{dx} = -A^{-1} \cdot \frac{dA(x)}{dx} \cdot A^{-1} \tag{E.3}$$

Therefore, in the particular case at hand, matrix A will be replaced with matrix $\mathbb{C}$ (the homogenized two-dimensional elasticity tensor), leading to the following:

$$\frac{d}{d\rho}\left(W \cdot \mathbb{C}^{-1}(\rho)\right) = W \cdot \mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\rho} \cdot \mathbb{C}^{-1} \tag{E.4}$$

E.2 Weighted Inverse Homogenized Elasticity Matrix Function

E.2.1 Mathematical Demonstration

As it has been mentioned in Equation (6.4), this function can be expressed as:

$$J_1(\mathbb{C}) = \alpha^T \mathbb{C}^{-1} \beta \tag{E.5}$$

Taking the derivative of both sides of the equation with respect to the density and developing the terms (taking into account the property stated in Equation (E.4)):

$$\frac{\partial h_1(\mathbb{C})}{\partial \rho} = \alpha^T \cdot \frac{d(\mathbb{C}^{-1})}{d\rho} \cdot \beta \tag{E.6}$$

$$\frac{\partial h_1(\mathbb{C})}{\partial \rho} = -\alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\rho} \cdot \mathbb{C}^{-1} \right) \cdot \beta \tag{E.7}$$

E.2.2 Code Implementation

The code implementation for J_1 is showcased below:

```
classdef ShFunc_Chomog_alphabeta < ShFunc_Chomog
...
    methods (Access = public)
    ...
        function computeGradientValue(obj)
            obj.computeChDerivative();
            dinvChAB = obj.computedChInv(obj.Chomog,obj.alpha,obj.beta);
            obj.gradient = dinvChAB;
        end
    end
end
```

Code E.1: Gradient computation of J_1. Extracted from [57]

```
classdef ShFunc_Chomog < ShapeFunctional
...
    methods (Access = protected)
    ...
        function computeChDerivative(obj)
            dC = obj.homogenizedVariablesComputer.dC;
            p  = obj.physicalProblem;
            tstr = p.variables.tstrain;
            nStre = size(tstr,1);
            nelem = size(tstr,4);
            ngaus = size(tstr,2);
            dChV = zeros(nStre,nStre,nelem,ngaus);
            for iStre = 1:nStre
                for jStre = 1:nStre
                    for igaus=1:ngaus
                        for kStre =1:nStre
                            for lStre = 1:nStre
                                dChVij = squeeze(dChV(iStre,jStre,:,igaus));
                                eik    = squeeze(obj.tstrain(iStre,igaus,kStre,:));
                                dCkl   = squeeze(dC(kStre,lStre,:,:,igaus));
                                ejl    = squeeze(obj.tstrain(jStre,igaus,lStre,:));
                                dChV(iStre,jStre,:,igaus) = dChVij + eik.*dCkl.*ejl;%Original
                            end
                        end
                    end
                end
            end
            obj.dCh = dChV;
        end
    end
end
```

Code E.2: $\mathbb{C}$ derivative computation. Extracted from [57]

E.3 Rational Weighted Inverse Homogenized Elasticity Matrix Function

E.3.1 Mathematical Demonstration

As previously stated on Equation (6.6), this function can be formulated as follows:

$$J_2(\mathbb{C}) = \frac{\alpha^T \mathbb{C}^{-1} \beta}{\alpha^T \mathbb{C}^{-1} \alpha} + \frac{\beta^T \mathbb{C}^{-1} \alpha}{\beta^T \mathbb{C}^{-1} \beta} \tag{E.8}$$

By following a procedure analogous to that of function J_1 (specifically, by differentiating both sides of the equation and applying the matrix property from Equation (E.4)), the derivative of J_2 can be expressed as:

$$\frac{\partial J_2(\mathbb{C})}{\partial \rho} = \frac{\left(\alpha^T \mathbb{C}^{-1} \beta\right)' \cdot \alpha^T \mathbb{C}^{-1} \alpha - \alpha^T \mathbb{C}^{-1} \beta \left(\alpha^T \mathbb{C}^{-1} \alpha\right)'}{\left(\alpha^T \mathbb{C}^{-1} \alpha\right)^2} + \\ + \frac{\left(\beta^T \mathbb{C}^{-1} \alpha\right)' \cdot \beta^T \mathbb{C}^{-1} \beta - \beta^T \mathbb{C}^{-1} \alpha \left(\beta^T \mathbb{C}^{-1} \beta\right)'}{\left(\beta^T \mathbb{C}^{-1} \beta\right)^2} \tag{E.9}$$

$$\frac{\partial J_2(\mathbb{C})}{\partial \rho} = \frac{\left(-\alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \beta\right) \cdot \alpha^T \mathbb{C}^{-1} \alpha - \alpha^T \mathbb{C}^{-1} \beta \left(-\alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \alpha\right)}{\left(\alpha^T \mathbb{C}^{-1} \alpha\right)^2} + \\ + \frac{\left(-\beta^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \alpha\right) \cdot \beta^T \mathbb{C}^{-1} \beta - \beta^T \mathbb{C}^{-1} \alpha \left(-\beta^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \beta\right)}{\left(\beta^T \mathbb{C}^{-1} \beta\right)^2} \tag{E.10}$$

$$\frac{\partial J_2(\mathbb{C})}{\partial \rho} = \frac{\alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \left[\alpha \cdot \left(\alpha^T \mathbb{C}^{-1} \beta\right) - \beta \cdot \left(\alpha^T \mathbb{C}^{-1} \alpha\right)\right]}{\left(\alpha^T \mathbb{C}^{-1} \alpha\right)^2} + \\ + \frac{\beta^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{\mathrm{d}\mathbb{C}}{\mathrm{d}\rho} \cdot \mathbb{C}^{-1}\right) \cdot \left[\beta \cdot \left(\beta^T \mathbb{C}^{-1} \alpha\right) - \alpha \cdot \left(\beta^T \mathbb{C}^{-1} \beta\right)\right]}{\left(\beta^T \mathbb{C}^{-1} \beta\right)^2} \tag{E.11}$$

Where, rearranging the terms:

$$\frac{\partial J_2(\mathbb{C})}{\partial \rho} = \frac{g_1}{\left(\alpha^T \mathbb{C}^{-1} \alpha\right)^2} + \frac{g_2}{\left(\beta^T \mathbb{C}^{-1} \beta\right)^2} \tag{E.12}$$

$$\begin{cases} g_1 = \alpha^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\rho} \cdot \mathbb{C}^{-1}\right) \cdot \left[\alpha \cdot \left(\alpha^T \mathbb{C}^{-1} \beta\right) - \beta \cdot \left(\alpha^T \mathbb{C}^{-1} \alpha\right)\right] \\ g_2 = \beta^T \cdot \left(\mathbb{C}^{-1} \cdot \frac{d\mathbb{C}}{d\rho} \cdot \mathbb{C}^{-1}\right) \cdot \left[\beta \cdot \left(\beta^T \mathbb{C}^{-1} \alpha\right) - \alpha \cdot \left(\beta^T \mathbb{C}^{-1} \beta\right)\right] \end{cases} \tag{E.13}$$

E.3.2 Code Implementation

The code implementation for J_2 is provided below:

```
classdef ShFunc_Chomog_fraction < ShFunc_Chomog
...
    methods (Access = public)
    ...
        function computeGradientValue(obj)
            obj.computeChDerivative();
            a = obj.alpha;
            b = obj.beta;
            beta1 = obj.invChAA*b - obj.invChAB*a;
            beta2 = obj.invChBB*a - obj.invChBA*b;
            g1    = obj.computedChInv(obj.Chomog,a,beta1);
            g2    = obj.computedChInv(obj.Chomog,b,beta2);
            grad = g1/(obj.invChAA)^2 + g2/(obj.invChBB)^2;
            obj.gradient = grad;
        end
  ...
        function computeInvChProyections(obj)
            invCh = inv(obj.Chomog);
            a = obj.alpha;
            b = obj.beta;
            obj.invChAB  = obj.projectTensor(invCh,a,b);
            obj.invChBA  = obj.projectTensor(invCh,b,a);
            obj.invChAA  = obj.projectTensor(invCh,a,a);
            obj.invChBB  = obj.projectTensor(invCh,b,b);
        end
    end
end
```

Code E.3: Gradient computation of J_2 (part 1). Extracted from [57]

```
classdef ShFunc_Chomog < ShapeFunctional
...
    methods (Access = protected)
        function dChInv = computedChInv(obj,Ch,alpha,beta)
            invCh      = inv(Ch);
            nStres     = obj.getnStre();
            weights    = alpha*beta';
            weightsInv = invCh*weights*invCh;
            dChInv     = zeros(size(obj.dCh,[3 4]));
            for iStre = 1:nStres
                for jStre = 1:nStres
                    DtCij  = squeezeParticular(obj.dCh(iStre,jStre,:,:),[1 2]);
                    wij    = weightsInv(iStre,jStre);
                    dChInv = dChInv - wij*DtCij;
                end
            end
        end
        ...
        function aCb = projectTensor(C,a,b)
            ab  = a*b';
            aCb = ab.*C;
            aCb = sum(aCb(:));
        end
    end
end
```

Code E.4: Gradient computation of J_2 (part 2). Extracted from [57]

E.4 Inverse Problem Matrix Function

E.4.1 Mathematical Demonstration

Keeping in mind that this function can be written as expressed in Equation (6.11):

$$J_3(\mathbb{C}) = \left[\mathbb{C} - \mathbb{C}^*\right]^2 \tag{E.14}$$

it is trivial to demonstrate that its gradient can be expressed as seen in Equation (6.13):

$$\frac{\partial J_3(\mathbb{C})}{\partial \rho} = 2\left(\mathbb{C} - \mathbb{C}^*\right)\frac{\partial \mathbb{C}}{\partial \rho} \tag{E.15}$$

E.4.2 Code Implementation

```
classdef ShFunc_Chomog_CC < ShFunc_Chomog
    properties (Access = private)
        selectiveC_Cstar = [1, 1,1
                            1, 1,1
                            1,1,1];
    end

    methods
        ...
        function computeCostAndGradient(obj,x)
            obj.computePhysicalData(x);

            %Cost
            costfunc = obj.Chomog - obj.Ch_star;
            costfunc = obj.selectiveC_Cstar.*costfunc;

            %Gradient
            nstre = size(obj.tstrain,1);
            ngaus = size(obj.tstrain,2);
            nelem = size(obj.tstrain,4);

            obj.compute_Chomog_Derivatives(x);
            DtC1 = zeros(ngaus,nelem);
            gradient = zeros(ngaus,nelem);
            for igaus=1:ngaus
                for a=1:nstre
```

```
                    for b=1:nstre
                        DtC1(igaus,:) = squeeze(obj.Chomog_Derivatives(a,b,igaus,:));
                        gradient(igaus,:) = gradient(igaus,:) + 2*costfunc(a,b)*DtC1(igaus,:);
                    end
                end
            end

            %Cost
            costfunc = sum(bsxfun(@times,costfunc(:),costfunc(:)));

            mass=obj.Msmooth;
            gradient=obj.filter.getP1fromP0(gradient(:));
            gradient = mass*gradient;
            if isempty(obj.h_C_0)
                obj.h_C_0 = costfunc;
            end
            costfunc = costfunc/abs(obj.h_C_0);
            gradient=gradient/abs(obj.h_C_0);

            obj.value = costfunc;
            obj.gradient = gradient;
        end
    end
end
```

Code E.5: Gradient computation of J_3. Extracted from [57]

Appendix F

Previous Work

F.1 Optimizers Comparison

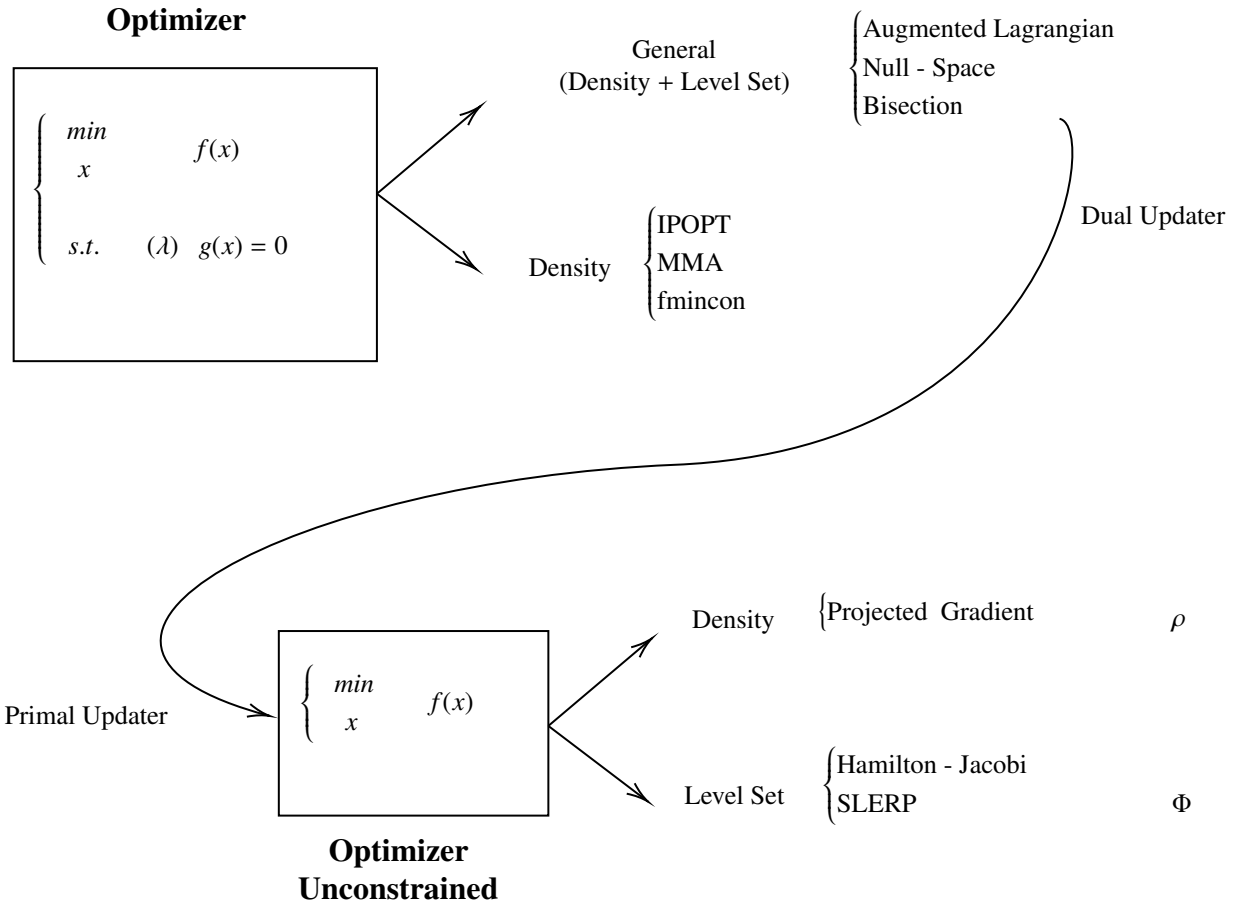
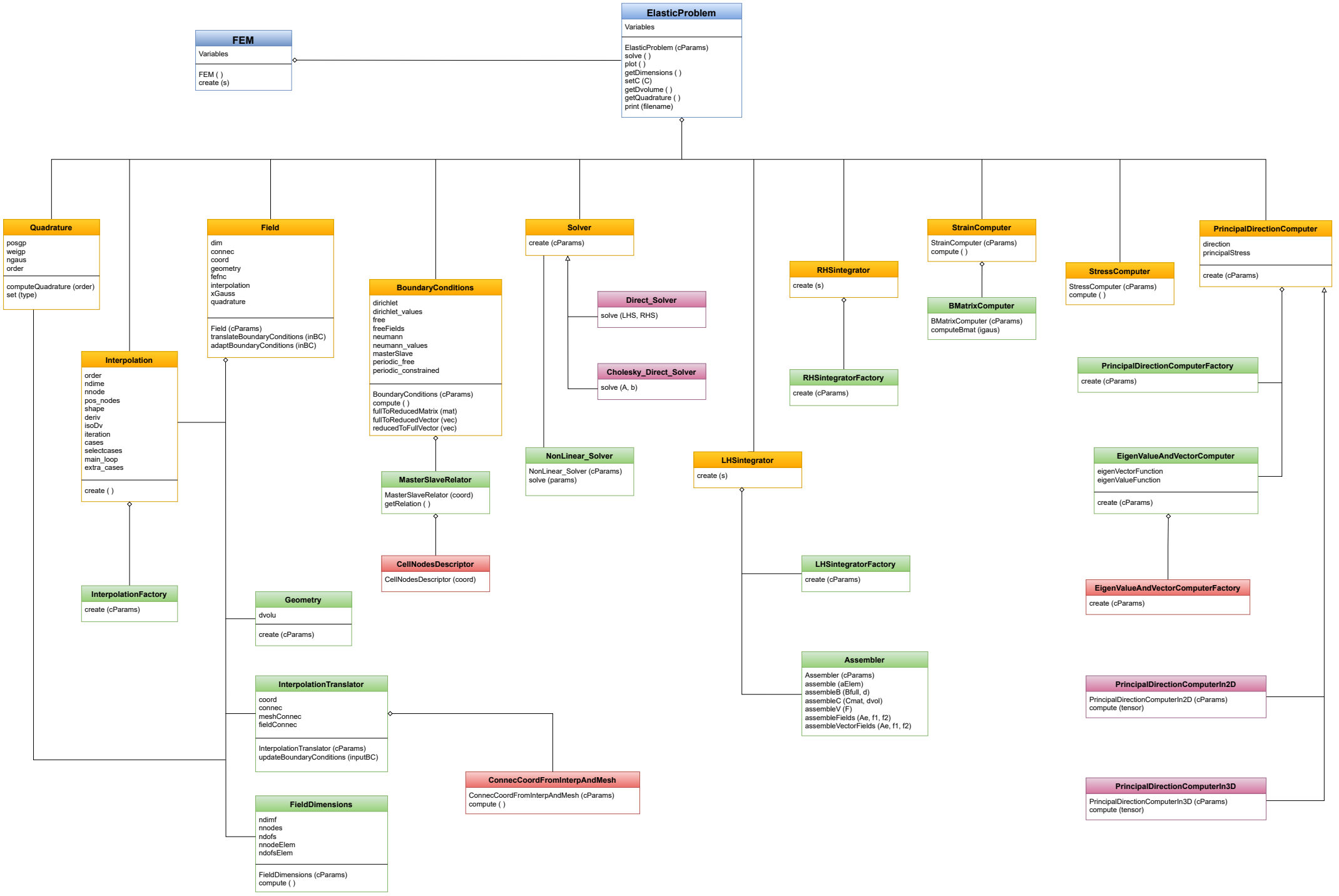


Figure F.1: Optimizers diagram

F.2 FEM UML Diagram


FEM
Variables
FEM ( )
create (s)
ElasticProblem
Variables
ElasticProblem (cParams)
solve ( )
plot ( )
getDimensions ( )
setC (C)
getDvolume ( )
getQuadrature ( )
print (filename)
Quadrature
posgp
weigp
ngaus
order
computeQuadrature (order)
set (type)
Field
dim
connec
coord
geometry
fefnc
interpolation
xGauss
quadrature
Field (cParams)
translateBoundaryConditions (inBC)
adaptBoundaryConditions (inBC)
BoundaryConditions
dirichlet
dirichlet_values
free
freeFields
neumann
neumann_values
masterSlave
periodic_free
periodic_constrained
BoundaryConditions (cParams)
compute ( )
fullToReducedMatrix (mat)
fullToReducedVector (vec)
reducedToFullVector (vec)
Solver
create (cParams)
Direct_Solver
solve (LHS, RHS)
Cholesky_Direct_Solver
solve (A, b)
RHSintegrator
create (s)
RHSintegratorFactory
create (cParams)
StrainComputer
StrainComputer (cParams)
compute ( )
BMatrixComputer
BMatrixComputer (cParams)
computeBmat (igaus)
StressComputer
StressComputer (cParams)
compute ( )
PrincipalDirectionComputer
direction
principalStress
create (cParams)
PrincipalDirectionComputerFactory
create (cParams)
Interpolation
order
ndime
nnode
pos_nodes
shape
deriv
isoDv
iteration
cases
selectcases
main_loop
extra_cases
create ( )
InterpolationFactory
create (cParams)
MasterSlaveRelator
MasterSlaveRelator (coord)
getRelation ( )
CellNodesDescriptor
CellNodesDescriptor (coord)
NonLinear_Solver
NonLinear_Solver (cParams)
solve (params)
LHSintegrator
create (s)
LHSintegratorFactory
create (cParams)
Assembler
Assembler (cParams)
assemble (aElem)
assembleB (Bfull, d)
assembleC (Cmat, dvol)
assembleV (F)
assembleFields (Ae, f1, f2)
assembleVectorFields (Ae, f1, f2)
EigenValueAndVectorComputer
eigenVectorFunction
eigenValueFunction
create (cParams)
EigenValueAndVectorComputerFactory
create (cParams)
Geometry
dvolu
create (cParams)
InterpolationTranslator
coord
connec
meshConnec
fieldConnec
InterpolationTranslator (cParams)
updateBoundaryConditions (inputBC)
ConnecCoordFromInterpAndMesh
ConnecCoordFromInterpAndMesh (cParams)
compute ( )
FieldDimensions
ndimf
nnodes
ndofs
nnodeElem
ndofsElem
FieldDimensions (cParams)
compute ( )
PrincipalDirectionComputerIn2D
PrincipalDirectionComputerIn2D (cParams)
compute (tensor)
PrincipalDirectionComputerIn3D
PrincipalDirectionComputerIn3D (cParams)
compute (tensor)

F.3 unfittedMesh

F.3.1 Graphic Representation

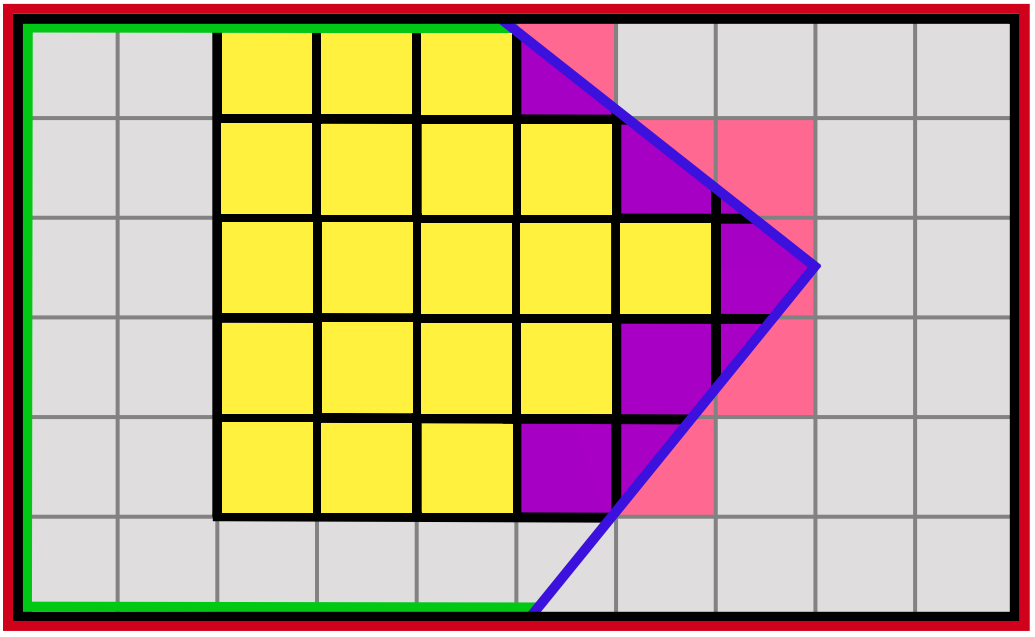
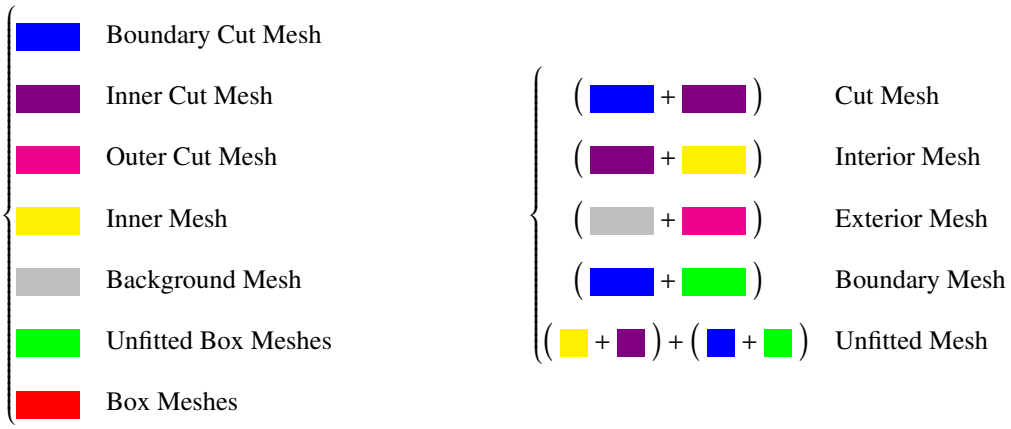
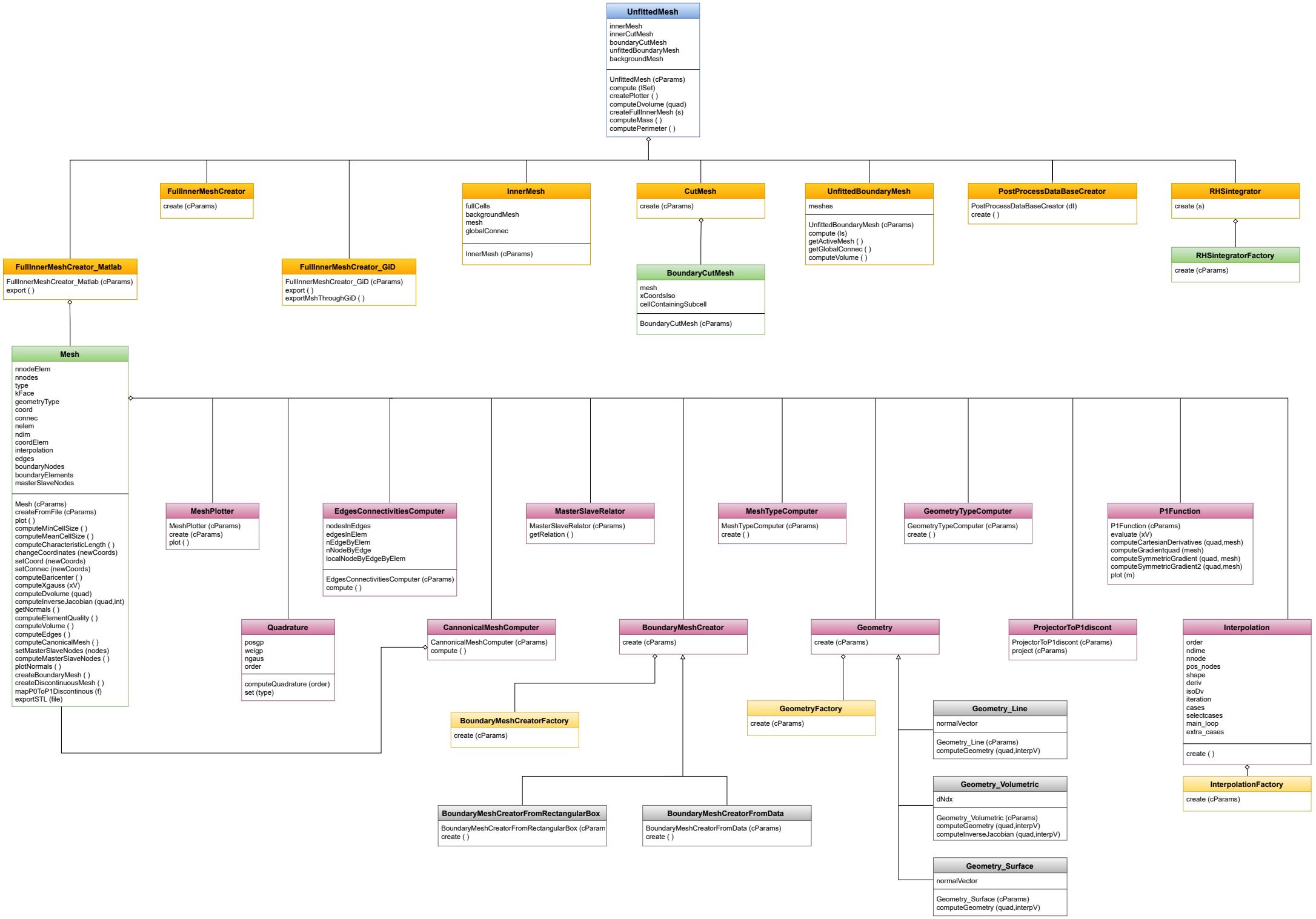


Figure F.2: unfittedMesh method example



F.3.2 UML Diagram


UnfittedMesh
innerMesh
innerCutMesh
boundaryCutMesh
unfittedBoundaryMesh
backgroundMesh
UnfittedMesh (cParams)
compute (ISet)
createPlotter ( )
computeDvolume (quad)
createFullInnerMesh (s)
computeMass ( )
computePerimeter ( )
FullInnerMeshCreator
create (cParams)
InnerMesh
fullCells
backgroundMesh
mesh
globalConnec
InnerMesh (cParams)
CutMesh
create (cParams)
UnfittedBoundaryMesh
meshes
UnfittedBoundaryMesh (cParams)
compute (ls)
getActiveMesh ( )
getGlobalConnec ( )
computeVolume ( )
PostProcessDataBaseCreator
PostProcessDataBaseCreator (dl)
create ( )
RHSintegrator
create (s)
RHSintegratorFactory
create (cParams)
FullInnerMeshCreator_Matlab
FullInnerMeshCreator_Matlab (cParams)
export ( )
FullInnerMeshCreator_GiD
FullInnerMeshCreator_GiD (cParams)
export ( )
exportMshThroughGiD ( )
BoundaryCutMesh
mesh
xCoordsIso
cellContainingSubcell
BoundaryCutMesh (cParams)
Mesh
nnodeElem
nnodes
type
kFace
geometryType
coord
connec
nelem
ndim
coordElem
interpolation
edges
boundaryNodes
boundaryElements
masterSlaveNodes
Mesh (cParams)
createFromFile (cParams)
plot ( )
computeMinCellSize ( )
computeMeanCellSize ( )
computeCharacteristicLength ( )
changeCoordinates (newCoords)
setCoord (newCoords)
setConnec (newCoords)
computeBaricenter ( )
computeXgauss (xV)
computeDvolume (quad)
computeInverseJacobian (quad,int)
getNormals ( )
computeElementQuality ( )
computeVolume ( )
computeEdges ( )
computeCanonicalMesh ( )
setMasterSlaveNodes (nodes)
computeMasterSlaveNodes ( )
plotNormals ( )
createBoundaryMesh ( )
createDiscontinuousMesh ( )
mapP0ToP1Discontinous (f)
exportSTL (file)
MeshPlotter
MeshPlotter (cParams)
create (cParams)
plot ( )
EdgesConnectivitiesComputer
nodesInEdges
edgesInElem
nEdgeByElem
nNodeByEdge
localNodeByEdgeByElem
EdgesConnectivitiesComputer (cParams)
compute ( )
MasterSlaveRelator
MasterSlaveRelator (cParams)
getRelation ( )
MeshTypeComputer
MeshTypeComputer (cParams)
create ( )
GeometryTypeComputer
GeometryTypeComputer (cParams)
create ( )
P1Function
P1Function (cParams)
evaluate (xV)
computeCartesianDerivatives (quad,mesh)
computeGradientquad (mesh)
computeSymmetricGradient (quad, mesh)
computeSymmetricGradient2 (quad,mesh)
plot (m)
Quadrature
posgp
weigp
ngaus
order
computeQuadrature (order)
set (type)
CannonicalMeshComputer
CannonicalMeshComputer (cParams)
compute ( )
BoundaryMeshCreator
create (cParams)
Geometry
create (cParams)
ProjectorToP1discont
ProjectorToP1discont (cParams)
project (cParams)
Interpolation
order
ndime
nnode
pos_nodes
shape
deriv
isoDv
iteration
cases
selectcases
main_loop
extra_cases
create ( )
InterpolationFactory
create (cParams)
BoundaryMeshCreatorFactory
create (cParams)
GeometryFactory
create (cParams)
Geometry_Line
normalVector
Geometry_Line (cParams)
computeGeometry (quad,interpV)
Geometry_Volumetric
dNdx
Geometry_Volumetric (cParams)
computeGeometry (quad,interpV)
computeInverseJacobian (quad,interpV)
Geometry_Surface
normalVector
Geometry_Surface (cParams)
computeGeometry (quad,interpV)
BoundaryMeshCreatorFromRectangularBox
BoundaryMeshCreatorFromRectangularBox (cParam
create ( )
BoundaryMeshCreatorFromData
BoundaryMeshCreatorFromData (cParams)
create ( )

Appendix G

Simulations

This chapter is dedicated to the conducted simulations. It focuses on presenting the parameters used in both the 2D and 3D simulations, as well as showcasing the results that were not included in Chapter 8, which pertain to the simulations performed with a final volume fraction of 0.85.

G.1 2D Simulations Parameters

The parameters utilized for the 2D simulations are provided below. It is important to note that the table does not specify the α and β values since they were employed but varied for each simulation, so their specific values were indicated for each case.

2D SIMULATION CONDITIONS	
Initial case	'circleInclusion'
Constraint	'volumeConstraint'
Problem type	'MICRO'
Material type	'ISOTROPIC'
Filter type	'P1'
Method	'SIMP_P3'
Diameter fraction	0.5
Cost	'chomog_alphabeta'
Final volume fraction	0.3

Table G.1: Common conditions in 2D J_1 simulations

G.2 3D Simulations Parameters

Regarding the common parameters utilized in the 3D simulations, excluding the alpha and beta values (as they were also specified in each presented case), the following parameters were employed:

3D SIMULATION CONDITIONS	
Initial case	'sphereInclusion'
Constraint	'volumeConstraint'
Problem type	'MICRO'
Material type	'ISOTROPIC'
Filter type	'P1'
Method	'SIMPALL'
Diameter fraction	0.4
Cost	'chomog_alphabeta'

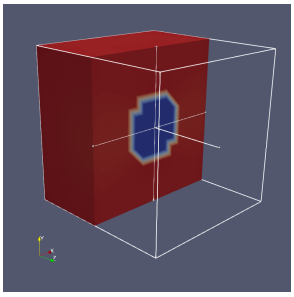
Table G.2: Common conditions in 3D J_1 simulations

G.3 3D Simulations Results

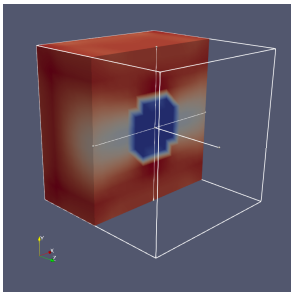
In this section, the results that were not included in Chapter 8 will be presented. Specifically, these results pertain to the optimization of normal materials using the MMA and Null Space optimizers, as well as the optimization of metamaterials using the Null Space optimizer. In both cases, a final volume fraction of 0.85 was employed.

G.3.1 Density-Based Method: MMA

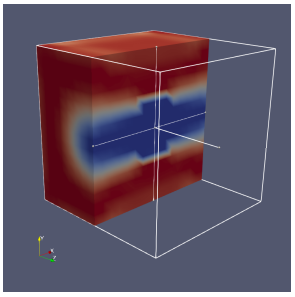
Cilinder: $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$



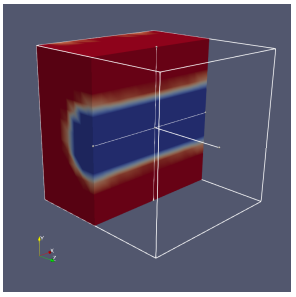
(a) Step 1.



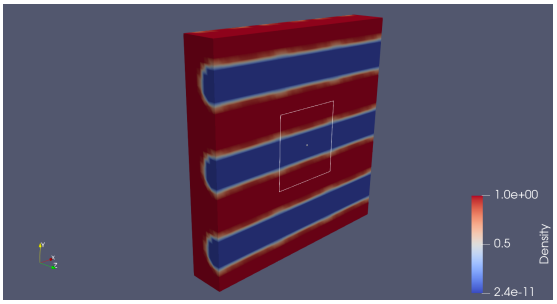
(b) Step 25.



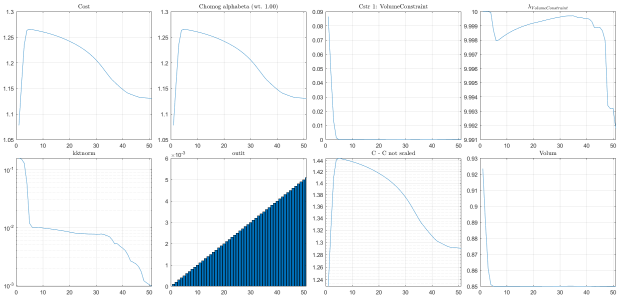
(c) Step 40.



(d) Step 65.



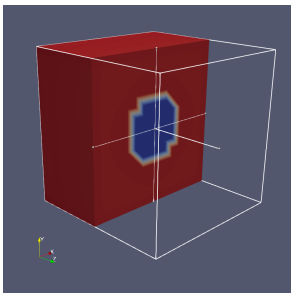
(e) Grid visualization.



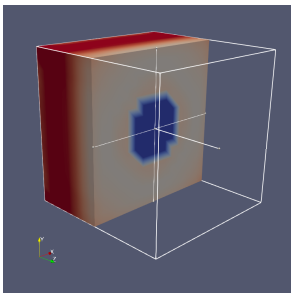
(f) Monitoring.

Figure G.1: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.85$).

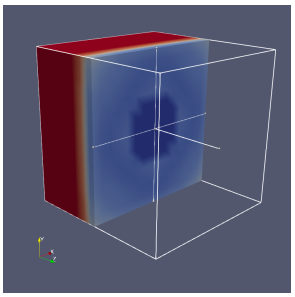
Vertical plate: $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$



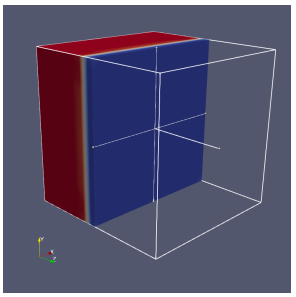
(a) Step 1.



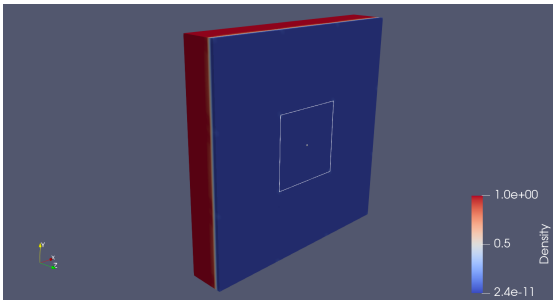
(b) Step 35.



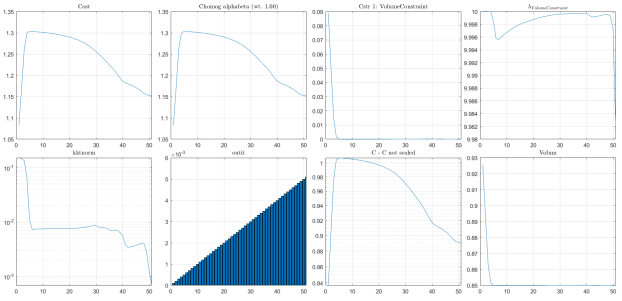
(c) Step 45.



(d) Step 51.



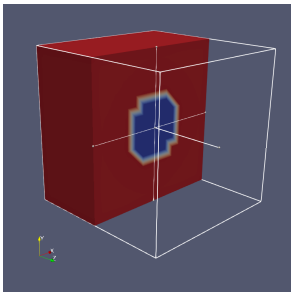
(e) Grid visualization.



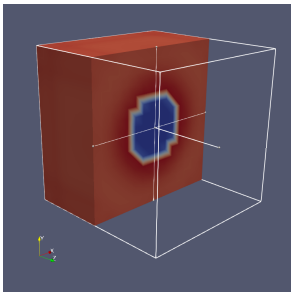
(f) Monitoring.

Figure G.2: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.85$).

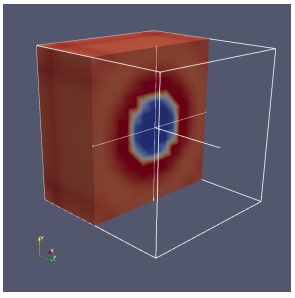
Sphere: $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$



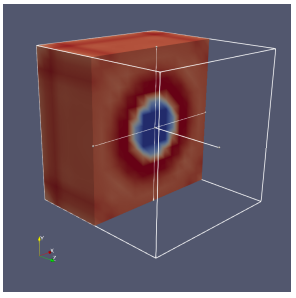
(a) Step 1.



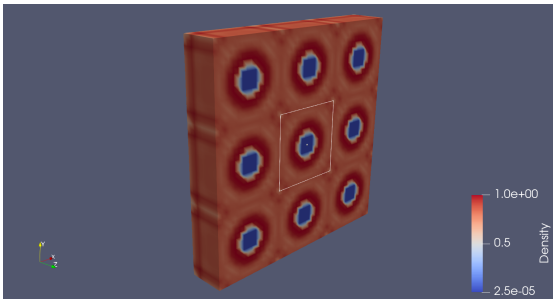
(b) Step 25.



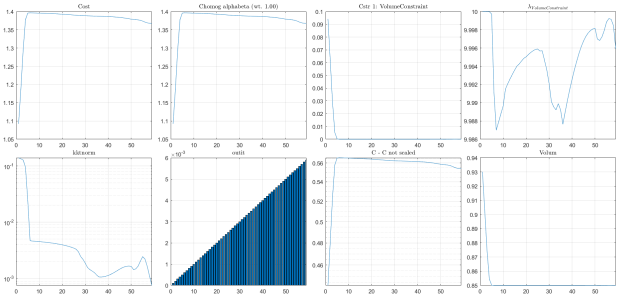
(c) Step 40.



(d) Step 59.



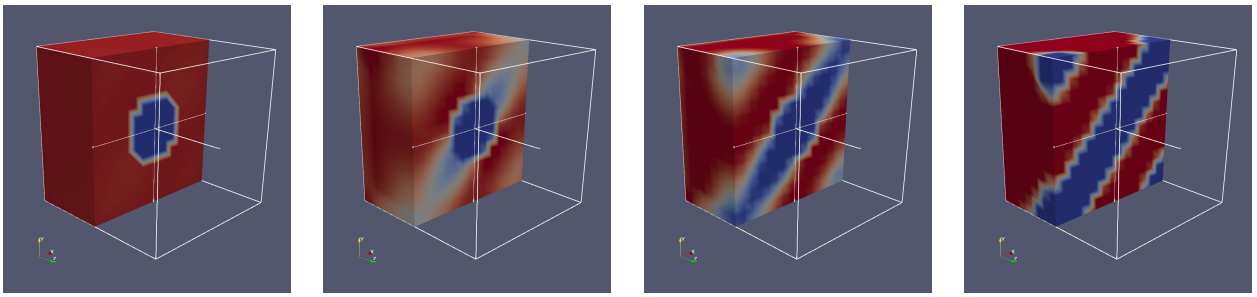
(e) Grid visualization.



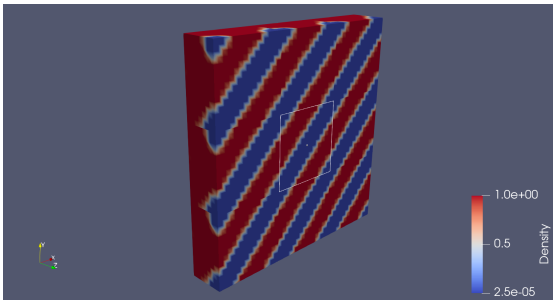
(f) Monitoring.

Figure G.3: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$ (MMA - $Vfinal = 0.85$).

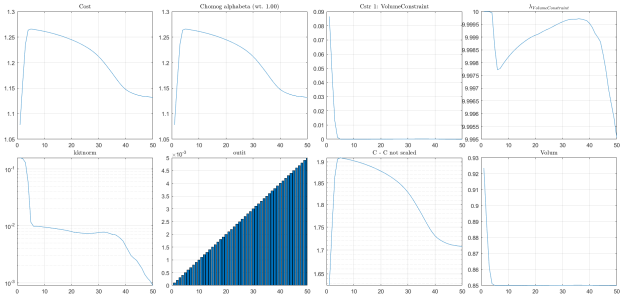
Diagonal: $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 30. (c) Step 40. (d) Step 50.



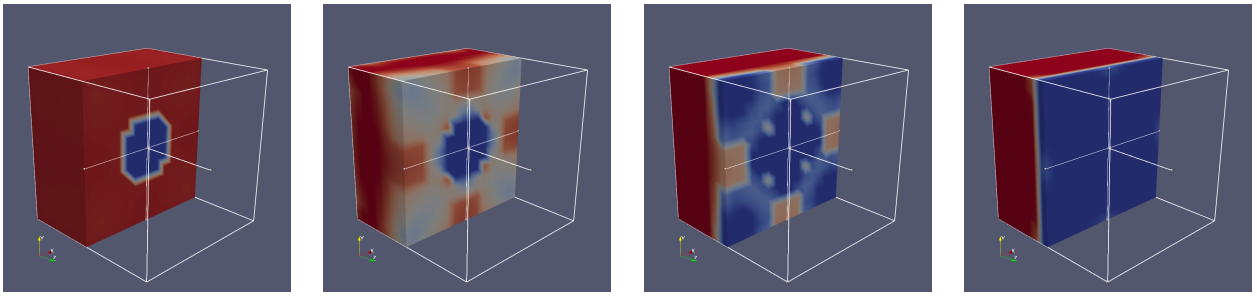
(e) Grid visualization.



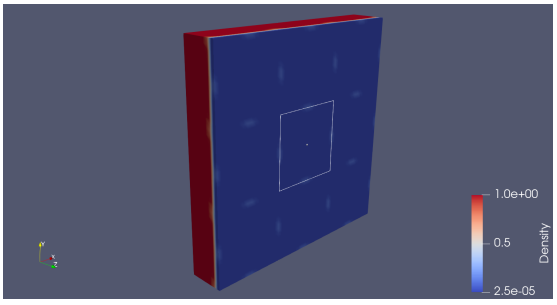
(f) Monitoring.

Figure G.4: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$ (MMA - $Vfinal = 0.85$).

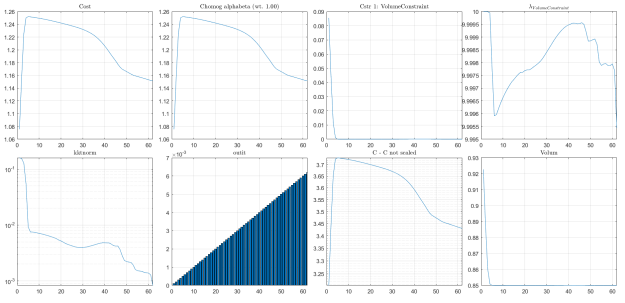
Shear: $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 40. (c) Step 50. (d) Step 62.



(e) Grid visualization.

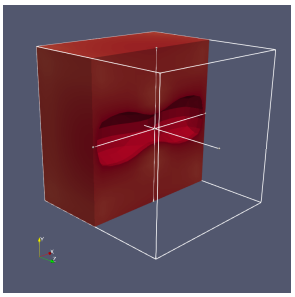


(f) Monitoring.

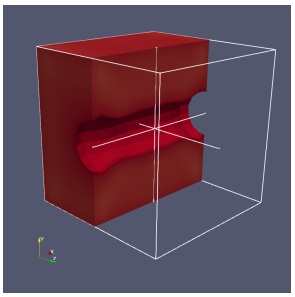
Figure G.5: Density results from the 3D J_1 cost function for $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$ (MMA - $Vfinal = 0.85$).

G.3.2 Level Set Method: Null Space

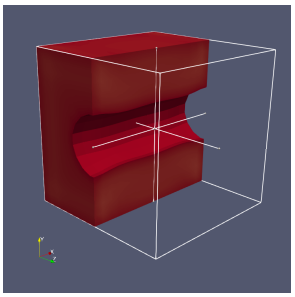
Cilinder: $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$



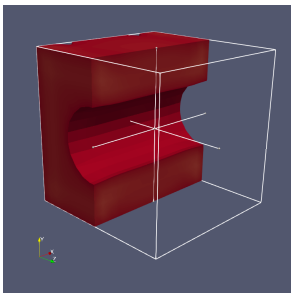
(a) Step 1.



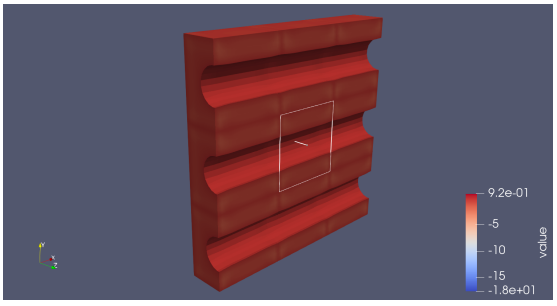
(b) Step 3.



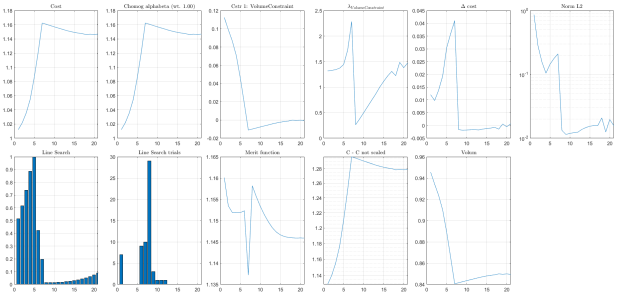
(c) Step 6.



(d) Step 10.



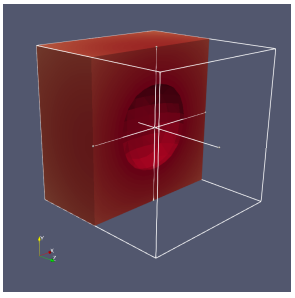
(e) Grid visualization.



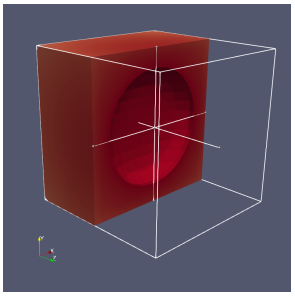
(f) Monitoring.

Figure G.6: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 0\ 0\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.85$).

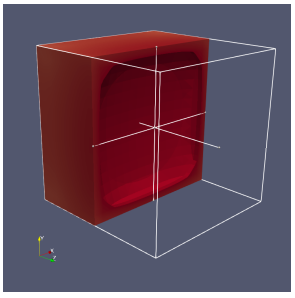
Vertical plate: $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$



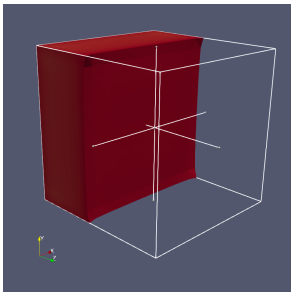
(a) Step 1.



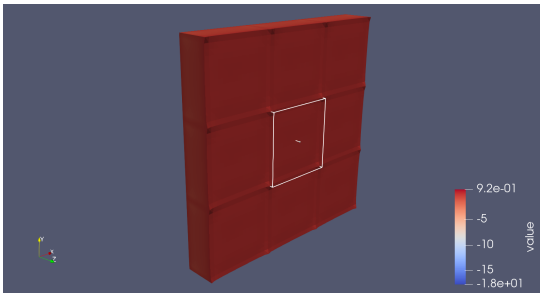
(b) Step 5.



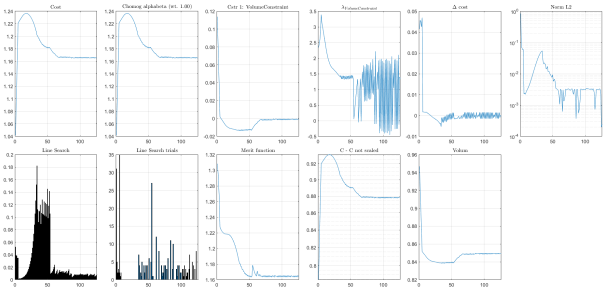
(c) Step 30.



(d) Step 80.



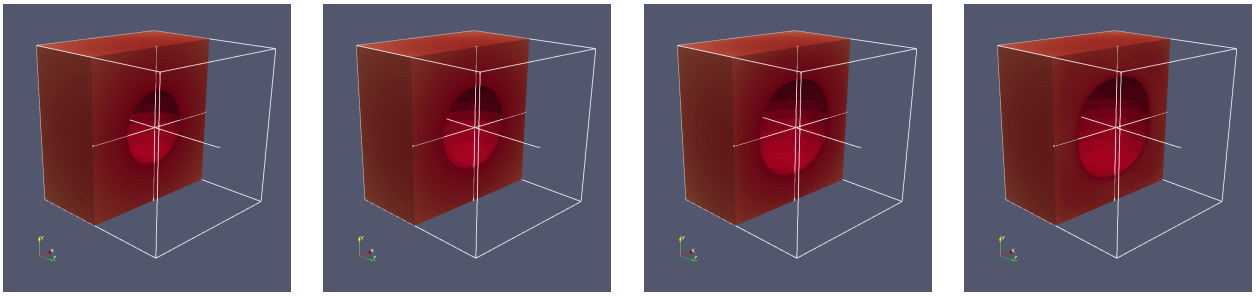
(e) Grid visualization.



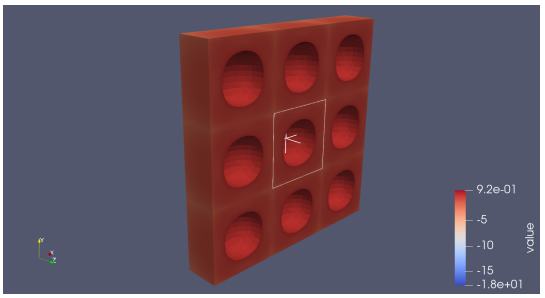
(f) Monitoring.

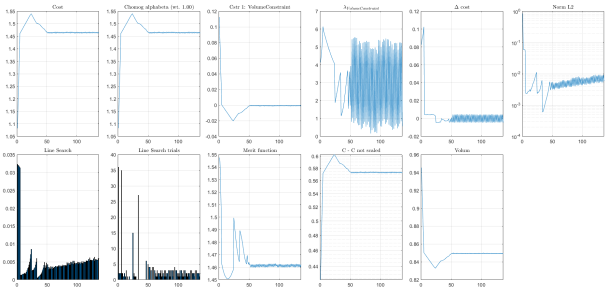
Figure G.7: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.85$).

Sphere: $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$



(a) Step 1. (b) Step 5. (c) Step 10. (d) Step 40.

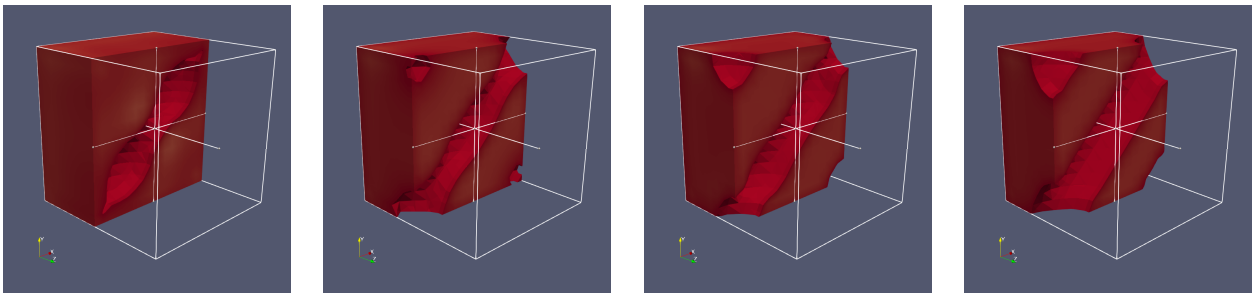




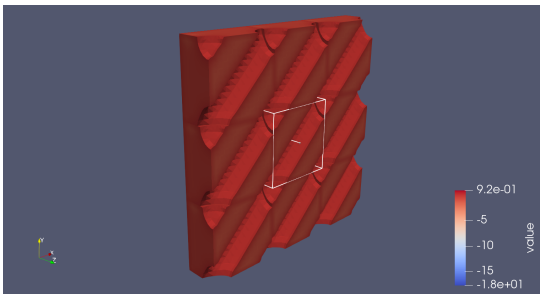
(e) Grid visualization. (f) Monitoring.

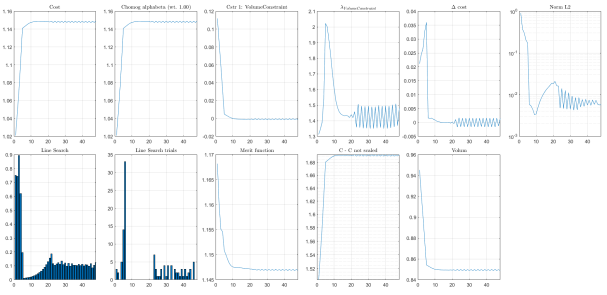
Figure G.8: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 1\ 0\ 0\ 0]$ (Null Space - $Vfinal = 0.85$).

Diagonal: $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$



(a) Step 1. (b) Step 3. (c) Step 5. (d) Step 30.

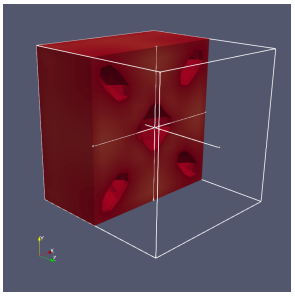




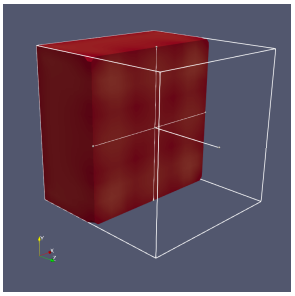
(e) Grid visualization. (f) Monitoring.

Figure G.9: Density results from the 3D J_1 cost function for $\alpha = \beta = [1\ 1\ 0\ 1\ 0\ 0]$ (Null Space - $Vfinal = 0.85$).

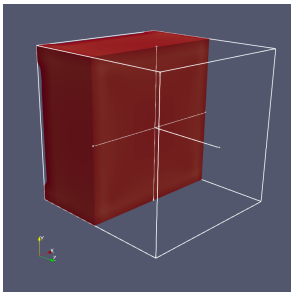
Shear: $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$



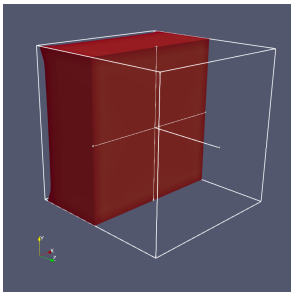
(a) Step 1.



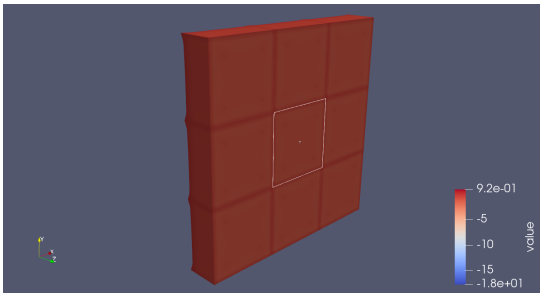
(b) Step 4.



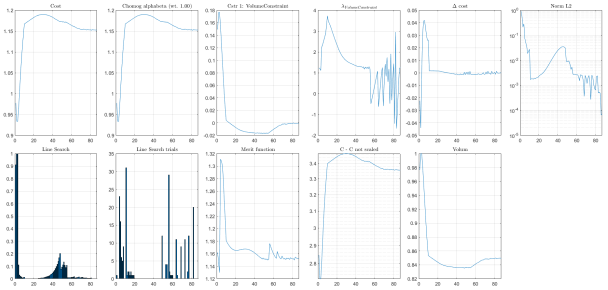
(c) Step 40.



(d) Step 70.



(e) Grid visualization.

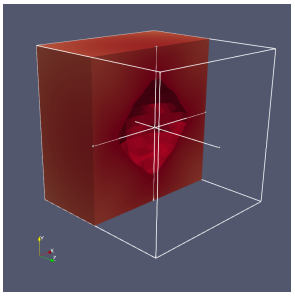


(f) Monitoring.

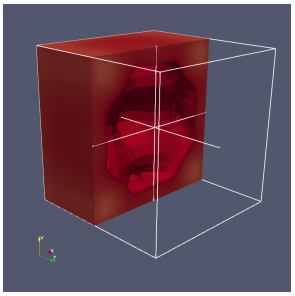
Figure G.10: Density results from the 3D J_1 cost function for $\alpha = \beta = [0\ 0\ 0\ 1\ 0\ 0]$ (Null Space - $Vfinal = 0.85$).

G.3.3 Metamaterials

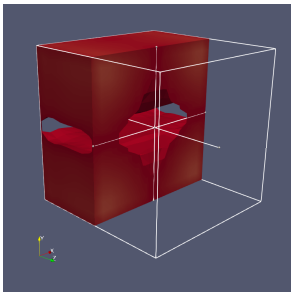
Case 1: $\alpha = [1\ 0\ 0\ 0\ 0\ 0]$ and $\beta = [0\ -1\ 0\ 0\ 0\ 0]$



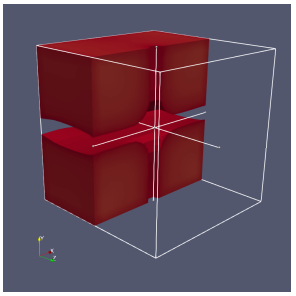
(a) Step 1.



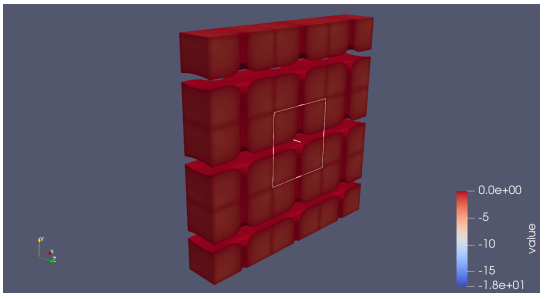
(b) Step 50.



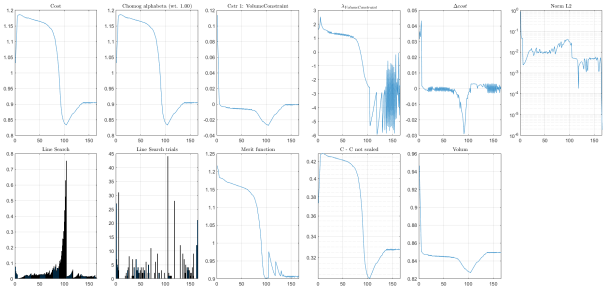
(c) Step 95.



(d) Step 166.



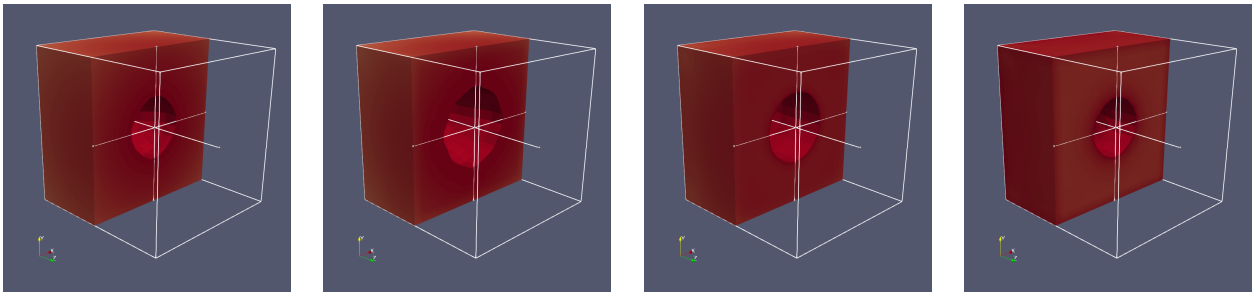
(e) Grid visualization.



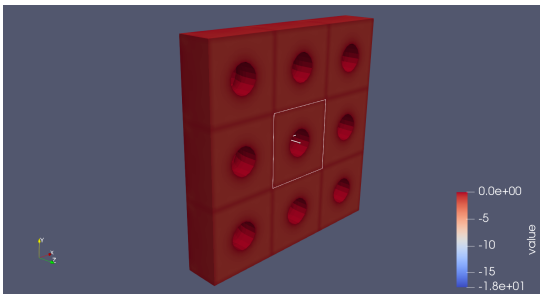
(f) Monitoring.

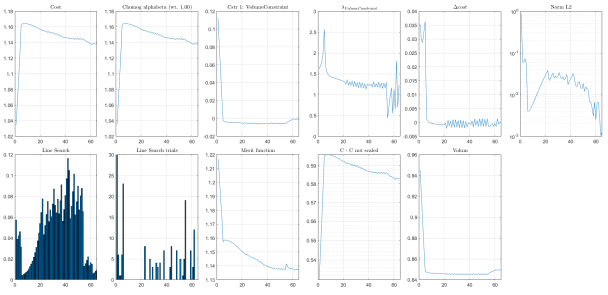
Figure G.11: Density results from the 3D J_1 cost function for case 1 metamaterial simulation ($Vfinal = 0.85$).

Case 2: α = [1 1 0 0 0 0] and β = [0 0 −1 0 0 0]



(a) Step 1. (b) Step 10. (c) Step 25. (d) Step 65.

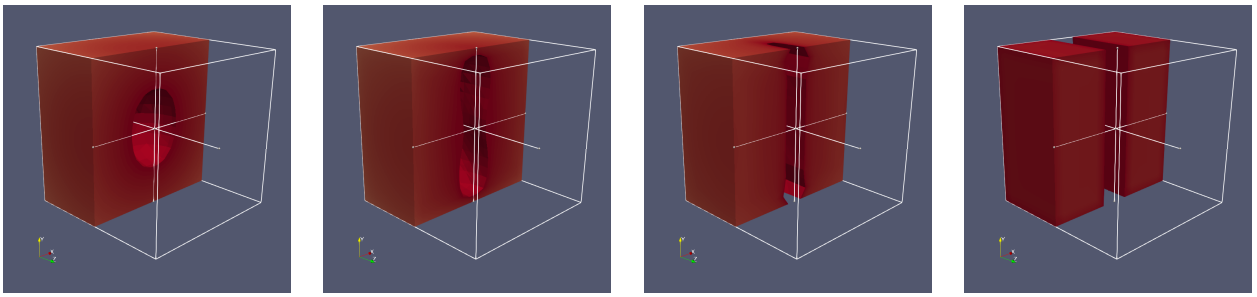




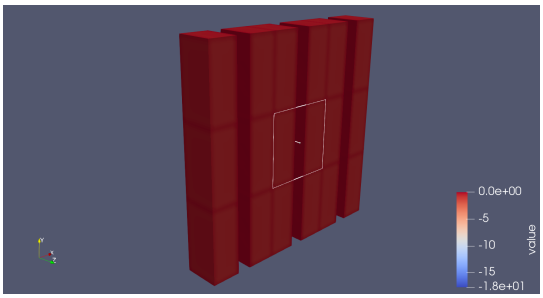
(e) Grid visualization. (f) Monitoring.

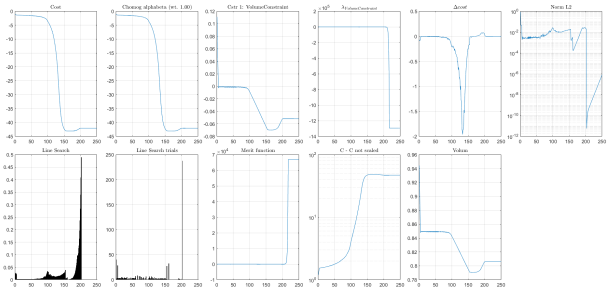
Figure G.12: Density results from the 3D J_1 cost function for case 2 metamaterial simulation ($Vfinal$ = 0.85).

Case 3: α = [1 0 0 0 0 0] and β = [−1 0 0 0 0 0]



(a) Step 1. (b) Step 70. (c) Step 110. (d) Step 220.





(e) Grid visualization. (f) Monitoring.

Figure G.13: Density results from the 3D J_1 cost function for case 3 metamaterial simulation ($Vfinal$ = 0.85).