Topology optimization of lattice materials with nonlinear elasticity law

Project report for

Numerical Analysis for Partial Differential Equations

A.Y. 2020 - 2021

Teresa Babini

Manfred Nesti

Tutors:

Prof. S. Perotto, Prof. S. Micheletti, Dr. N. Ferro

POLITECNICO
MILANO 1863

Contents

Chapter 1

Introduction and state of the art

Thanks to additive manufacturing techniques, in particular 3D printing, in the last years the interest about **lattices structures** and **metamaterials** strongly increased. Lattice structures and metamaterials are a class of cellular materials characterized by a regular, periodic microstructure which can be idealized as a network of slender beams or rods and which can have prescribed properties such as stiffness, strength along some directions, high energy absorbing, lightness, prescribed Poisson's ratio (also negative). These different properties can be obtained by solving a **Topology Optimization (TO)** problem which, given the prescribed macroscopic properties we want to obtain, provide us the optimized unit cell which repeated ensures that properties to the so build material.

Research on TO mainly deals with the design of monoscale structures, which are usually made of homogeneous materials, but recent advances of multiscale structural modeling enables the consideration of microscale material heterogeneities and constituent nonlinearities when assessing the macroscale structural performance (see [3]). However, due to the modeling complexity and the expensive computing requirement of multiscale modeling, there has been very limited research on TO of multiscale nonlinear structures.

Many work have already done to find, given same macroscopic properties, an optimized unit cell which provides them under a **linear elasticity law** (see [10]). The main goal we tried to reach in this work is to extend this method in order to take into account a possible **nonlinear elasticity law**. In our roadmap the main steps we followed are the following.

First of all we started looking for the state of the art in this field, looking for the basic but needed background in some topics like TO, its main algorithms and the homogenization technique, as explained in section 2. In particular, our starting point, in which all the tools are combined to build the linear method, is the work in [10], described in section 2.4. The second step was to build a nonlinear model (see [1], [2]) in order to extend the method to a nonlinear regime. In particular, during this phase, many chances arises regarding the usage and the tuning of some

filters for the solution of the problem (see 2.1.2 and 3.4). Moreover, also a grid adaptation procedure is often indicated, you will find the details in 2.2 and 3.4. Finally, we implemented in `FreeFem++` our optimization procedure (3.3), performing some tests to validate the model and simulating some nonlinear structures you can find in 3.4.

Chapter 2

General settings and mathematical tools

2.1 Topology Optimization

Topology Optimization (TO) is a mathematical method that optimizes material layout within a given design space, for a given set of loads, boundary conditions and constraints with the goal of minimizing a predefined cost function. For example, typical optimality criteria are minimum volume, minimum compliance (or maximum stiffness), maximum fundamental frequency in the dynamic case, while constraints can be maximum allowed displacements and stresses or a given fraction of the initial volume.

TO has a wide range of applications in aerospace, mechanical, bio-chemical and civil engineering. Currently, engineers mostly use TO at the concept level of a design process. Due to the free forms that naturally occur, the result is often difficult to manufacture. For that reason the result emerging from TO is often fine-tuned for manufacturability. Adding constraints to the formulation in order to increase the manufacturability is an active field of research. In some cases results from TO can be directly manufactured using additive manufacturing; TO is thus a key part of design for additive manufacturing.

For such an optimization procedure, the three main elements are **design variable** ρ, the **cost function** C and the **constraints**.

The most generic way in which can be defined a TO problem is the following:

Find $\rho = \arg\min_\rho C\left(\mathbf{u}\left(\rho\right), \rho\right)$ subject to

- governing equations of static or dynamic equilibrium
- resource constraints: $\int_\Omega \rho d\mathbf{x} \leqslant \alpha|\Omega|$
- performance constraints

- practical constraints
- bounds on the design variables: $\rho \in (\rho_{\min}, 1)$

where

- Ω is the design space which indicates the allowable volume within which the design can exists
- $\rho : \Omega \to [0, 1]$ is the design variable, namely the material distribution in the domain that can be 0 (void) or 1 (material) and, during the optimization, it can assume all the intermediate values
- α is the volume fraction of $|\Omega|$ in which we allow to be material
- $\rho_{\min}$ is the minimum value that ρ can assume during the optimization and it ensures the problem to be well-defined

2.1.1 The SIMP method

One of the well known mathematical techniques for topology optimization is the so-called `SIMP` (Solid Isotropic Material with Penalization for intermediate density) method. This belongs to the family of **density-based methods**, where the problem unknown is a scalar field, referred to as density $\rho \in L^{\infty}(\Omega)$, taking values between 0 (absence of material) and 1 (presence of material).
Nevertheless, all the intermediate values between 0 and 1 are allowed because the formulation of the problem require density ρ to be a continuous function (otherwise the problem would not be well defined) and therefore the intermediate values have to be penalized in the formulation. A possibility is to adopt the standard **power law penalization** function ρ^p with, for example, $p \geqslant \max\left(\frac{2}{1-\nu}, \frac{4}{1+\nu}\right)$ like in [8]. In practice, intermediate values of the density are penalized and this is obtained through a suitable penalty exponent, which pushes the density towards either the value zero or one.
The final layout of the structure is obtained by extracting the parts of the domain where the density is 1.
Consequently, the `SIMP` approach applied in the design of lattice structures entails solving the model with a **modified Hooke law**, which replaces **Young's coefficient** E with $\rho^p E$ and consequently, the **Lamé coefficients** λ and μ with $\rho^p \lambda$ and $\rho^p \mu$ respectively.
For example, given a loaded structure $\Omega \in \mathbb{R}^2$, we aim, under a volume constraint, at identifying the optimal topology which minimizes the compliance or, equivalently, maximizes the structure stiffness. In particular, we assume the traction, $\mathbf{f} \in \Gamma_N \to \mathbb{R}^2$, to be applied on a portion Γ_N of

the body boundary $\partial\Omega$. The compliance is given by $\int_{\Gamma_N} \mathbf{f} \cdot \mathbf{u}\ d\gamma$ with $\mathbf{u} = (u_1, u_2)^T : \Omega \to \mathbb{R}^2$ the induced displacement field.

The mathematical model underlying the structure deformation is represented by the **linear elasticity equation**

$$\begin{cases} -\nabla \cdot \sigma(\mathbf{u}) = \mathbf{0} & \text{in } \Omega \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_D \\ \sigma(\mathbf{u}) = \mathbf{f} & \text{on } \Gamma_N \\ \sigma(\mathbf{u}) = \mathbf{0} & \text{on } \Gamma_F \end{cases} \tag{2.1}$$

where $\sigma(\mathbf{u}) = 2\mu\varepsilon(\mathbf{u}) + \lambda \operatorname{tr}(\varepsilon(\mathbf{u}))\, I$ denotes the stress tensor, with $\varepsilon(\mathbf{u}) = \frac{\nabla\mathbf{u} + \nabla\mathbf{u}^T}{2}$ the small displacement strain tensor and $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$ the Lamé coefficients.

We have denote by E the Young's modulus, ν the **Poisson's ratio**, tr is the trace operator, I the identity tensor, $\mathbf{n}$ the unit outward normal vector to $\partial\Omega$, Γ_D the portion of the boundary where the structure is clamped and Γ_F the traction-free boundary.

With a view of the optimization problem, we first provide the weak form of the modified linear elasticity equation:

find $\mathbf{u} \in U = \left\{ \mathbf{v} \in \left(H^1(\Omega)\right)^2 : \mathbf{u} = \mathbf{0} \text{ on } \Gamma_D \right\}$ s.t. $a(\mathbf{u}, \mathbf{v}) = C(\mathbf{v})\ \ \forall \mathbf{v} \in U$

with

$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \sigma_p(\mathbf{u}) : \varepsilon\mathbf{v}\ d\mathbf{x}$, $C(\mathbf{v}) = \int_{\Gamma_N} \mathbf{f} \cdot \mathbf{v}\ d\gamma$ and $\sigma_p(\mathbf{u}) = \rho^p \sigma(\mathbf{u})$.

Notice that $C(\mathbf{u}) = a(\mathbf{u}, \mathbf{u})$ coincides with the compliance to be minimized.

The full `SIMP` methods thus reads

Find ρ s.t.

$\min_{\rho \in L^\infty(\Omega)} C(\mathbf{u}(\rho))$ s.t.

1. $a(\mathbf{u}(rho), \mathbf{v}) = C(\mathbf{v})\ \ \forall \mathbf{v} \in U$
2. $\int_\Omega \rho\ d\mathbf{x} \leqslant \alpha|\Omega|$
3. $\rho_{\min} \leqslant \rho \leqslant 1$

where $\alpha \in (0, 1)$ denotes the maximum allowable volume fraction, with respect to the original volume $|\Omega|$, and $0 < \rho_{\min} < 1$ is a lower bound for the density, which ensures the elasticity system to be well-defined.

The SIMP algorithm is

Algorithm 1: SIMP on a fixed grid

Result: ρ

Input: TOL, kmax, $\rho_{\min}$, α;

Set: $\rho_h^0 \leftarrow 1$, $k \leftarrow 0$, err $\leftarrow 1 + TOL$

while $err > TOL$ & $k < kmax$ **do**

1. $\rho_h^{k+1} \leftarrow$ IPOPT(ρ_h^k, $\rho_{\min}, Mit = 10, \alpha$, ...);
2. $\rho_h^{k+1} \leftarrow$ HELMHOLTZ$\left(\rho_h^{k+1}, \tau\right)$;
3. err $\leftarrow \|\rho_h^{k+1} - \rho_h^k\|_\infty$;
4. $k \leftarrow k + 1$;

end

where HELMHOLTZ$\left(\rho_h^{k+1}\right)$ is the Helmholtz-filtered version of ρ_h (for further details see section (2.1.2)).

The optimization problem is solved via IPOPT (Interior Point OPTimizer) package: a common large-scale nonlinear optimization tool based on the interior point algorithm (see [6]). Both equality and inequality constraints can be tackled via suitable slack variables. Constraints may involve both the control variable (for example, the density) as well as functions of the control variable (for example the total volume of the structure).

The SIMP is known to suffer from two main **drawbacks**:

1. presence of checkerboards, avoidable with the employment of higher order finite elements for the displacement with respect to the density (but it requires higher CPU times);

2. strong mesh dependency since the solution is not unique. This is due to the two-field formulation, involving density and displacement. Certain combinations of finite elements turn out to be unstable, similarly to the two-field pressure-velocity formulation of the Stokes problem. Possible remedies to mesh dependency consist of adding explicit limitations on the allowable density distributions or of filtering the density.

2.1.2 Filters for the optimization procedure

Usually in a TO procedure, to solve some of SIMP drawbacks various **filtering techniques** are adopted. During the fabrication of macro and micro structures, processes may result in (uniformly) too thin (eroded) or too thick (dilated) structures compared to the intended topology. Here are some examples of filters found in literature and implemented in our code.

To make the discrete natured morphology operators applicable to gradient-based optimization they need to be redefined as continuous and differentiable functions. There are different ways to do this. We have found a smoothed **Heaviside** step function defined as

$$\bar{\rho} = 0.5 + \frac{\tanh\left(\beta_H\left(\rho - 0.5\right)\right)}{2\left(\tanh\left(0.5\beta_H\right)\right)} \tag{2.2}$$

Alternatively, in order to enhance the discreteness of the final designs, a hyperbolic tangent threshold projection is employed in [1], corresponding to another smoothed Heaviside function. We will refer to this as **Sigmund** filter and it is defined as:

$$\bar{\rho} = (1 - lb)\,\frac{\tanh\left(\beta\eta\right) + \tanh\left(\beta\left(\rho - lb - \eta\right)\right)}{\tan\left(\beta\eta\right) + \tanh\left(\beta\left(\rho - lb - \eta\right)\right) + lb} \tag{2.3}$$

Sigmund filter has two main input parameters to be tuned: β and η. The bigger the first one, the more the filter sharpens the density since it tends to the real Heaviside function. On the other hand, the second one represents the filtering threshold. Filter in equation (2.3) differ from formula (8) in [1] from the added parameter $lb = \rho_{min}$ (lower bound of the density), which prevents density to reach values under its minimum acceptable value.

An alternative method of filtering the design variable is obtained by averaging it over the neighbor cells. This operation could be obtained as a solution of the **Helmholtz** partial differential problem

$$\begin{cases} -\tau^2\delta\rho_f + \rho_f = \rho_h & \text{on } \Omega \\ \tau^2\nabla\rho_f \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \end{cases} \tag{2.4}$$

with τ real parameter measuring the thickness of the smoothed density.

In general it can be utilized in every optimization problem which needs to be regularized and does not have a defined length scale, i.e discretization leads to a mesh-dependent results.

We recall, for simplicity, those three filters have been identified in our code by those different names

- Heaviside: eq. (2.2)
- Helmholtz: eq. (2.4)
- Sigmund: eq. (2.3)

Filters could be applied inside the optimization procedure or after every `IPOPT` whole cycle; in the first case the density used in the primal and dual problem is always filtered at each `IPOPT` iteration, therefore the gradient of the cost function should be changed according to the filter used; in the second case we apply the filter after the `IPOPT` procedure has ended and then we restart it from the found filtered density.

In general, filters are usually easy for implementation and can help in eliminating **checkerboards effects**. Still, often remains difficulties to boundary identification in post-processing

and moreover the majority of filters needs tuning of their parameters or are problem specific. Another more universal method to overcome the drawbacks of the `SIMP` method is mesh adaptation.

2.2 Anisotropic mesh adaptation

Consider a generic problem in a domain $\Omega \subset \mathbb{R}^2$ for which we look for a weak solution in a functional space V like $H^1(\Omega)$ solving it numerically with FEM.
When we solve numerically a problem like this, one kind of error we introduce is the **discretization error** which depends on the mesh refinement h of the triangulation $\mathcal{T}_h$.
We can estimate the error as $\|e_h\|_V \leqslant S = \sqrt{\sum_{K\in\mathcal{T}_h} \eta_K^2}$ where $\|\cdot\|_V$ is a properly chosen norm on V, S is the **global estimator** of the error, while η_K is the **local estimator** of the error in K. Of course we would like $\|e_h\|_V \sim \tau$ where τ is a **fixed tolerance** and we could be tempted to refine the grid everywhere to obtain this. However, in order to have an efficient grid that minimizes the number of elements necessary to obtain the desired accuracy, we can equidistribute the error on each element of the triangulation $K \in \mathcal{T}_h$ in order to have η_K almost equal in each K.
In order to do this, we need to identify the regions in which the error is higher (due to a more pronounced variability of the solution) and balance it either by a smaller local grid size h_K ($h-$**adaptivity**) or by a higher polynomial degree r ($p-$**adaptivity**). In our work, we focused of $h-$adaptivity.
Since we need to know the error in a region, two approaches can be used. The first one is to use an **a priori error estimate**, by replacing the exact solution with a well-chosen approximation, easily computable on each single element, so we talk about a priori adaptivity.
A second approach is instead base on the use of an **a posteriori error estimate**, able to link the approximation error to the behaviour of the approximates numerical solution, known after solving the problem numerically. In such case, the optimal computational grid will be constructed through an iterative process where solution, error estimate and modification of the computational grid are recomputed until reaching the requested accuracy. This second approach, which is the one we used, is called a posteriori adaptivity.
A good estimator of the error not only need to be computable on each single element, but should also be robust, namely s.t.

- $\exists C_1 \sim 1 : C_1\|e_h\|_V \leqslant S$
- $\exists C_2 \sim 1 : S \leqslant C_2\|e_h\|_V$

When we have an estimator of the error, we can chose different strategy for the adaptivity procedure, for example

- fixed a number N_h of elements we can create the grid which minimizes the error estimate S
- fixed the desired tolerance τ we can create the grid with the minimum number of elements s.t. $S \sim \tau$
- fixed a number N_h of elements or the desired tolerance τ we create a grid s.t. $\eta_K = \frac{\tau}{\sqrt{N_h}}$ for each K for example
  - refining the triangles K for which $\eta_K \geqslant C\frac{\tau}{\sqrt{N_h}}$
  - unrefining the triangles K for which $\eta_K < C\frac{\tau}{\sqrt{N_h}}$

In our work, the algorithm we used was already implemented in a library and it uses the **remeshing strategy**, in which the grid is recreated every time using a metric.
Moreover, we can also use **anisotropic meshes**, in which the triangles can be very stretched: this allows to sharply capture the solution where it has strong gradient.
The anisotropic properties of a generic triangle $K \in \mathcal{T}_h$ are extracted out of the spectral properties of the standard affine map T_k between the equilateral reference element $\hat{K}$ inscribed in the unit circle and K, i.e. $\mathbf{x} = T_K(\hat{\mathbf{x}}) = M_K\hat{\mathbf{x}} + \mathbf{t}_K$ where $\mathbf{x} \in K$, $\hat{\mathbf{x}} \in \hat{K}$, $M_K \in \mathbb{R}^{2\times 2}$, $\mathbf{t}_K \in \mathbb{R}^2$. In particular, we factorize the Jacobian M_K through the polar decomposition $M_K = B_K Z_K$ with $B_K \in \mathbb{R}^{2\times 2}$ a SPD matrix stretching the triangle K and $Z_K \in \mathbb{R}^{2\times 2}$ an orthogonal matrix which rotates K. Then, we introduce the classical eigenvalue-eigenvector factorization of B_K as $B_K = R_K^T \Lambda R_K$ with $R_K^T = [\mathbf{r}_{1,K}, \mathbf{r}_{2,K}]$ and $\Lambda_K = \mathrm{diag}(\lambda_{1,K}, \lambda_{2,K})$ with $\lambda_{1,K} \geqslant \lambda_{2,K} > 0$. The eigenvectors $\mathbf{r}_{1,K}, \mathbf{r}_{2,K}$ identify the directions of the semi-axes of the ellipse circumscribed to K, while the eigenvalues $\lambda_{1,K}, \lambda_{2,K}$ measures the length of these semi-axes. A measure of the anistropic deformation is provided by the aspect ration of the element K defined as $s_K = \frac{\lambda_{1,K}}{\lambda_{2,K}} \geqslant 1$: the higher s_K, the larger the deviation from the equilateral shape, for which $s_K = 1$.
Since the unknown density function has very strong gradient across the material-void interfaces, anisotropic mesh is preferred to the isotropic one and it has been proven is effectiveness, for example, in reducing the computational costs and in getting rid of the checkerboard phenomenon, giving better results then the common filters used (ref. [8]).

2.2.1 The SIMPATY method (SIMP with AdaptiviTY)

The latter issues we have with `SIMP` suggest that can be useful to use mesh adaptivity techniques. At every iteration the mesh is changed according to the solution found in the previous

steps. The choice on how to change each element is done according to a posteriori error estimate; in our case, we resort to a **recovery based error estimator** which employs the density as driving quantity (ref. [7]).

As a consequence, an alternative to the `SIMP` method is given by the `SIMPATY` method (SIMP with AdaptiviTY) which alternate a mesh-adaptivity procedure to the optimization through `IPOPT`. This time, the stopping criterion is based on the **mesh cardinality**.

This is the `SIMPATY` algorithm.

Algorithm 2: SIMPATY: SIMP with AdaptiviTY

Result: ρ

Input: CTOL, MTOL, kmax, $\rho_{\min}$, $\mathcal{T}_h^0$, α;

Set: $\rho_h^0 \leftarrow 1$, $k \leftarrow 0$, errC $\leftarrow 1 + TOL$

while $errC > TOL$ & $k < kmax$ **do**

1. $\rho_h^{k+1} \leftarrow \text{IPOPT}\big(\rho_h^k, \rho_{\min}, Mit = nk, \alpha, ...\big)$;
2. $\mathcal{T}_h^{k+1} \leftarrow adaptmesh\left(\mathcal{T}_h^k, \rho_h^{k+1}, MTOL, method\right)$;
3. errC $\leftarrow |\#\mathcal{T}_h^{k+1} - \#\mathcal{T}_h^k| / \#\mathcal{T}_h^k$;
4. $k \leftarrow k + 1$;

end

2.3 Homogenization technique to design lattice materials

The homogenization method is an **asymptotic technique** whose goal is to assign macroscopic effective properties to microscopic entities, which are arranged periodically. This approach plays a crucial role in multiscale simulations since it allows one to deal with the macroscale only, the effects of the microscale being inherited through homogenization.

In other words, homogenization extracts homogeneous effective parameters from disordered or heterogeneous media. Starting from a microscopic description of a problem, we seek a macroscopic, or effective, description. This process of making an asymptotic analysis and seeking an averaged formulation is called homogenization.

In this section, we analyze also the **converse technique**, known as inverse homogenization. This can be formulated as a control problem or, specifically, as a TO problem on a unit cell. The aim is to find the optimal arrangement of material at the microscale so that desired effective properties are guaranteed at the macroscale. Notice that the flow of information is opposite with respect to the classical homogenization. The macroscale is fixed or prescribed, whereas the microscale (the unit cell) is modified to match the desired requirements.

2.3.1 Direct homogenization

Direct homogenization has been employed in different fields of application to modify the macroscale model according to the microscale layout. This technique relies on the periodic arrangement of a microstructure which constitutes the base cell, Y. Such elementary entity represents the domain of interest and it is analyzed in order to retrieve its effect on the macroscale.

With a particular focus on the linear elastic problem, it is observed that the stiffness of an optimal designed structure, subject to given loads and constraints, is increased by inserting small substructures. Consequently, different authors have investigated the possibility of employing topology optimization at a microscale as well, aiming at yielding optimized microstructures (metamaterials). The ultimate goal is to combine the microscopic optimized structures with a standard TO performed at the macroscale. This link is made possible by employing homogenization techniques, which are widely used to incorporate the information provided by the microscale into macroscale models.

2.3.2 Inverse homogenization

We refer to inverse homogenization as to the procedure concerning the design of a base cell Y, whose contribution to the macroscale is prescribed, according to the direct homogenization process in the previous section.

In order to modify the formulation of the direct method, we have to account for variations in the initial distribution of material in the base cell. This goal can be pursued via TO, yielding optimized structures according to specific, user-defined, constraints and objectives.

In the next section we show an example of application of the inverse homogenization technique and the other mathematical tools cited before in the design of a lattice material following a linear elastic law.

2.4 Design lattices with linear elastic law

The linear elasticity law in eq. (2.1) can be rewritten through the use of tensors that facilitates the application of the homogenization technique. We stick to the convention of denoting by E_{ijkl} the fourth-order stiffness tensor, so that the stress tensor σ can be written component wise as

$$\sigma_{ij} = E_{ijkl}\varepsilon_{kl} \tag{2.5}$$

where the stiffness tensor in two dimensions E for isotropic materials is

$$E = \begin{bmatrix} E_{1111} & E_{1122} & 0 \\ E_{1122} & E_{2222} & 0 \\ 0 & 0 & E_{1212} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix} \tag{2.6}$$

and the component of the strain tensor ε are

$$\varepsilon_{kl} = \frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right) \tag{2.7}$$

where x_l, with $l = 1, 2$, are the spatial coordinates.

The displacement field $\mathbf{u}$ is constrained to solve this equations, that constitutes the governing static equilibrium of the system.

In [10] is shown an example of the design of compliant mechanism and material structure using a numerical topology optimization method joint with inverse homogenization and mesh adaptation. In particular, the aim of the structure is to have a negative Poisson's ratio ν. For the design of NPR materials (materials with Negative Poisson's Ratio), are considered microstructures, where the smallest repetitive unit, called the base cell, will be the design domain, and the design goal is to minimize the error in obtaining the prescribed elastic properties for a fixed amount of material in the base cell.

The homogenization technique relies on the repetition of the base cell, therefore in order to preserve this physical feature, we impose periodic boundary conditions. In this way, we enforce that the displacement field $\mathbf{u}$ is equal in correspondence with opposite boundaries.

The actual objective becomes to compute the homogenized (or effective) stiffness tensor, E^H, representing a macroscopic mean value of the tensor E, after neglecting the microscale fluctuations E^*. To this end, we resort to an asymptotic expansion of the displacement field $\mathbf{u}$ with respect to the base cell size, considering only the first two terms. Then it can be shown that the homogenized tensor E^H is given by

$$E^H_{ijkl} = \frac{1}{|Y|}\int_Y E_{ijpq}\left(\varepsilon^{0,kl}_{pq} - \varepsilon^{*,kl}_{pq}\right)\, dY$$

where $|Y|$ is the measure of the cell Y, $\varepsilon_{0,kl}$ identifies a fixed strain field, chosen among the four linearly independent possible fields (k, l being equal to $1, 2$), while $\varepsilon_{*,kl}$ is the $Y-$periodic fluctuation strain, i.e., the weak solution to the equation

$$\int_Y E_{ijpq}\varepsilon^{*,kl}_{pq}\varepsilon_{ij}(v)\, dY = \int_Y E_{ijpq}\varepsilon^{0,kl}_{pq}\varepsilon_{ij}(v)\, dY \;\; \forall v \in V \tag{2.8}$$

being $V \subset (H^1(Y))^4$ a periodic Sobolev function space.

Thus, by combining the latter two equations, we obtain the final form of the effective stiffness tensor

$$E^H_{ijkl} = \frac{1}{|Y|}\int_Y E_{pqrs}\left(\varepsilon^{0,kl}_{pq} - \varepsilon^{*,kl}_{pq}\right)\left(\varepsilon^{0,ij}_{rs} - \varepsilon^{*,ij}_{rs}\right)\ dY \tag{2.9}$$

This equation, with the previous one, constitute the state equations to be employed in the inverse homogenization technique, as detailed in the following section.

Let us fix the objective function, J, as a control over the quadratic deviation between the computed value of the homogenized stiffness tensor, E^H, and the requested one, E^W , i.e., $J = \sum_{ijkl}\left(E^H_{ijkl}(\rho) - E^W_{ijkl}\right)^2$.

Hence, the minimization of J should lead to a micro-design, whose macrofeatures are the ones desired by the user. Thus, the final system for the micro-optimization is obtained by solving the following problem:

Find ρ such that

$\min_{\rho\in L^\infty(Y)} J(\rho)$ is achieved under constraints

1. $(2.8)_\rho$and $(2.9)_\rho$ are satisfied
2. + Periodicty conditions
3. $\int_Y \rho\ dY \leqslant \alpha|Y|$
4. $\rho_{\min} \leqslant \rho \leqslant 1$

Chapter 3

Design lattices with nonlinear elastic law

Our main goal in this work has been to solve a TO problem, using inverse homogenization and mesh adaptation techniques, applied to a lattice material in nonlinear regime.
Homogenization methods like the one treated in section 2.4, known as asymptotic approach (ref. [3]), can be used to calculate the effective material properties mainly only in the small deformation regime. However, when the material undergoes finite deformation, i.e. in nonlinear regime, the material properties will be strongly dependent on the deformation and the effective material properties have to be calculated for each deformation state.
The mathematical model underlying the structure deformation in a nonlinear regime is derived through the minimization (for simplicity, setting at zero) the total elastic energy of the system, following a more energy-based approach. In this work therefore we decided to adopt a total Lagrangian formulation.
In order to have a reference solution from [1] and a partially clear way to treat it, our work focused on tensile deformation test in x direction. In a tensile test, materials are uniformly stretched either longitudinally or transversely. The material behaviour can be characterized using the unit cell subjected to boundary displacements. As shown in fig. (3.1), the unit cell is fixed at the lower left corner, symmetric boundary conditions are employed for each node pair on the left side and for the bottom side, a constant displacement difference $d_1 = 0.3$ is assumed longitudinally on the right side and free movement (homogeneous Neumann condition) is imposed on the top side.

$$\begin{cases} \boldsymbol{u} \cdot \boldsymbol{n} = u_1 & = 0 \quad \text{on } \Gamma_1 \\ \boldsymbol{u} \cdot \boldsymbol{n} = u_2 & = 0 \quad \text{on } \Gamma_4 \\ u_1 & = d_1 \quad \text{on } \Gamma_2 \\ \frac{\partial \Psi(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \boldsymbol{n} & = 0 \quad \text{on } \Gamma_3 \end{cases} \tag{3.1}$$

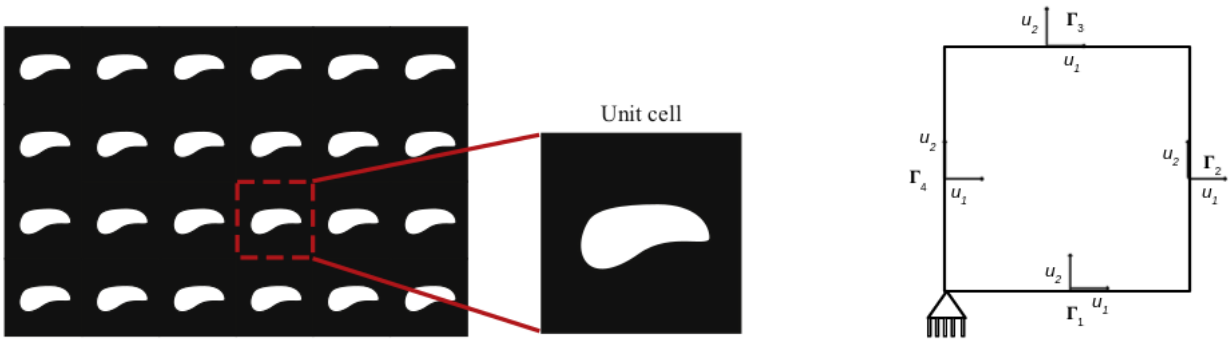


Figure 3.1: Schematic illustration of BC of a unit cell [1]

3.1 Lagrange multiplier approach

We start to define our TO problem and Interior Point OPTimizer (IPOPT) input function from following the Lagrange functional:

$$\mathscr{L}(\rho, \boldsymbol{u}, \boldsymbol{\lambda}) = J(\boldsymbol{u}) + \int_Y \rho^p \frac{\partial \Psi(\boldsymbol{u})}{\partial \boldsymbol{u}} \boldsymbol{\lambda} \, dY \tag{3.2}$$

where

- $(\rho, \boldsymbol{u}, \boldsymbol{\lambda})$ are three independent variables representing:
  - ρ the design variable, the density characterizing the unit cell Y
  - $\boldsymbol{u} = [u_1, u_2]$ the displacement field
  - $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]$ the Lagrange multiplier
- $J(\boldsymbol{u})$ is the proper cost function we would like to minimize to obtain the desired features; in our problem it will be the squared error between the actual and prescribed Poisson's ratio under certain tensile test. Therefore

$$J(\boldsymbol{u}) = \frac{1}{2}(c(\boldsymbol{u}) - c^*)^2 = \frac{1}{2}(\nu(\boldsymbol{u}) - \nu^*)^2$$

- $\Psi(\boldsymbol{u},\rho)$ is a proper strain energy function describing the material behaviour. Under a linear regime $\Psi(\boldsymbol{u},\rho) = \sigma(\boldsymbol{u},\rho) : \varepsilon(\boldsymbol{u})$, representing the classical Kirchhoff- St.Venant strain energy (2.5). This law fails in compression or tensile deformation, therefore for nonlinear materials there have been defined different explicit functions of Ψ, according to the specific case under investigation. In our study we have used $\Psi(\rho,\boldsymbol{u}) = \frac{\lambda_L}{2}(J-1)^2 + \mu E_{ij}E_{ij}$ to circumvent the drawbacks of the Kirchhoff-St.Venant model under excessive compression (formula (6d) from ref.[2]). The other terms characterizing this strain energy are:
  - E_{ij} the Green-Lagrange strain tensor: $E_{ij} = \frac{1}{2}(F_{ki}F_{kj} - \delta_{ij})$
  - J the determinant of the deformation gradient F: $F_{ij} = \frac{\partial x_i}{\partial X_j} = \delta_{ij} + \frac{\partial u_i}{\partial x_j}$
  - the design variable with penalty coefficient ρ^p is implied in the Young's coefficient E inside the Lamè coefficients λ_L and μ

By requiring that the (Gateaux) derivatives of $\mathscr{L}$ with respect to the variables $\rho, \boldsymbol{u}, \boldsymbol{\lambda}$ evaluated at the solution $\left(\hat{\rho}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\lambda}}\right)$ vanish for any test function properly chosen, we obtain the state equation (corresponding to (2.8) state equation for the linear regime), the adjoint equation and the gradient of the cost function we wanted to minimized in the optimization problem.

3.1.1 State equation: primal problem

To retrieve the state equation, we imposed $\frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}}\left(\hat{\rho}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\lambda}}\right)\boldsymbol{\varphi} = 0 \ \forall \boldsymbol{\varphi} \in V$; explicitly the variational formulation of the problem is
Find $\hat{\boldsymbol{u}} \in V$ such that

$$\begin{aligned}\int_Y \hat{\rho}^p \frac{\partial \Psi(\hat{\boldsymbol{u}})}{\partial \boldsymbol{u}}\boldsymbol{\varphi} dY = \int_Y \hat{\rho}^p \frac{\partial \Psi}{\partial E_{ij}}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} = \int_Y \hat{\rho}^p \sum_{i,j}\left[\lambda_L (J-1) J C_{ij}^{-1}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\psi} + 2\mu E_{ij}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\right] dY = \\ = \int_Y \hat{\rho}^p \lambda_L (J-1) J\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\right] \\ + \hat{\rho}^p 2\mu\left[E_{11}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} + E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\right] dY = 0 \ \forall \boldsymbol{\varphi} \in V\end{aligned}$$

where

$$\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\varphi} = \sum_k \frac{1}{2}\left(\frac{\partial u_k}{\partial x_i}\frac{\partial \varphi_k}{\partial x_j} + \frac{\partial u_k}{\partial x_j}\frac{\partial \varphi_k}{\partial x_i} + \frac{\partial \varphi_k}{\partial x_i}\delta_{kj} + \frac{\partial \varphi_k}{\partial x_j}\delta_{ki}\right) \tag{3.3}$$

This nonlinear problem has been solved using Newton's method. Rewriting the problem as
Find $\boldsymbol{u}$ such that $F(\boldsymbol{u}) = 0$, the Newton's method is

Given $\boldsymbol{u}^0 \ \forall \ k = 0, 1, ...$ up to convergence, solve:

$$\begin{cases} F'\left(\boldsymbol{u}^k\right)\boldsymbol{\delta u} = -F\left(\boldsymbol{u}^k\right) \\ \boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{\delta u} \end{cases}$$

Therefore the tangent problem applied to our primal problem is
Given $\boldsymbol{u}^0 \ \forall \ k = 0, 1, ...$ up to convergence, solve:

$$\begin{cases} \frac{\partial}{\partial \boldsymbol{u}}\left[\int_Y \hat{\rho}^p \frac{\partial \Psi\left(\boldsymbol{u}^k\right)}{\partial \boldsymbol{u}} \boldsymbol{\delta u} \ dY\right]\boldsymbol{\varphi} = -\int_Y \hat{\rho}^p \frac{\partial \Psi\left(\boldsymbol{u}^k\right)}{\partial \boldsymbol{u}} \boldsymbol{u}^k \boldsymbol{\varphi} dY \quad \forall \ \boldsymbol{\varphi} \ \in V \\ \boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{\delta u} \end{cases} \tag{3.4}$$

As initial guess $\boldsymbol{u}^0$ for the Newton iterations has been chosen a velocity field that fulfills the Dirichlet boundary condition due to the tensile test, i.e. $\boldsymbol{u}^0 = [d_1 \cdot x, 0]$. As a consequence in the variational problem (3.4) we have to impose

- homogeneous Dirichlet boundary condition $[\delta u_1, \delta u_2] = [0, 0]$ on Γ_2
- symmetry boundary conditions $\boldsymbol{\delta u} \cdot \boldsymbol{n} = 0$ on $\Gamma_1 \cup \Gamma_4$
- Neumann homogeneous boundary condition on Γ_3

As convergence criterion was chosen a tolerance $tol = 10^{-12}$ and a maximum number of iterations $loopmax = 100$.

The right hand side of the weak formulation of the tangent problem in (3.4) is

$$
\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{u}}\left(\int_Y \hat{\rho}^p \frac{\partial \boldsymbol{\Psi}\left(\boldsymbol{u}^k\right)}{\partial \boldsymbol{u}} \boldsymbol{\delta u}\, dY\right)\boldsymbol{\varphi} = \\
&= \int_Y \hat{\rho}^p \frac{\partial}{\partial \boldsymbol{u}}\left[\lambda_L\left(J-1\right)J\left(C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right)\right. \\
&\quad \left. + 2\mu_L\left(E_{11}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right)\right]\boldsymbol{\varphi} dY \\
&= \int_Y \hat{\rho}^p \left\{\left[\lambda_L \frac{\partial\left(J-1\right)J}{\partial \boldsymbol{u}}\right]\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right]\right. \\
&\quad + \lambda_L\left(J-1\right)J\frac{\partial}{\partial \boldsymbol{u}}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right] \\
&\quad \left. + 2\mu_L\frac{\partial}{\partial \boldsymbol{u}}\left[E_{11}\frac{\partial E_{11}}{\partial u}\boldsymbol{\delta u} + E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta} u + E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}} \;+ E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta} u\right]\right\}\boldsymbol{\varphi} dY \\
&= \int_Y \hat{\rho}^p \lambda_L \frac{\partial\left[J^2-J\right]}{\partial J}\frac{\partial J}{\partial \boldsymbol{u}}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right]\boldsymbol{\varphi} \\
&\quad + \hat{\rho}^p \lambda_L\left(J-1\right)J\left[\frac{\partial C_{11}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{11}^{-1}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial C_{12}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{12}^{-1}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right. \\
&\quad \left. + \frac{\partial C_{21}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{21}^{-1}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial C_{22}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{22}^{-1}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right] \\
&\quad + \hat{\rho}^p 2\mu_L\left[\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{11}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{12}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right. \\
&\quad \left. + \frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{21}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{22}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right]\, dY \\
&= \int_Y \hat{\rho}^p \lambda_L\left(2J-1\right)\frac{\partial}{\partial \boldsymbol{u}}\left[F_{11}F_{22}-F_{12}F_{21}\right]\boldsymbol{\varphi}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\right] + \\
&\quad + \hat{\rho}^p \lambda_L\left(J-1\right)J\left[\frac{\partial C_{11}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{11}^{-1}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial C_{12}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{12}^{-1}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right. \\
&\quad \left. + \frac{\partial C_{21}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{21}^{-1}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial C_{22}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u}\boldsymbol{\varphi} + C_{22}^{-1}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right] \\
&\quad + \hat{\rho}^p 2\mu_L\left[\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{11}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{12}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right. \\
&\quad \left. + \frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{21}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} + \frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} + E_{22}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi}\right]\, dY
\end{aligned}
$$

where in detail:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{u}}\left[F_{11}F_{22}-F_{12}F_{21}\right]\boldsymbol{\varphi} &= \left[\frac{\partial F_{11}}{\partial \boldsymbol{u}}F_{22}+\frac{\partial F_{22}}{\partial \boldsymbol{u}}F_{11}-\frac{\partial F_{12}}{\partial \boldsymbol{u}}F_{12}-\frac{\partial F_{21}}{\partial \boldsymbol{u}}F_{12}\right]\boldsymbol{\varphi} \\
&= \frac{\partial \varphi_1}{\partial x_1}F_{22}+\frac{\partial \varphi_2}{\partial x_2}F_{11}-\frac{\partial \varphi_1}{\partial x_2}F_{21}-\frac{\partial \varphi_2}{\partial x_1}F_{12} \\
\frac{\partial C_{ij}^{-1}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} &= \sum_{k,l}\frac{\partial C_{ij}^{-1}}{\partial E_{kl}}\frac{\partial E_{kl}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} = \sum_{k,l}-D_{ijkl}\frac{\partial E_{kl}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\boldsymbol{\delta u} \\
D_{ijkl} &= -\frac{\partial C_{ij}^{-1}}{\partial E_{kl}} = C_{ik}^{-1}C_{jl}^{-1}+C_{il}^{-1}C_{jk}^{-1} \\
\frac{\partial^2 E_{ij}}{\partial \boldsymbol{u}^2}\boldsymbol{\delta u}\boldsymbol{\varphi} &= \sum_k \frac{1}{2}\left(\frac{\partial \varphi_k}{\partial x_i}\frac{\partial \delta u_k}{\partial x_j}+\frac{\partial \varphi_k}{\partial x_j}\frac{\partial \delta u_k}{\partial x_i}\right)
\end{aligned}$$

and refer also to formula (3.3)

The left hand side of the weak formulation of the tangent problem in (3.4) is

$$\begin{aligned}
-\int_Y \hat{\rho}^p \frac{\partial \Psi\left(\boldsymbol{u}^k\right)}{\partial \boldsymbol{u}}\boldsymbol{\varphi}dY = &-\int_Y \hat{\rho}^p \lambda_L\left(J-1\right)J\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\right] \\
&+\hat{\rho}^p 2\mu\left[E_{11}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}+E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\varphi}\right]dY
\end{aligned}$$

All previous tensors and derivatives have to be evaluated at the previous time step solution $\boldsymbol{u}^k$. In the course of the derivation of the variational problem some formulas from [2] have been taken such as (8), (9d) and (14) and in a validation phase have been verified (section 3.4.1)

3.1.2 Adjoint equation: dual problem

The adjoint equation in the optimization procedure has been derived imposing $\frac{\partial \mathscr{L}}{\partial \boldsymbol{u}}\left(\hat{\rho}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\lambda}}\right)\boldsymbol{\psi} = 0 \ \forall \boldsymbol{\psi} \in V$

Explicitly the dual problem is

Find $\hat{\boldsymbol{\lambda}} \in V$ such that

$$\frac{\partial}{\partial \boldsymbol{u}}\left(\int_Y \rho^p \frac{\partial \boldsymbol{\psi}\left(\hat{\boldsymbol{u}}\right)}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\, dY\right)\boldsymbol{\psi}+\frac{\partial J\left(\boldsymbol{u}\right)}{\partial \boldsymbol{u}}\boldsymbol{\psi} = 0 \quad \forall \boldsymbol{\psi} \in V \tag{3.5}$$

Since the Poisson's ratio is computed with

$$\nu_{12} = -\frac{\int_{\Gamma_3} u_2 - \int_{\Gamma_1} u_2}{d_1} \tag{3.6}$$

where d_1 is the traction imposed at the right boundary and $u_2 = 0$ *on* Γ_1 because of symmetry

conditions:

$$\frac{\partial J\left(\boldsymbol{u}\right)}{\partial \boldsymbol{u}}\boldsymbol{\psi}=\frac{\partial}{\partial \boldsymbol{u}}\left[\frac{1}{2}\left(\nu_{12}-\nu^{*}\right)^{2}\right]\boldsymbol{\psi}=\left(\nu_{12}-\nu^{*}\right)\frac{\partial}{\partial \boldsymbol{u}}\left[-\frac{\int_{\Gamma_3}u_2}{d_1}\right]\boldsymbol{\psi}=\frac{-\left(\nu_{12}-\nu^{*}\right)}{d_1}\int_{\Gamma_3}\psi_2$$

The other term of the weak formulation of the dual problem is found deriving the energy similarly as done in the **Newton's method**.

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{u}}\left(\int_Y\frac{\partial\boldsymbol{\psi}\left(\hat{\boldsymbol{u}}\right)}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\,dY\right)\boldsymbol{\psi}=\\
&=\int_Y\hat{\rho}^{p}\frac{\partial}{\partial u}\left[\lambda_L\left(J-1\right)J\left(C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right)\right.\\
&\quad\left.+2\mu_L\left(E_{11}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right)\right]\boldsymbol{\psi}dY\\
&=\int_Y\hat{\rho}^{p}\left\{\left[\lambda_L\frac{\partial\left(J-1\right)J}{\partial \boldsymbol{u}}\right]\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right]\right.\\
&\quad+\lambda_L\left(J-1\right)J\frac{\partial}{\partial \boldsymbol{u}}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right]\\
&\quad\left.+2\mu_L\frac{\partial}{\partial \boldsymbol{u}}\left[E_{11}\frac{\partial E_{11}}{\partial u}\lambda+E_{12}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{21}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\ +E_{22}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right]\right\}\boldsymbol{\psi}dY\\
&=\int_Y\hat{\rho}^{p}\lambda_L\frac{\partial\left[J^2-J\right]}{\partial J}\frac{\partial J}{\partial \boldsymbol{u}}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right]\boldsymbol{\psi}\\
&\quad+\hat{\rho}^{p}\lambda_L\left(J-1\right)J\left[\frac{\partial C_{11}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{11}^{-1}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial C_{12}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{12}^{-1}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right.\\
&\quad\left.+\frac{\partial C_{21}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{21}^{-1}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial C_{22}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{22}^{-1}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right]\\
&\quad+\hat{\rho}^{p}2\mu_L\left[\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{11}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{12}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right.\\
&\quad\left.+\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{21}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{22}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right]\ dY\\
&=\int_Y\hat{\rho}^{p}\lambda_L\left(2J-1\right)\frac{\partial}{\partial \boldsymbol{u}}\left[F_{11}F_{22}-F_{12}F_{21}\right]\boldsymbol{\psi}\left[C_{11}^{-1}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{12}^{-1}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{21}^{-1}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+C_{22}^{-1}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\right]+\\
&\quad+\hat{\rho}^{p}\lambda_L\left(J-1\right)J\left[\frac{\partial C_{11}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{11}^{-1}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial C_{12}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{12}^{-1}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right.\\
&\quad\left.+\frac{\partial C_{21}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{21}^{-1}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial C_{22}^{-1}}{\partial \boldsymbol{u}}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+C_{22}^{-1}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right]\\
&\quad+\hat{\rho}^{p}2\mu_L\left[\frac{\partial E_{11}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{11}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{11}\frac{\partial^2 E_{11}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial E_{12}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{12}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{12}\frac{\partial^2 E_{12}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right.\\
&\quad\left.+\frac{\partial E_{21}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{21}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{21}\frac{\partial^2 E_{21}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}+\frac{\partial E_{22}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{22}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}+E_{22}\frac{\partial^2 E_{22}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}\right]\ dY
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{u}}\left[F_{11}F_{22}-F_{12}F_{21}\right]\boldsymbol{\psi} &= \left[\frac{\partial F_{11}}{\partial \boldsymbol{u}}F_{22}+\frac{\partial F_{22}}{\partial \boldsymbol{u}}F_{11}-\frac{\partial F_{12}}{\partial \boldsymbol{u}}F_{12}-\frac{\partial F_{21}}{\partial \boldsymbol{u}}F_{12}\right]\boldsymbol{\psi} \\
&= \frac{\partial \psi_1}{\partial x_1}F_{22}+\frac{\partial \psi_2}{\partial x_2}F_{11}-\frac{\partial \psi_1}{\partial x_2}F_{21}-\frac{\partial \psi_2}{\partial x_1}F_{12} \\
\frac{\partial C_{ij}^{-1}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}} &= \sum_{k,l}\frac{\partial C_{ij}^{-1}}{\partial E_{kl}}\frac{\partial E_{kl}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}} = \sum_{k,l}-D_{ijkl}\frac{\partial E_{kl}}{\partial \boldsymbol{u}}\boldsymbol{\psi}\frac{\partial E_{ij}}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}} \\
D_{ijkl} &= -\frac{\partial C_{ij}^{-1}}{\partial E_{kl}} = C_{ik}^{-1}C_{jl}^{-1}+C_{il}^{-1}C_{jk}^{-1} \\
\frac{\partial^2 E_{ij}}{\partial \boldsymbol{u}^2}\hat{\boldsymbol{\lambda}}\boldsymbol{\psi} &= \sum_k \frac{1}{2}\left(\frac{\partial \psi_k}{\partial x_i}\frac{\partial \hat{\lambda_k}}{\partial x_j}+\frac{\partial \psi_k}{\partial x_j}\frac{\partial \hat{\lambda_k}}{\partial x_i}\right)
\end{aligned}$$

3.1.3 Functional gradient: gradJ

For the optimization procedure, in particular for the `IPOPT` function used in our code, it is necessary to compute the derivative of the Lagrange functional with respect to the control variable, namely with respect to the density ρ, and then evaluate it at the degrees of freedom of the finite element space of the density:

$$\frac{\partial \mathscr{L}\left(\hat{\rho},\hat{\boldsymbol{u}},\hat{\boldsymbol{\lambda}}\right)}{\partial \rho}\left(\hat{\rho},\hat{\boldsymbol{u}},\hat{\boldsymbol{\lambda}}\right)v = \int_Y p\hat{\rho}^{p-1}v\frac{\partial \Psi\left(\hat{\boldsymbol{u}}\right)}{\partial \boldsymbol{u}}\hat{\boldsymbol{\lambda}}dY \tag{3.7}$$

In some attempts of our optimization procedure, we have used a specific filter on the design variable inside each optimization iteration of `IPOPT` algorithm, therefore it would influence the evaluation of the cost function J (for further details see section 2.1.2).
For example if density is filtered from ρ to $\bar{\rho}$ using the function

$$\bar{\rho} = \frac{\tanh\left(\beta\eta\right)+\tanh\left(\beta\left(\rho-\eta\right)\right)}{\tanh\left(\beta\eta\right)+\tanh\left(\beta\left(1-\eta\right)\right)}$$

then instead of $p\rho^{p-1}v$ inside problem (3.7) we have to use

$$p\bar{\rho}^{p-1}\frac{\partial \bar{\rho}}{\partial \rho}v = p\bar{\rho}^{p-1}\frac{1}{\tanh\left(\beta\eta\right)+\tanh\left(\beta\left(1-\eta\right)\right)}\frac{\beta}{\cosh\left(\beta\left(\hat{\rho}-\eta\right)\right)^2}v$$

3.2 Final algorithm

Summing up, our optimization procedure reads like the following.

Algorithm 3: Final Algorithm

Result: ρ

Input: CTOL, MTOL, maxit, $\rho_{\min}$, $\mathcal{T}_h^0$, α;

Set: ρ_h^0, $k \leftarrow 0$, errC $\leftarrow 1 + TOL$

while $errC > TOL$ & $k < maxit$ **do**

1. $p^{k+1} \leftarrow p_{\max} - 2e^{-k/2}$
2. $\rho_h^{k+1} \leftarrow \text{IPOPT}\big(\rho_h^k, \rho_{\min}, Mit = IPOPTmaxiter, tol = IPOPTtol, ...\big)$;
3. $\rho_h^{k+1} \leftarrow \text{HELMHOLTZ}\big(\rho_h^{k+1}, \tau\big)$;
4. $\mathcal{T}_h^{k+1} \leftarrow adaptmesh\,\big(\mathcal{T}_h^k, \rho_h^{k+1}, MTOL, method\big)$;
5. $\rho_h^{k+1} \leftarrow \text{HEAVISIDE}\big(\rho_h^{k+1}, \beta_H\big)$;
6. $\rho_h^{k+1} \leftarrow \text{SIGMUND}\big(\rho_h^{k+1}, \beta = \beta^k, \eta = \eta^k\big)$;
7. errC $\leftarrow |\#\mathcal{T}_h^{k+1} - \#\mathcal{T}_h^k|/\#\mathcal{T}_h^k$;
8. $k \leftarrow k + 1$;

end

where

- `IPOPT` algorithm automatically solves many time the primal problem (using Newton's method to handle nonlinearity) and one time the dual problem for each iteration. It also includes and enforces the constraints on volume and on ρ and it stops or because of convergence (the target Poisson coefficient is reached) or because the maximum number of iterations is reached. In any case, as described previously, we need to apply some filters to make the density sharper and usable in practice, so we need to restart `IPOPT` from the new density filtered a fixed number of times corresponding to `maxit`;
- the filters (Helmholtz problem, Heaviside function and hyperbolic tangent as in [1]) can be used, as for mesh adaptation, at each macro iteration, but may be more convenient not to use always all of them, the details on this part is in section 3.4;
- the previous chances, with other parameters, like the filter ones, the initial density ρ_0, the allowable volume fraction, the penalty law and so on, need to be properly tuned in order to obtain a physically reasonable structure.

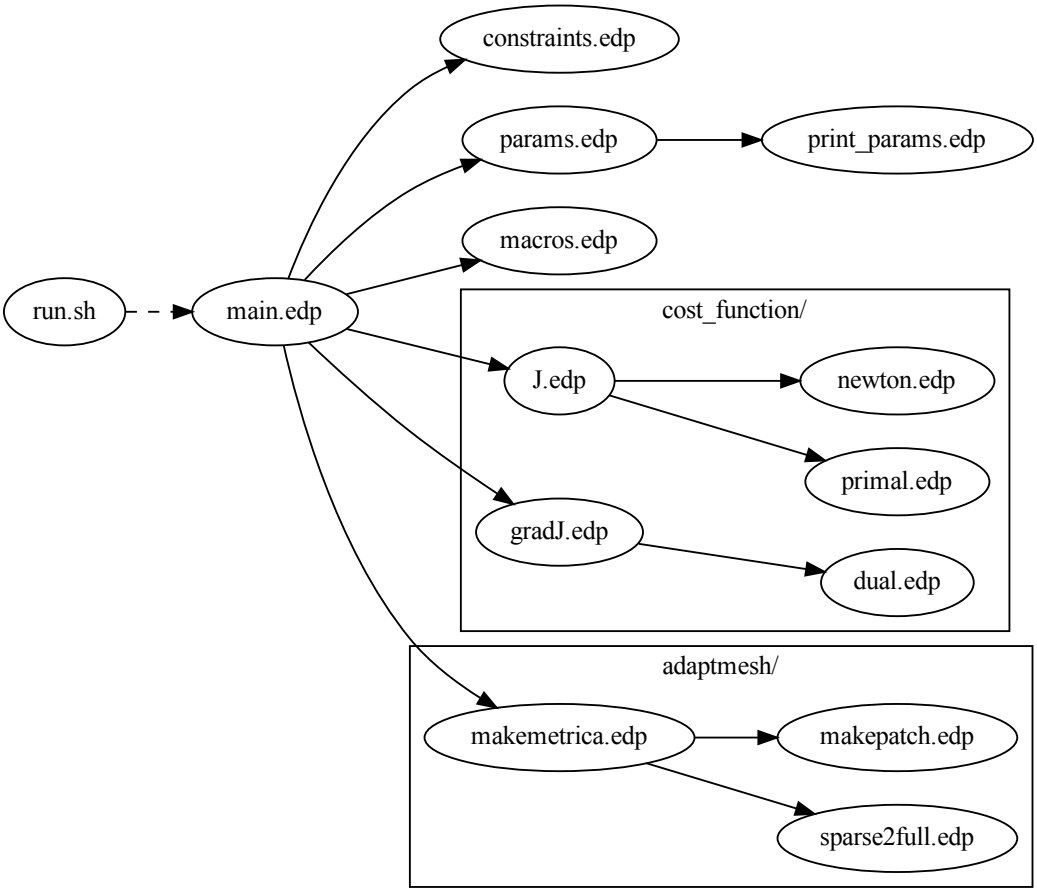
For all this tuning phase more details are in section 3.4, while in the next section we go further into the implementation of the method.

3.3 Implementation in FreeFem++

For the implementation we used `FreeFem++`, which is an open source programming language based on `C++`, focused on solving partial differential equations using the finite element method. To install `FreeFem++` and for the documentation you can refer to `https://doc.freefem.org/documentation/index.html`.
Our code is available in a GitHub repository at `https://github.com/ManfredNesti/lattice_materials`.

3.3.1 Nonlinear method

In `nonlinear/` you can find the software we developed to solve the nonlinear unit cell optimization problem. This is the graph showing the dependencies bewteen files



params.edp & print_params.edp

This is the module you can use to set your parameters for the simulation.
In the load section you can decide if to start a simulation from scratch of if or to load a previously

computed density (and its mesh) to start from that point. If you use the first option, you need to give an initial density that you can hard-code or set randomly, otherwise you need to give the path and the index of the density you want to load.

```
// Load and mesh
int ld = 0;         // 1: to load a previous computed density, 0: otherwise
string path = "";   // path of the previous computed density
int idxld = 0;      // ii index of the previous computed density
mesh Th;
int n = 30;         // square mesh subdivisions

if(ld) { // load a mesh
  Th = readmesh(path + idxld + "_mesh.txt");
  plot(Th, wait = 0, cmm = "Loaded mesh");
} else { // create a new mesh
  Th = square(n, n, [x, y]);
}

// Inital density
int rnd = 0;     // {0: hard-coded, 1: random} initial density
int seed = 10;   // seed for random initial density
if (ld) {
  ifstream fu(path + idxld + "_density.txt");
  fu >> v[];
  plot(v, wait = 0, fill = 1, cmm = "Loaded density");
} else if (rnd) { // random initial density
  srandom(seed);
  for(int ii = 0; ii < Xhp.ndof; ii++) v[][ii] = random() / 3e9;
  plot(v, fill = 1, cmm = "Initial density", wait = 0);
} else { // hard-coded initial density
  v = (1 - xlb) * cos(pi * (x - 0.5)) * cos(pi * (y - 0.5)) + xlb;
  plot(v, fill = 1, cmm = "Initial density", wait=0);
}
```

Listing 3.1: params.edp: load of initial density and mesh

In the params section you can specify the desired Poisson's ratio ν and Young coefficient E of the base material you want to use. Notice that the λ_L (`L`) coefficient is computed using the 2D formula since we are using 2D materials. In this section you can also set the target Poisson's ratio `nutarget` you want to obtain.

In the spaces sections you find the spaces for the density (simple of periodic), for the displacements and for the constraints on the area.

```
// Params
real nu12;                          // estimated Poisson's ratio
real nu = 0.3;                      // Poisson's ratio of the material
real E = 1.;                        // Young's modulus of the material
real L = E * nu / (1 - nu ^ 2);     // lambda Lame coefficient of the ↩
   material
real M = E / (2 * (1 + nu));        // mu Lame coefficient of the material
real d1 = 0.3;                      // u traction given
real vf = 0.4;                      // allowable volume fraction
real nutarget = -1.0;               // Poisson's ratio target

```

```
// Density space
fespace Xhp(Th,P1);
// fespace Xhp(Th, P1, periodic = [[2,y],[4,y],[1,x],[3,x]]);
Xhp v,               // density
    vHelm,           // Helmholtz filtered density
    vHeav,           // Heaviside filtered density
    vSig,            // Sigmund tanh filtered density
    psi,
    lz,
    uz;             // density, filtered density, test density, params for IPOPT
Xhp m11, m12, m22; // metric computation utilities

// Displacement space
fespace Vh(Th, [P1, P1]);
Vh [u1,u2],                  // displacement
   [u10,u20] = [d1*x,0],     // old displacement (for Newton)
   [psi1,psi2],              // test displacement
   [varu1,varu2],            // displacement increments (for Newton)
   [err1,err2],              // errors (for Newton)
   [l1, l2],                 // lambda
   [phi1, phi2];             // test lambda

// Area space
fespace Xh0(Th, P0); // periodic = [[2,y],[4,y],[1,x],[3,x]]);
Xh0 vol = area;
real Volume = int2d(Th)(1.);
```

Listing 3.2: params.edp: params and spaces

In the last sections you can find the parameters you need to tune. For example, you can change the tolerance and maximum number of iteration for `IPOPT` algorithm, you can choose which filters to use and their parameters, like the β and η parameters of the hyperbolic tangent filter proposed by Sigmund. Finally, you choose whether to use mesh adaptivity and its tolerance.

```
// Power penalty law
int pen = 0;             // 1: for to use the power penaly law, 0: otherwise
real q = 3.;             // startin q exponent
real qmax = 3.;          // maximum value for q exponent in the main cycle

// Helmholtz filter
int helm = 0;            // 1: for to use Helmholtz filter, 0: otherwise
real radius = 1./80^2;   // filtering radius

// Heaviside filter
int heav = 0;            // 1: for to use Heaviside filter, 0: otherwise

// Main params
int maxit = 6;           // maximum number of iteration of the main cycle

// Hyperbolic tangent filter of Sigmund
int sig = 1;             // 1: to use the Sigmund filter, 0: otherwise
real beta0 = 3.5;        // starting beta coefficient
real eta0 = 0.45;        // starting eta coefficient
real betaincr = 1;       // beta increment at each main cycle iteration
real etaincr = 0.02;     // eta increment at each main cycle iteration

// Mesh adaptivity
```

```
int meshad = 1;          // 1: for to use mesh adaptivity, 0: otherwise
real tolmesh = 0.0001;   // desired mesh tollerance
real stopmesh = tolmesh + 1.;
real tau = .00001;
real maxsK;
int ntold = Th.nt;
```

Listing 3.3: params.edp: parameters to be tuned

The `print_params.edp` module helps you taking track of your tuning trials, printing in the `output.txt` file all the parameters you have chosen.

macros.edp

This module contains all the macro definitions we need in the simulation, like the gradient of a vector, the deformation gradient and the Green-Lagrange strain tensor, the second Piola-Kirchhoff stress tensor and so on. In the previous section you can find all their definitions and other more variables.

constraints.edp

This module contains the functions which return the volume constraints in the format `IPOPT` is able to use. These functions are indeed directly passed in the `IPOPT` call in `main.edp`

cost_function/ folder

This module contains the submodules related to the cost function J and its gradient (see `J.edp` and `gradJ.edp`), which respectively contains the primal problem (to be solved iteratively using the Newton's method) and the dual problem (see `primal.edp`, `dual.edp` and `newton.edp`).

adaptmesh/ folder

This folder contains some modules used to create the metric that `FreeFem++` fucntion `adaptmesh` reads to perform the grid adaptation.

run.sh

To start a simulation you need to run this script with `./run.sh`. This script creates a new folder in `nonlinear/results` named with the current timestamp, it runs `main.edp` and saves all the output in `output.txt` which it moves in the current simulation folder, together with all the other `FreeFem++` outputs of the simulation, .

main.edp

This is the main script which is called by `run.sh`. It includes all necessary functions and starts the iterative algorithm 3 using the index `ii` for the optimization procedure. The index `jj` is used inside `IPOPT`, which solves many time the primal problem estimating ν and a single time the dual for each `jj`. In the main cycle you can find the power penalty law, the filters and the mesh adaptivity that can be performed also only at some specific iterations (but not at the last one) activating the corresponding `if` condition. During all the optimization (and in particular before and after some filter usage) the density and the meshes are always plotted and saved, moreover, some useful information are printed both in the terminal and in the `output.txt` file.

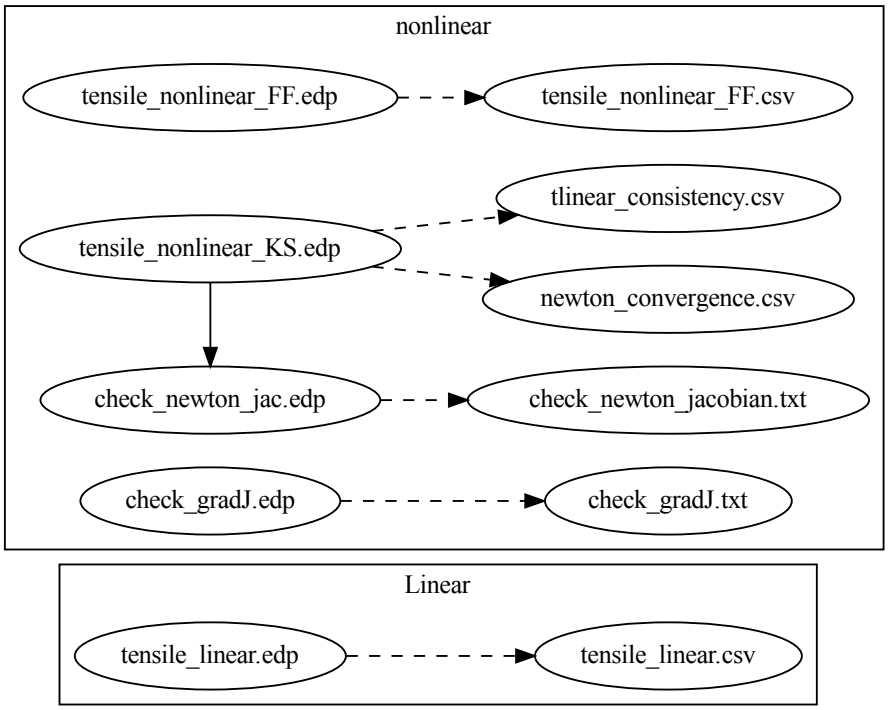
```
while(stopmesh > tolmesh && ii < maxit){
   jj = 0;

   // Power penalty law
   if(pen) q = qmax - 2 * exp(- ii / 2.);
   else q = qmax;

   IPOPT(J, gradJ, ipC, ipGradC, v[],
     lb  = xlb[], ub  = xub[],
        clb = clb  , cub = cub  ,
     checkindex = 1, structjacc = [gvi,gvj],
       maxiter = IPOPTmaxiter + 100 * (ii == 0), warmstart = ii,
     lm = lm, uz = uz[], lz = lz[], tol=  IPOPTtol
     );

   // Helmholtz filter
   if(helm && ii != maxit - 1){
      solve helmholtz(vHelm, psi) = int2d(Th)(radius * (dx(vHelm) * dx(psi)
          + dy(vHelm) * dy(psi)))
                                    + int2d(Th)(vHelm * psi)
                                    - int2d(Th)(v * psi);
      plot(v,  wait = 0, fill = 1, value = 1, ps = "results/" + ii + "
         _density_preHelm.ps", cmm = "Density preHelm " + ii);
      plot(vHelm, wait = 0, fill = 1, value = 1, ps = "results/" + ii + "
         _density_postHelm.ps", cmm = "Density postHelm " + ii);
      v = vHelm;
   }

   // Mesh filter
   if(meshad && ii != maxit - 1){
     maxsK = 1;
     ntold = Th.nt;
     makemetrica(tau);
      Th = adaptmesh(Th, metric = [m11[],m12[],m22[]], nbvx = 12000,
         periodic = [[2,y],[4,y],[1,x],[3,x]]);
      savemesh (Th, "results/" + ii + "_mesh.msh" );
      plot(Th, wait = 0, cmm="New mesh");
      cout << "maxsK = " << maxsK << endl;
   }

   // Heaviside filter
   if(heav && ii != maxit - 1){
    plot(v,  wait = 0, fill = 1, value = 1, ps = "results/" + ii + "
```

```
      _density_preHeav.ps", cmm = "Density preHeav " + ii);
40      real betaH = 10*(ii < 19) + 7*(ii == 19);
41      vHeav = 0.5 + tanh(betaH*(v-0.5))/(2*tanh(0.5*betaH));
42     plot(vHeav, wait = 0, fill = 1, value = 1, ps = "results/" + ii + "↩
          _density_postHeav.ps", cmm = "Density postHeav " +ii);
43     v = vHeav;
44    }
45
46    // Sigmund filter
47    if(sig && ii != maxit - 1){
48      real beta = beta0 + betaincr * ii;
49      real eta = eta0 + etaincr * ii;
50      plot(v,  wait = 0, fill = 1, value = 1, ps = "results/" + ii + "↩
          _density_preSig.ps", cmm = "Density preSig " + ii);
51      vSig = (1-xlb) * (tanh(beta * eta) + tanh(beta * (v - xlb - eta)))
52                     / (tanh(beta * eta) + tanh(beta * (1 - xlb - eta))) + ↩
                          xlb;
53      plot(vSig, wait = 0, fill = 1, value = 1, ps = "results/" + ii + "↩
          _density_postSig.ps", cmm = "Density postSig " + ii);
54      v = vSig;
55    }
56
57   // Mesh projection
58   if(meshad && ii != maxit - 1){
59    v = v;
60    xlb = xlb;
61    xub = xub;
62    gvi = gvi;
63    gvj = 0:Xhp.ndof-1;
64    lz = lz;
65    uz = uz;
66    m11 = m11;
67    m12 = m12;
68    m22 = m22;
69
70    stopmesh = abs(Th.nt - real(ntold))/ntold;
71   }
72
73    ii += 1;
74
75    cout << "Iteration " << ii << ", Mesh error = " << stopmesh << endl;
76
77    {ofstream fout("results/" + ii + "_density.txt");  fout << v[];}
78
79    plot(v,  wait = 0, fill = 1, value = 1, ps = "results/" + ii + "_density↩
        .ps", cmm = "Density "+ii);
80    savevtk("results/" + ii + "_density.vtk", Th, v, dataname = "Density", ↩
        order = ffordervel);
81 }
82
83 cout << Th.nt << endl;
84
85 plot(v,  wait = 0, fill = 1, value = 1, ps = "results/final_density.ps", ↩
    cmm = "Final Density");
```

Listing 3.4: main.edp

3.3.2 Validation tests

During the implementation phase we started from scratch by some simple cases and tests, in order to test step-by-step our work and validate the model. In `tests/` you can find the material related to this topic. This is the graph showing the dependencies between files.



tensile_linear.edp

The first thing we did was to develop a simple tensile problem in which a rectangular full 2D material (density equal to 1 everywhere) is stretched on the right boundary, free-movement condition characterize top boundary and symmetric BCs are imposed on the other boundaries. Solved the problem, the Poisson's ratio is computed using formula (3) proposed in [1]; we had to find a way to compute the transverse and axial strain valid in the formulation of our optimization problem for a nonlinear material. Therefore, once was done through the `convect` function of `FreeFem++`, more reliable with respect to the physical meaning of the quantity used in the formula but not suited for the derivation of the cost function J, and then with an average of the displacements on the boundaries of interest (as in eq. (3.6)). These two estimates, with the same imposed displacement on the right boundary, are saved into `tensile_linear.csv` for variable values of the latter. We can see that, for each value of the imposed displacement, the Poisson's ratio remains the same and that it is equal to the imposed one. Moreover, the Poisson's ratios estimated in the two ways coincide.

```
//*********************************************************
// Estimate of nu using convect
//*********************************************************
real d1conv = int1d(Th, 2)(u1 - convect([ax, ay], -1, u1));
real d2conv = int1d(Th, 3)(u2 - convect([bx, by], -1, u2));
cout << "d1conv = " << d1conv << endl;
cout << "d2conv = " << d2conv << endl;

real nu12conv = - d2conv / d1conv;
cout << "Estimated nu using convect: " << nu12conv << endl;

//*********************************************************
// Estimate of nu Estimate of nu using border integrals
//*********************************************************
real d1Sig = int1d(Th, 2)(u1) - int1d(Th, 4)(u1);
real d2Sig = int1d(Th, 3)(u2) - int1d(Th, 1)(u2);
cout << "d1Sig = " << d1Sig << endl;
cout << "d2Sig = " << d2Sig << endl;

real nu12Sig = - d2Sig / d1Sig;
cout << "Estimated nu by Sigmund formula: " << nu12Sig << endl;
```

Listing 3.5: tensile_linear.edp: estimation of Poisson's ratio

tensile_nonlinear_FF.edp

After that, we started taking into account the nonlinearity. We started from the example of nonlinear elasticity law provided by `FreeFem++` documentation here `https://doc.freefem.org/models/nonlinear-elasticity.html` in which the nonlinearity comes in the ε tensor, in which also the second order terms are considered.
We adapt the code to solve the same tensile problem of the previous test, adding and making change from 1 to 0 the parameter `nl` representing the weight of the nonlinear part of ε. In `tensile_nonlinear_FF.csv` you can see that we have $\nu = 0.68$ for `nl` equal to 1 and that $\nu \to 0.3$ for `nl` tending to 0: this confirms that the nonlinear model is consistent with the linear one since it gives the same result in linear regime on the same test problem. This results can be used as a possible comparison for our nonlinear model.

tensile_nonlinear_KS.edp & check_newton_jac.edp

Since, in our model, the nonlinearity comes from a nonlinear elastic energy-based formulation (see [2]), differently from `FreeFem++` model, in this test we take the primal problem we have formulated and implemented, testing if also this is consistent with the linear case. In `tensile_nonlinear_KS\` folder you can find the output of this test.

- In `linear_consistency.csv` you can see that if the enforced displacement tends to 0, the estimated ν tends to 0.3, since, reducing the enforced displacement we are limiting the nonlinear effects. So, our primal problem is also consistent with the linear case.

- In `newton_convergence.csv` you find the reducing increments used as a convergence criterion for the Newton's method.
- Finally, in `check_newton_jacobian.txt`, you can find the results of a validation test on the Newton jacobian computation, to verify if the derivation of it in section (3.1.1) was done correctly.

check_gradJ.edp

Finally, in this script was validated whether if the gradient of the cost functional J and consequently the dual problem have been computed correctly.

```
out << "--- Check gradient of J ---" << endl;
real J1 = J(rhoneg[]);
real J2 = J(rhopos[]);
real JJ = J(rho[]); // calling J -> solve primal -> find [u1,u2] ↩
    corresponding to rh
out << " J1 = " << J1 << endl;
out << " J2 = " << J2 << endl;
out << " JJ = " << JJ << endl;

J2 = J2 - J1;
out << " J2 - J1 = " << J2 << endl;
J2 = (2*eps)^(-1)*J2;
out << " (J2 - J1) / (2 * eps) = " << J2 << endl << endl;

real[int] gradientJ = gradJ(rho[]);
out << " gradJ = " << gradientJ << endl;
real dJ = gradientJ '* drho[];
out << " dJ = gradientJ '* drho = " << dJ << endl;
J1 = J2 - dJ;
out << " Difference (J2 - J1) / (2 * eps) - dJ = " << J1 << endl;
```

Listing 3.6: check_gradJ.edp: check of the gradient

3.4 Results

3.4.1 Results on validation tests

Here are presented all the graphical results of the validation tests explained in section (3.3.2). In fig. 3.2 is confirmed that Poisson's ratio can be computed in both way, giving the same result for any traction imposed in linear regime. In both fig. 3.3(a) and 3.3(b) we see how for small weight of the nonlinear component of the strain in the first case and for small traction imposed d_1 in the second case, we recover Poisson's ratio computed in the linear regime 0.3. Actually those two graphs cannot be compared directly one with the other because quantity d_1 and nl have a different meaning in the model: one represents the right traction imposed, having

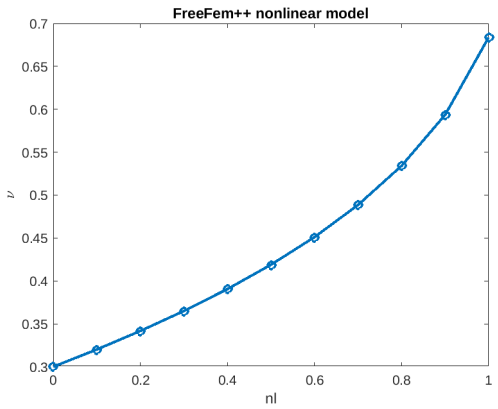
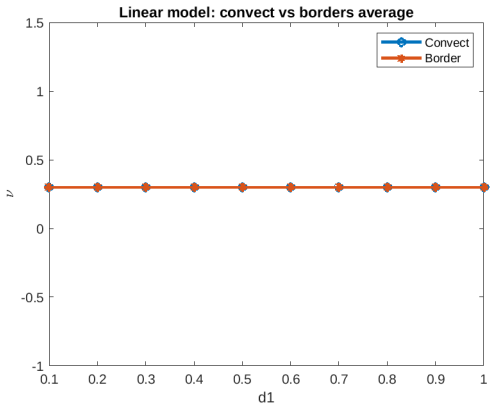
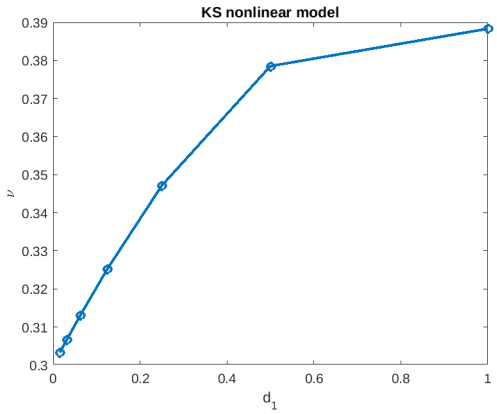


Figure 3.2: Check on the linear model for the computation of Poisson's ratio



(a)



(b)

Figure 3.3: Check on the nonlinear model

more real physical meaning, the second one is a dummy variable used to activate nonlinear component that are neglected in the linear model.

Regarding the validation of the computation of the derivatives used in Lagrangian formulation, `check_newton_jac.edp` gave as residuals in L^∞ norm:

$$\left\| \frac{[F(\boldsymbol{u} + \varepsilon\, \boldsymbol{du}) - F(\boldsymbol{u} - \varepsilon\, \boldsymbol{du})]}{2\varepsilon} - Jac(\boldsymbol{u}) \right\|_{L^\infty} = 9.09059e-08$$

therefore we can evaluate correct our formulas.

For what concern `check_gradJ.edp` results, similarly it gave really small residuals, confirming that the whole Lagrangian formulation has been derived correctly:

$$\left\| \frac{J(\rho + \varepsilon\, d\rho) - J(\rho - \varepsilon\, d\rho)}{2\varepsilon} - gradJ(\rho) \right\|_{L^\infty} = 7.18473e-12$$

3.4.2 Parameters tuning

In the optimization procedure the target Poisson's ratio ν^* is always reached thanks to `IPOPT` algorithm, the difficult part was to obtain a reasonable structure for our material in a reasonable number of iterations, considering also the role in computational cost and time of Newton's method iterations. In order to do so, after having validated our model for the nonlinear material (see section before (3.4.1)), we tried different combination of features, filters, parameters to get a better performance.

The right traction imposed to the structure d_1 and the Poisson coefficient we want to reach ν^* can be changed according to the TO problem we want to solve. We started from imposing a small traction 0.01 to recover a linear regime, then to have results that could be compared with literature results found in [1] we impose $d_1 = 0.3$ traction and $\nu^* = -1$.

As can be seen in the code `params.edp`, it is possible to choose the initial density of the optimization procedure; there are been implemented three main possible alternatives:

- random density, generated by `random()` function of `FreeFem++`;
- loaded density, taken from previous results
- fixed density, chosen previously by a specific convenient function

The best structure obtained up to now started from $\rho_0 = (1 - \texttt{lb})\cos(\pi(x - 0.5))\cos(\pi(y - 0.5)) + \texttt{lb}$. In this way the structure is symmetric and sufficiently smooth from the beginning (with random distribution of density this was not possible) and moreover we prevent density to be smaller than its lower bound $\texttt{lb} = \rho_{\min}$ and bigger than 1.

Taking examples from papers [1], [4] and [5] on the use of the filter parameters, we have tried multiple combinations of those, with an insight on the tuning of the input parameters, and we end up with a proper algorithm looking at our results in terms of quality of the figure and of the computational time of the procedure.

First of all, we did not find convenient to apply **Heaviside** filter, since we were more familiar with the smoothed Heaviside function used in [1] paper. Moreover we also tried to be helped by **Helmotz** filter to smooth our density when `IPOPT` gave too harsh structures. At the end of our attempts we focused on the use of **Sigmund** filter. Concerning the use of it, we preferred to apply the Sigmund filter only after the optimization phase, to sharpen the structure obtained from `IPOPT` that has already reached the target Poisson's ratio ν^*. Moreover, we prefer not to interfere with `IPOPT` algorithm since it is an optimization procedure which does not give a lot of control on when it is actually filtered the density. It would be necessary to use β much smaller, otherwise the structure loose significance, as seen from some tries, and using a filter with such

a small β does not really give much gain on the method. Whereas, filtering the density after `IPOPT` is more efficient and it can be used with more meaningful parameters threshold η and β.
For what concerns the **penalty coefficient** p, in literature were found different approaches. At the beginning, we tried to make p change gradually from 1 to a certain $p_{\max}$ following the exponential function $p = p_{\max} - 2\exp(-\texttt{ii}/2.)$, where `ii` is the index identifying the iteration of the main cycle (see 3.2); then we tried to make it change with a step function every defined `K` iterations. From results obtained up to now, it is better also for simplicity of the algorithm, to keep p fixed equal to 3.
The last parameters of the algorithm to be tuned were `IPOPTmaxiter` and `maxit`, since they both depend on the filters used and changed considerably when added mesh adaptation. The first one represents the stopping criterion in case `IPOPT` does not reach the desired tolerance in the optimization procedure. We have seen from our attempts that it is better at the first `IPOPT` cycle (corresponding to `ii` = 0) to let `IPOPT` do a large number of iterations (we have chosen 150) so that we are sure it converges to the desired ν^*; then `IPOPTmaxiter` can be reduced to 50, since after the first cycle less iterations are necessary to reach the desired tolerance. Every `IPOPT` cycle after the first is mainly necessary to restore ν^* after the filters and mesh adaptation, introduced to better capture a reasonable structure for manufacturability. Indeed at the beginning of the tuning phase, when we were not able to use properly the filters, the maximum number of iteration `maxit` to be performed in the main cycle was taken equal to 12; when we reached a good compromise between **Sigmund** filter and **mesh adaptation** we decided to consider acceptable the results obtained after only 6 iterations of the algorithm.

3.4.3 Optimized materials obtained

We decided to optimize the structure of material in order to have $\nu = -1$ under a traction $d_1 = 0.3$. We started from the initial density $\rho_0 = (1 - \texttt{lb})\cos(\pi(x - 0.5))\cos(\pi(y - 0.5)) + \texttt{lb}$ you can see in fig. 3.4.
After a first optimization cycle of `IPOPT` the result is in fig. 3.5.
As you can see, of course this structure has not a very clear physical meaning since in most of the domain the density value is intermediate between 0 and 1: this means that some filters are needed.
In fig. 3.6 you can see the plots obtained with `maxit` = 6, in which, at each iteration, the filter by Sigmund is applied starting from $\beta = 3.5$ and $\eta = 0.45$ and increasing at each step these parameters of 1 and 0.02 respectively.
As you can see, at each iteration the filter penalizes the intermediate values of the density using an increasing threshold η and an increasing slope β, so that the final one has mostly value 0 and 1 bu it exhibits a ladder effect due to the filter.

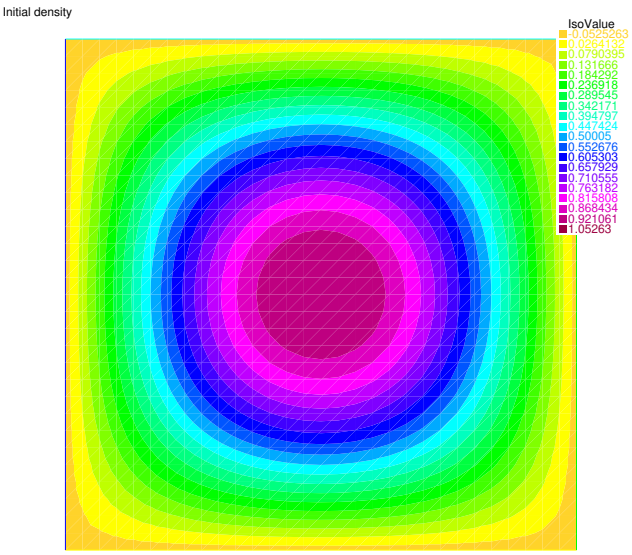


Figure 3.4: Initial density

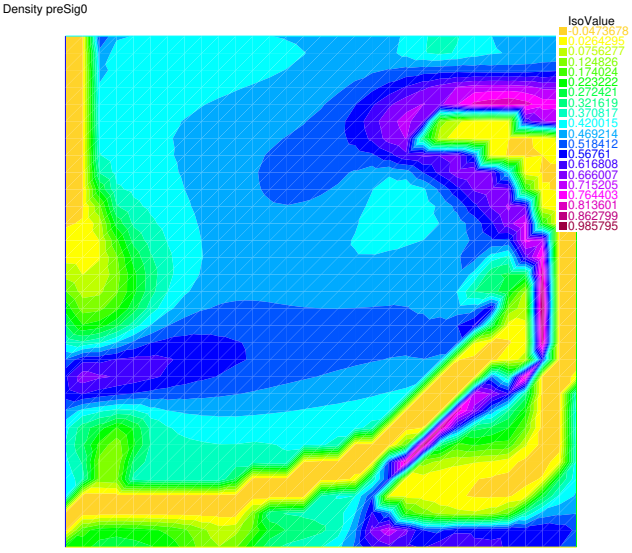
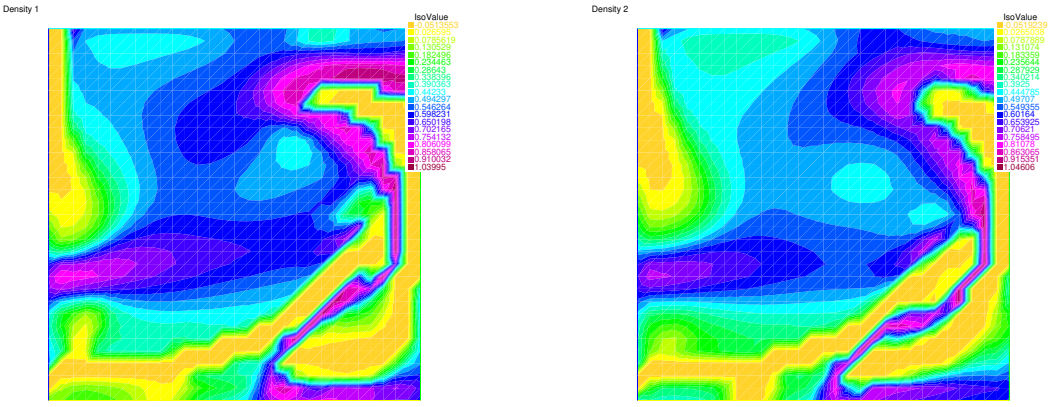
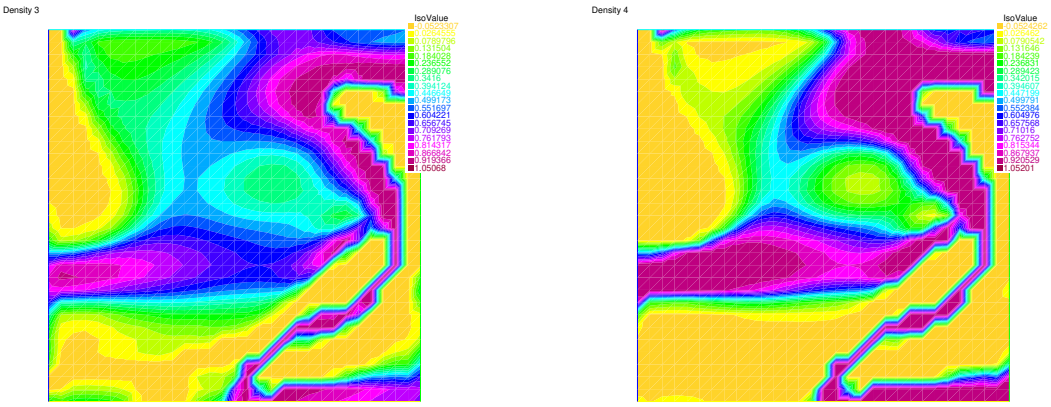


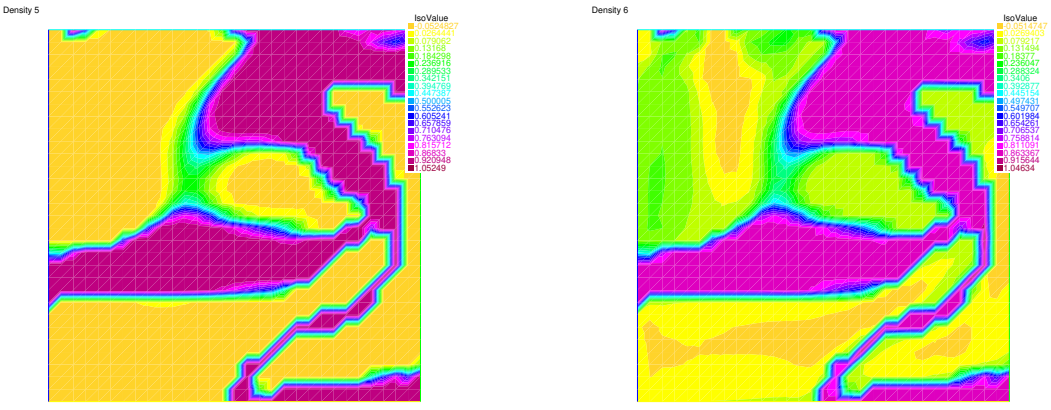
Figure 3.5: Density after the first IPOPT cycle



(a) ii = 1 (b) ii = 2



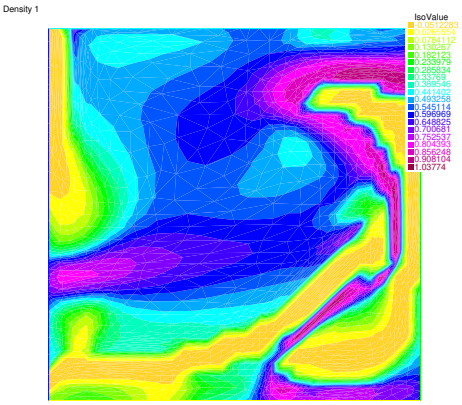
(c) ii = 3 (d) ii = 4



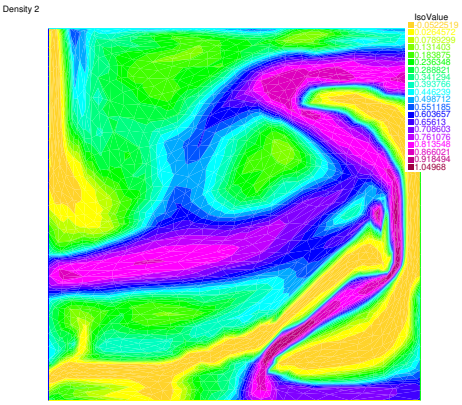
(e) ii = 5 (f) ii = 6

Figure 3.6: Structure with increasing Sigmund filters parameters

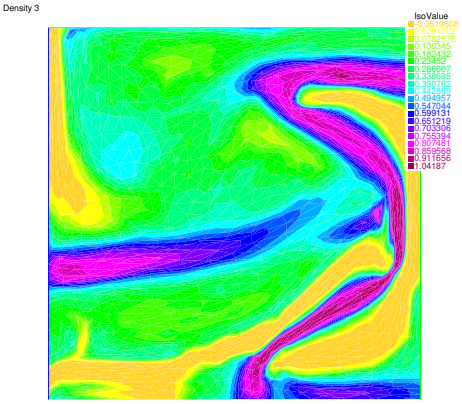
We tried to make the border of the structure smoother using mesh adaptivity. This is an example in which also mesh adaptivity is performed at each step together with sigmund filter, see Figure 3.7 and Figure 3.8.
You can see the power of the grid adaptation which strongly reduces the ladder effect, making the boundary of the structure very smooth and helping also the filter to set at 0 the values of the density corresponding to the void.
However, the usage of grid adaptation at each step is too strong, indeed you can see that, already from the second iteration, the optimizer tends to divide in two parts the structure.
We found that using grid adaptation at each step but keeping the threshold $\eta = 0.5$ constant during the iteration avoid this problem, as you can see in fig. 3.9.
However, there is a spike in the structure, which is quite dissimilar from the starting one. So we tried again with η increasing, but using the mesh adaptivity only at the fourth iteration. This trials is plotted in fig. 3.10.
The fifth iteration of this simulation exhibits a nice structure, similar to the first one but with a smoother boundary and a density which is mostly equal to 0 and 1. You can find the final result in fig. 3.11.
Since we imposed symmetric boundary conditions on left and bottom edge, we need to reflect our cell on those sides before repeating it to build up the material. In fig. 3.12 you can see the reflection of the unit cell to build up a 2×2 square.
Using this as a unit cell, we can finally build up the final lattice material by periodicity, as in fig. 3.13.



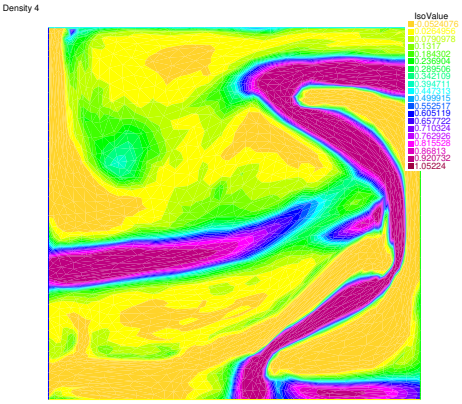
(a) ii = 1



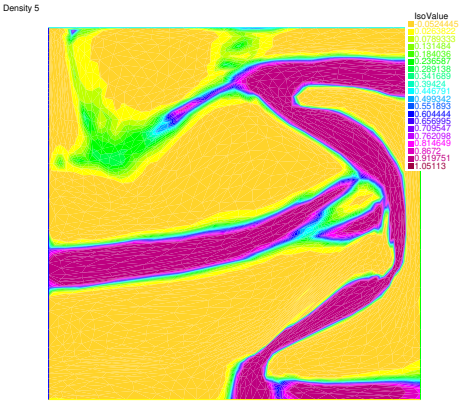
(b) ii = 2



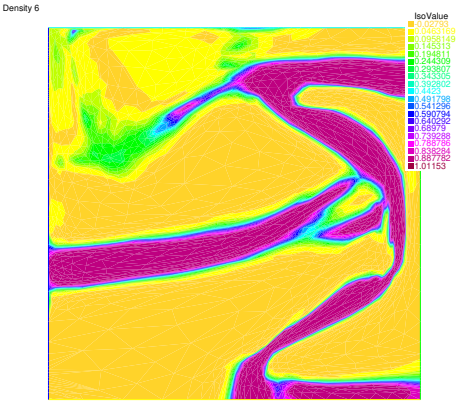
(c) ii = 3



(d) ii = 4

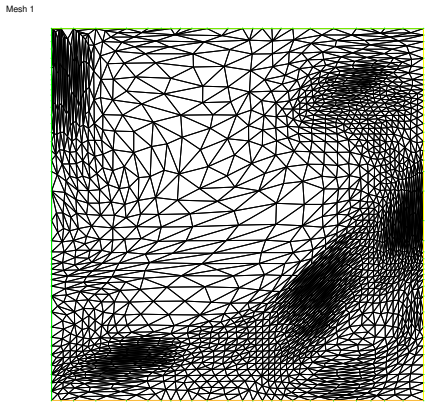


(e) ii = 5

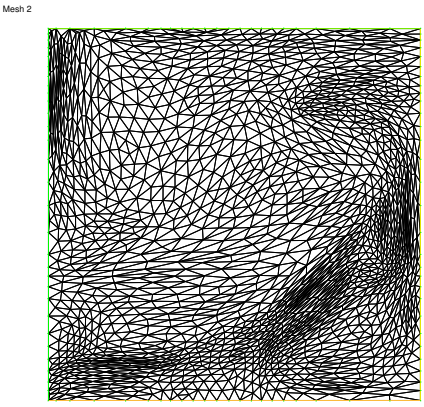


(f) ii = 6

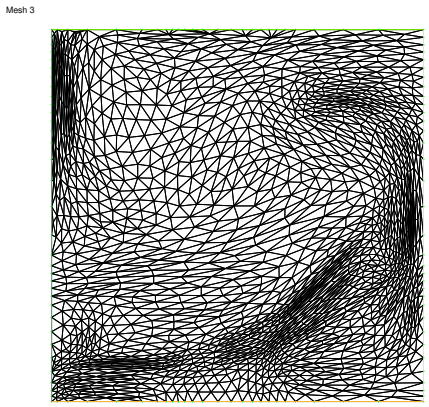
Figure 3.7: Structure with increasing Sigmund filters parameters and grid adaptation



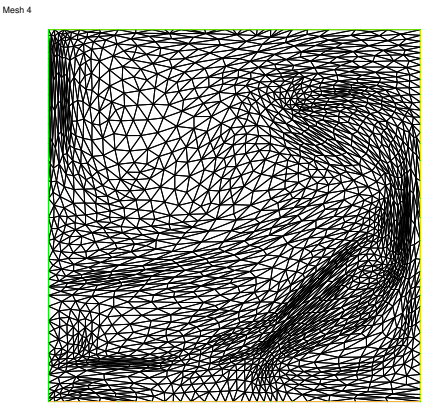
(a) ii = 1



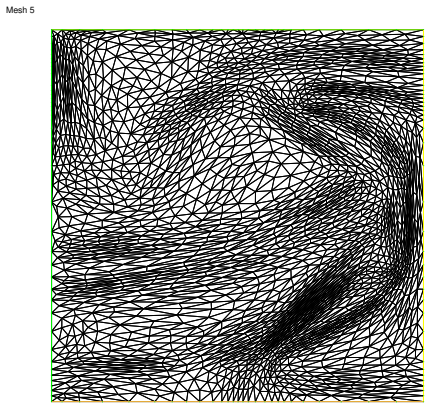
(b) ii = 2



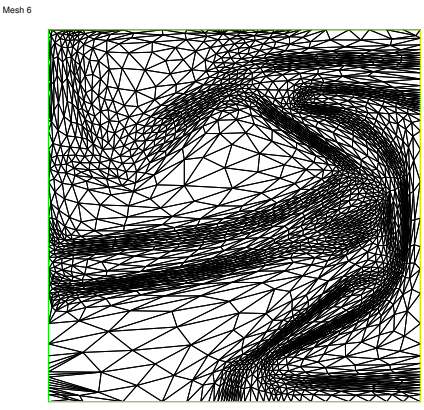
(c) ii = 3



(d) ii = 4

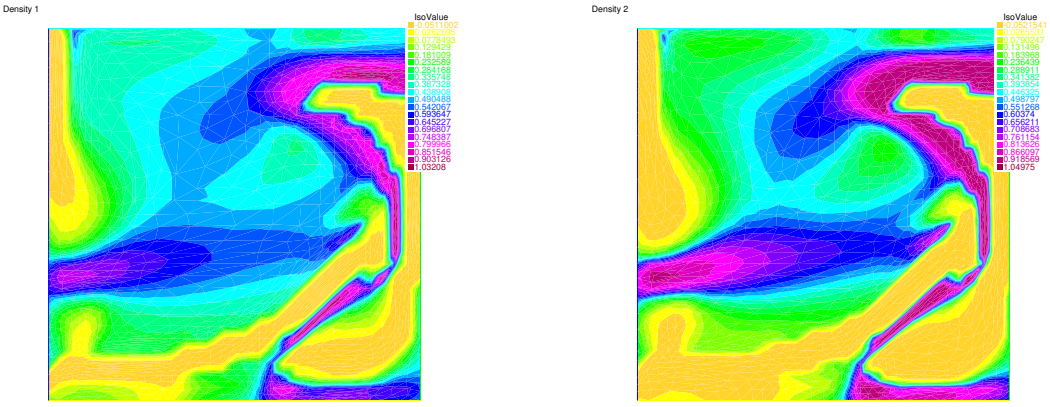


(e) ii = 5

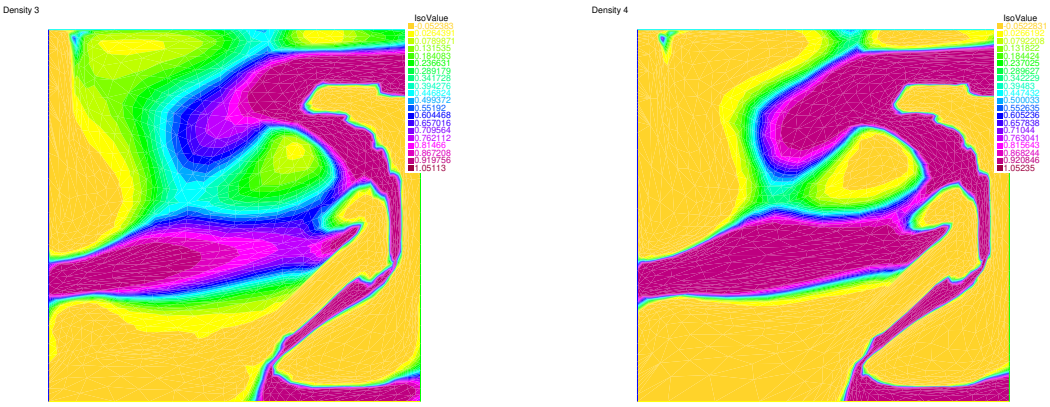


(f) ii = 6

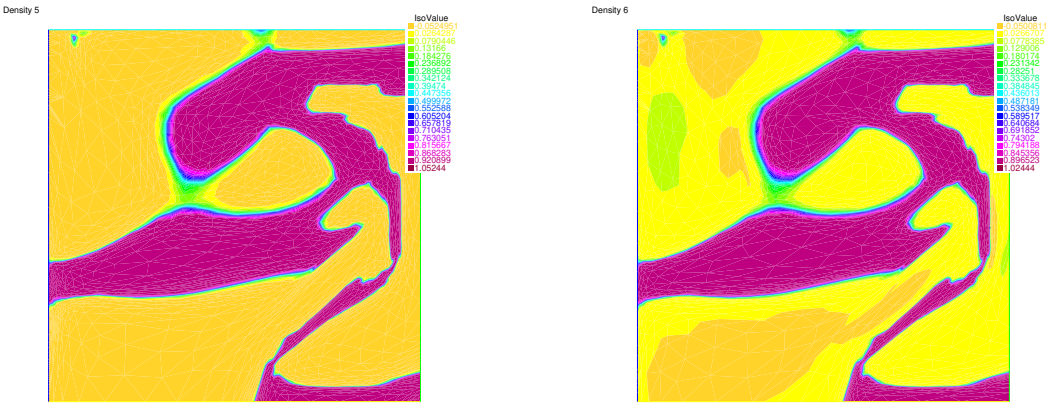
Figure 3.8: Mesh of structures with increasing Sigmund filters parameters and grid adaptation



(a) ii = 1 (b) ii = 2

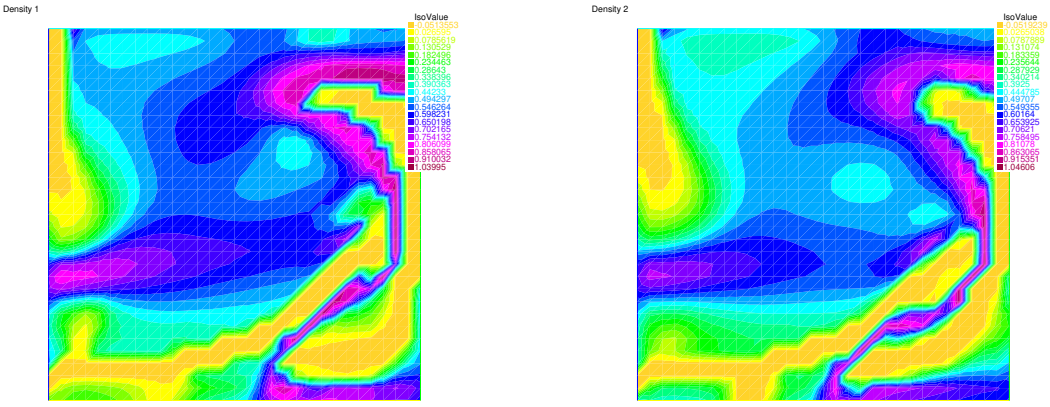


(c) ii = 3 (d) ii = 4



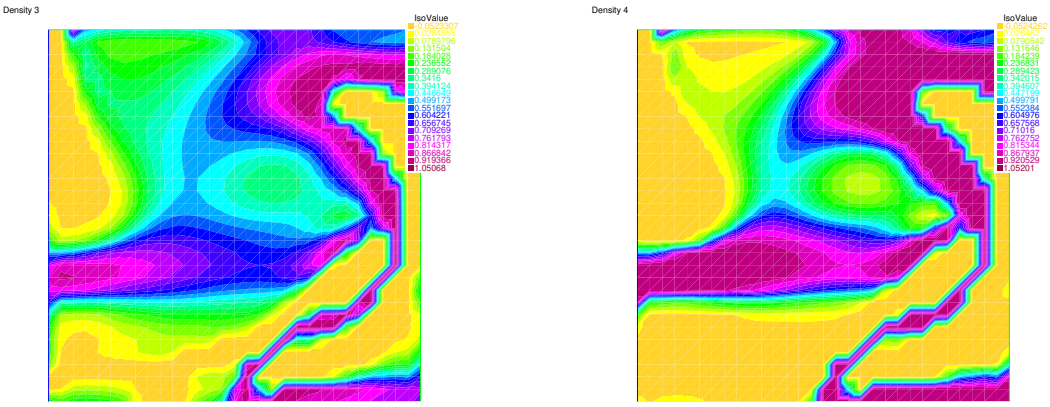
(e) ii = 5 (f) ii = 6

Figure 3.9: Structure with fixed $\eta = 0.5$ and grid adaptation



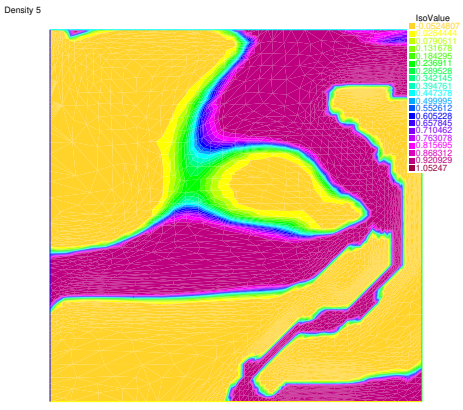
(a) ii = 1

(b) ii = 2



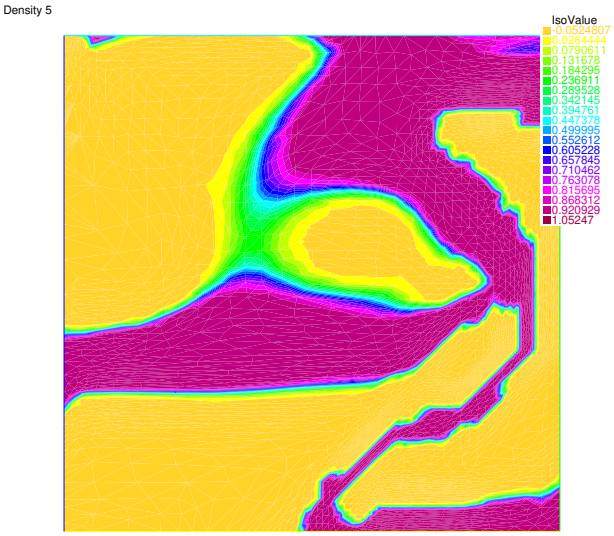
(c) ii = 3

(d) ii = 4

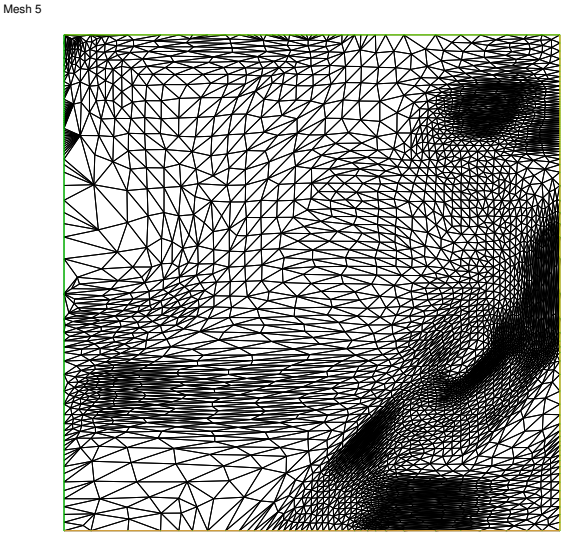


(e) ii = 5

Figure 3.10: Final structure optimization



(a) Final structure



(b) Final mesh

Figure 3.11: Final structure and mesh

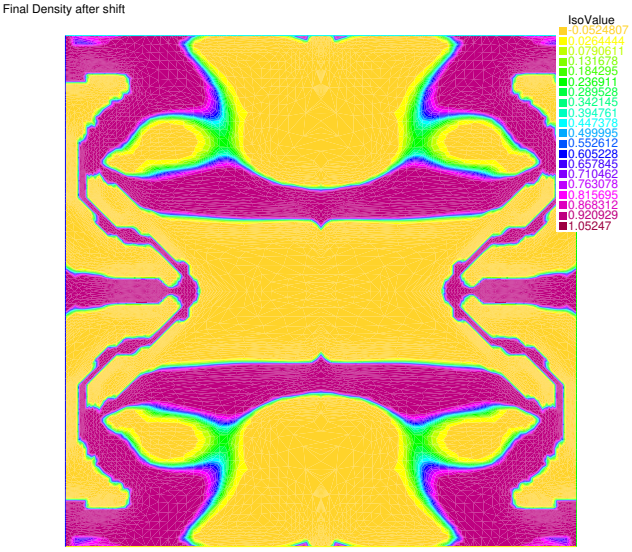


Figure 3.12: Reflected (2×2) final structure

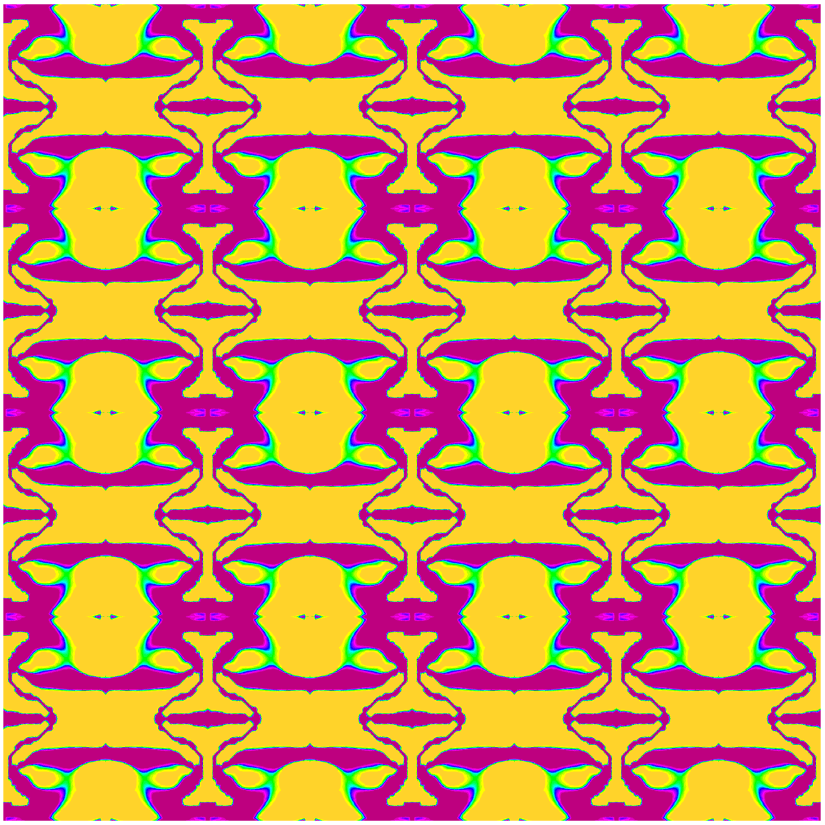


Figure 3.13: Final lattice material

Chapter 4

Conclusions

In this project was studied a possible model for nonlinear material under a tensile test and was investigated one possible good algorithm to solve a TO problem whose aim was to obtain a 2D continuum material characterized by a negative Poisson's ratio $\nu^* = -1$. Topology Optimization problems are characterized by different numerical issues and instabilities. In some works were found stable and efficient way to construct material structures that could be manufactured through the use of a 3D printing, thanks to a proper use of filters and other techniques.

In chapter 3 is shown how we derived a suitable model for material in a nonlinear regime that could be applied in the `IPOPT` optimization algorithm for the `SIMP` method. As it is explained in section 3.4.1, our model has been sufficiently verified, both regarding the choice of the cost function J, and consequently its gradient and the evaluation of the Poisson's ratio ν, and the Newton's method derivation through the composition of derivatives of the elastic energy assumed for the treatment of the material behaviour. All our tests give good results and we evaluate our model reliable to be used in a `SIMPATY` algorithm.

The second main task of the project was to find the best combination of filters, input parameters and finally application of mesh adaptation technique to obtain a final layout well defined, intrinsically smooth, with reduced checkerboards effects, so that the post-processing phase can be strongly reduced and the structure can directly move on to the production manufacturing phase. In particular in section 3.4.2 and 3.4.3 is explained how our decisions were driven according to the results of the simulations. We have experienced how difficult is to tune filters to use in a TO problem, since they are very problem specific and we think this issue it's increased by the complexity of the nonlinear problem to solve at every iteration, differently from the linear material.

In the model treated in [1], our reference work, periodic boundary conditions were imposed on the displacement variable $\boldsymbol{u}$ on all the edges of the unit cell Y. For our model, instead, has been

necessary the imposition of symmetry boundary condition for the displacement in the bottom and left side of the cell and we tested the final layout reflecting the unit cell according to this constraint.

In the final best results obtained we still rely on the benefit of a smoothed Heaviside filter, but we were able to see the increase in qualitative and quantitative performance of the method when using mesh adaptation. The algorithm is still sensitive to small changes in the optimization requests and therefore it is not so stable with respect to the constraints and ν^*. Therefore, possible future developments could generalize this method to other TO problems, other tests or using other elastic energy for nonlinear materials and apply it also to the 3D setting.

Bibliography

[1] F. Wang, O. Sigmund, J.S. Jensen, *Design of materials with prescribed nonlinear properties*: Journal of the Mechanics and Physics of Solids (2014); 69: 156–174

[2] Anders Klarbring, Niclas Strömberg, *Topology optimization of hyperelastic bodies including non-zero prescribed displacements*: Struct Multidisc Optim (2013); 47:37–48

[3] Liang Xia, Piotr Breitkopf, *Recent Advances on Topology Optimization of Multiscale Nonlinear Structures*: Arch Computat Methods Eng (2017); 24:227–249

[4] O. Sigmund, *Manufacturing tolerant topology optimization*: Acta Mech Sin (2009); 25: 227–239

[5] Shengli Xu, Yuanwu Cai, Gengdong Cheng, *Volume preserving nonlinear density filter based on heaviside functions*:Struct Multidisc Optim (2010); 41:495–505

[6] Andreas Wächter, Lorenz T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*: Math. Program., Ser. A (2006); 106: 25–57

[7] Micheletti S., Perotto S., *Anisotropic adaptation via a Zienkiewicz–Zhu error estimator for 2D elliptic problems.*In: G. Kreiss, P. Lötstedt, A. Målqvist, M. Neytcheva (eds.) Numerical Mathematics and Advanced Applications, pp. 645–653. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg (2010)

[8] Micheletti S., Perotto S., Soli L., *Topology optimization driven by anisotropic mesh adaptation: Towards a free-form design*: Computers and Structures (2019); 214: 60–72

[9] Grégoire Allaire, *A brief introduction to homogenization and miscellaneous applications.* In:E. Cancès and S. Labbé, Editors. ESAIM: PROCEEDINGS (September 2012); Vol. 37, p. 1-49

[10] N. Ferro, S. Micheletti, S. Perotto, *Density-Based Inverse Homogenization with Anisotropically Adapted Elements.* In: van Brummelen H., Corsini A., Perotto S., Rozza G. (eds)

Numerical Methods for Flows. Lecture Notes in Computational Science and Engineering, vol 132. Springer, Cham. (2020).

[11] Alfio Quarteroni, *Numerical Models for Differential Problems - Third Edition*, vol. 16: Springer; 2017: 98-112

[12] Ulrik Darling Larsen, Ole Sigmund, and Siebe Bouwstra, *Design and Fabrication of Compliant Micromechanisms and Structures with Negative Poisson's Ratio*, Journal of microelectromechanical systems, Vol. 6, N. 2, (June 1997)

[13] Andreassen E., Andreasen C.S., *How to determine composite material properties using numerical homogenization*. Comp. Mater. Sci. 83, 488–495 (2014)

[14] Sigmund O., *Materials with prescribed constitutive parameters: an inverse homogenization problem*. Internat. J. Solids Structures 31(17), 2313–2329 (1994)

[15] Neves M.M., Rodrigues H., Guedes J.M., *Optimal design of periodic linear elastic microstructures*. Comput. Struct. 76(1-3), 421–429 (2000)

[16] Sigmund O., *Tailoring materials with prescribed elastic properties*. Mechanics of Materials 20 (1995) 351-368

[17] Sigmund O.: Design of Material Structures Using Topology Optimization. Technical University of Denmark, Lyngby, Denmark (1994)

[18] Fengwen Wang, *Systematic design of 3D auxetic lattice materials with programmable Poisson's ratio for finite strains*. Journal of the Mechanics and Physics of Solids 114 (2018); 303–318