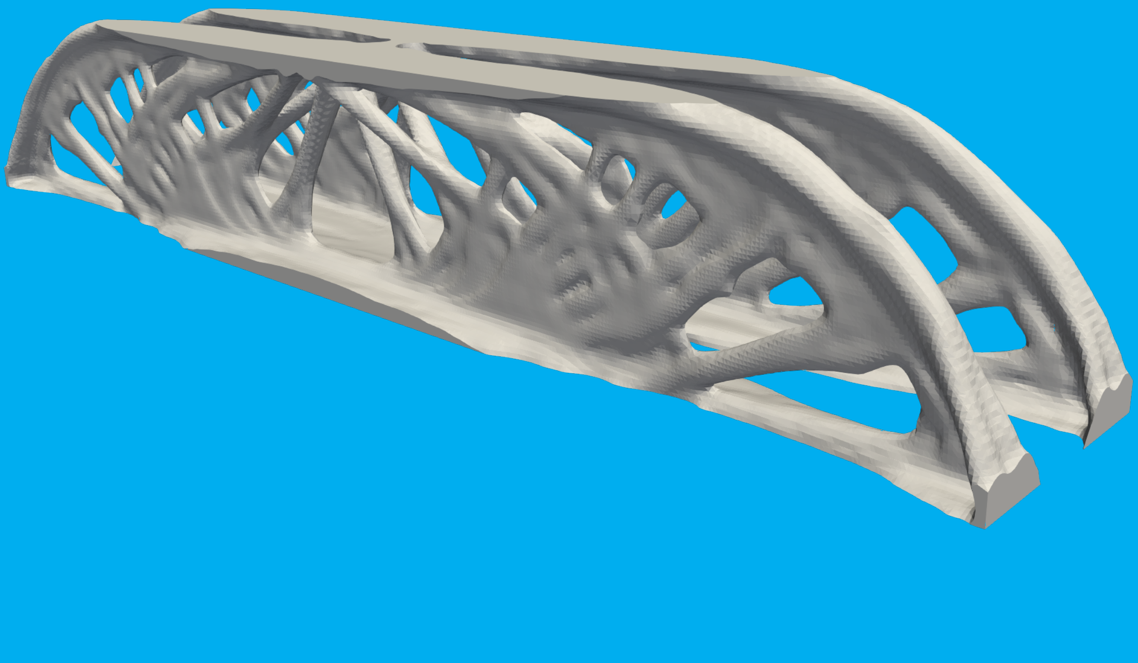
Topology Optimization

Using CutFEM

Reinier Giele



TU Delft

DTU

Topology Optimization

Using CutFEM

by

Reinier Giele

to obtain the degrees

Master of Science
in Aerospace Engineering
at the Delft University of Technology,

&

Master of Science
in Engineering (European Wind Energy)
at the Technical University of Denmark,

Supervisors:	Dr. N. Aage,	DTU
	Dr. C. S. Andreasen,	DTU
	Dr. B. Chen,	TU Delft

Summary

The demand for better performing structural designs gives rise to the interest in topology optimization. An accurate geometry description is fundamental in the optimization process. The Cut Finite Element Method (CutFEM) can describe the object surface on a fixed grid with an immersed boundary. To this level set method, multiple techniques from the density method can be added. In this thesis project, the performance of 3D topology optimization using CutFEM is tested.

A topology optimization model using CutFEM was developed. For the elements on the boundary (Cut Elements), a cut is made to split the element in a solid and fluid part. The Gauss points that represent each part are calculated in order to find the stiffness of the Cut Elements. The performance has been tested by performing finite element analysis with CutFEM.

In order to perform topology optimization, the sensitivities are calculated with the adjoint method, a filter was used with Heaviside projection and mapping, and the Method of Moving Asymptotes (MMA) is used in order to find the design of the next iteration. In order to increase the length scale, an optional robust design method was implemented which creates an eroded and dilated design. The performance of topology optimization using CutFEM was tested by optimizing a structure for minimum compliance for a set of loading conditions. This was compared to topology optimization with classical Solid Isotropic Material with Penalization (SIMP) method.

Firstly, it was found that Cut Elements are an accurate method to perform a finite element analysis. Next, it was found that topology optimization using CutFEM is able to obtain a better objective function than topology optimization using the SIMP method. The computational costs of the CutFEM method are substancially higher. In 3D topology optimization using CutFEM, the design changes can happen everywhere on the boundary, so that the initial structural design is of less importance than for 2D. Next, it was found that the robust design method works well in increasing the length scale, but the objective function is decreased and the initial design is more important. It is thought that CutFEM could best be used to perform optimization with the initial design computed by the SIMP method.

Finally, the CutFEM method has been used in a Navier Stokes fluid solver with Brinkman penalization implementation. It was found that this does not work, and a fluid solver with a hard boundary method is required. It is recommended to implement the CutFEM method in a fluid solver with Nitsches method.

Acknowledgement

With this thesis project I will conclude my studies of the European Wind Energy Master at Delft University of Technology, and the Technical University of Denmark. I have greatly enjoyed the project, which can hopefully make its contribution, and be used to grow into something more.

I would like to express my gratitude to some people specific. First of all, Niels Aage and Casper Andreasen, for all their time and energy, and many fruitfull discussions. They really managed to transfer their enthusiasm for the topic. I have learned a lot, and without them this project would not have been there. Next, I would like to thank Boyang Chen for all of his adequate advise on how to approach problems, and for introducing me to the academic way of thinking. Furthermore, I would like to thank Ole Sigmund, for introducing me to this field of research and bringing me into the team. And finally, Karin, Gertjan, and Marieke, thank you for all your unbounded and unconditional support, both in and out of the scope of this project.

I would also like to take this moment to reflect on my time as a student. It has been a great journey, with many enjoyable moments with different groups of people. I have had the privilege to get some fantastic experiences abroad, but have always been welcomed to a fantastic home, both in Delft and Copenhagen. Every day I feel blessed with and proud of such a wonderful group of friends, who encourage me in every step. Thank you to everyone who has contributed to all of this.

Reinier Giele
Copenhagen, July 2019

Contents

1 Introduction

A *structure* is defined by Gordon (1978)[43] as "any assemblage of materials which is intended to sustain loads". Thus, *structural optimization* is defined by Christense and Klarbring (2008) [30] as "the subject of making an assemblage of materials sustain loads in the best way". Here, the definition of "best" depends on the point of view, and can be specified in different ways. This is dependent on the optimization problem, with the *objective* and *constraints*.

Topology optimization is one class of structural optimization and is a numerical method that optimises material layout in a prescribed design domain for best structural performance [78], see Figure 1.1. The structure is subject to loads, boundary conditions, and definition of maximum performance of the system. The design is analysed and optimised in iterations with the use of mathematical programming techniques.

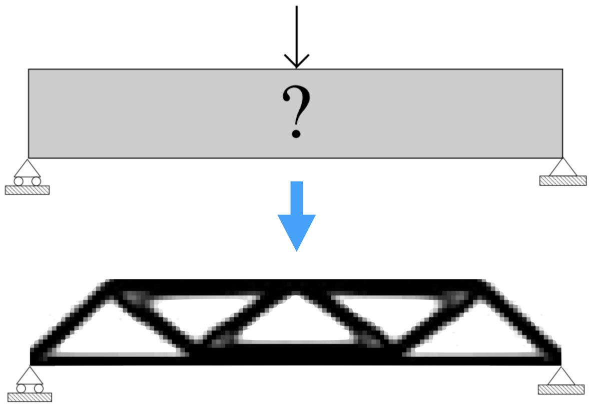


Figure 1.1: Topology Optimization

As aforementioned, the structure will be subject to an applied load. Structural loads are forces, deformations, or accelerations applied to a structure or its components. These loads can be caused by different phenomena. In this paper, the loads coming from *fluids* will be considered. This is relevant at for example wind turbines or aircraft. *Fluid–structure interaction (FSI)* is the interaction of some movable or deformable structure with an internal or surrounding fluid flow [25]. FSI is a strongly coupled phenomenon, where the fluid flow may depend on the structural deformation, and the structural deformation may depend on the fluid flow. The challenges on the optimization method of a strongly coupled FSI problem are summarized by Jenkins (2015) [50] as: (a) describing the geometry of the structure, (b) modelling the flow, the structural response and their interactions, (c) discretizing the FSI model, and (d) solving efficiently and robustly the resulting numerical problems. So, the interface, its representation, and the modelling accuracy are crucial in order to obtain a valid FSI model.

To deal with the challenges around the boundary between solid and fluid, a new numerical method will be used: The *Cut Finite Element Method (CutFEM)* [27] is an immersed boundary technique where embedded boundaries or interfaces with different phases are cut-off. An example can be seen in Figure 1.2. A possible advantage of CutFEM is to make the discretization as independent as possible of the geometric description and minimize the complexity of mesh generation, while retaining the accuracy and robustness of a standard finite element method [27].

CutFEM offers intersting possibilities to the topology optimization of FSI problems. This has not been carried out before. The goal of this project, is to use CutFEM for topology optimization of an FSI problem, and measure its performance. This is very fundamental but later might lead to usage in industry.

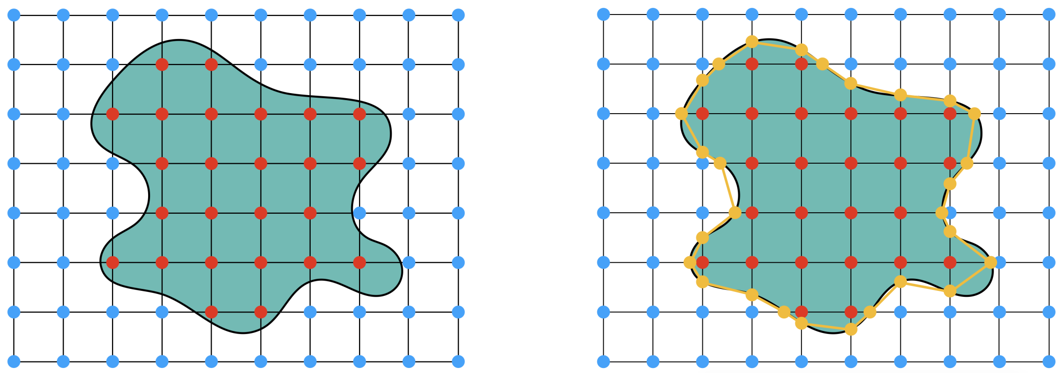

(a) The phase described in the nodes (b) The boundary described by CutFEM

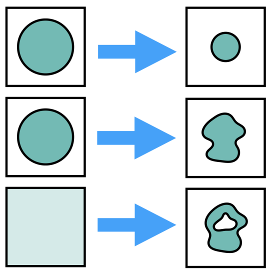
Figure 1.2: CutFEM: a structural geometry is described by Cut Elements on the boundary

1.1. State of the art

In this section, for all relevant fields, the background information is given, followed by different approaches in this field. The relevant fields are *topology optimization*, the *density method* and *level set method*, *CutFEM*, and *fluid-structure interaction*.

Topology Optimization:
Optimization of design can be traced back to the origins of structural mechanics. Galileo Galilei wrote in 1638 about the optimum shape of cantilever beams in his work "Discorsi". In 1869 Maxwell wrote about structural optimization [62]. Maxwell shows how certain trusses have reciprocal diagrams which represent the forces in the trusses, which approach had a major impact on the field of structural engineering. The first paper on topology optimization was published in 1904 by the Australian inventor Michell [64]. He states that "*a frame* (today called truss) (is optimal) *attains the limit of economy of material possible in any frame-structure under the same applied forces, if the space occupied by it can be subjected to an appropriate small deformation, such that the strains in all the bars of the frame are increased by equal fractions of their lengths, not less than the fractional change of length of any element of the space.*" Thus, the structures are optimal based on defined criteria. Structural optimization can be divided into the three classes listed below [14] [78], see Figure 1.3 for a visual representation.

- *Sizing optimization*: the goal is to find the optimum of a design variable while the layout and connectivity that is already known. For example, the optimal thickness distribution of a linearly elastic plate or the optimal member areas in a truss structure.
- *Shape optimization*: the goal is to find the optimum shape of the domain, where this domain is now the design variable.
- *Topology optimization*: the goal is to optimize both the material layout and the connectivity inside a design domain. This is the most general form of structural optimization. For example, finding the optimum of features such as the number and location and shape of holes and the connectivity of the domain.


(a) Conceptual

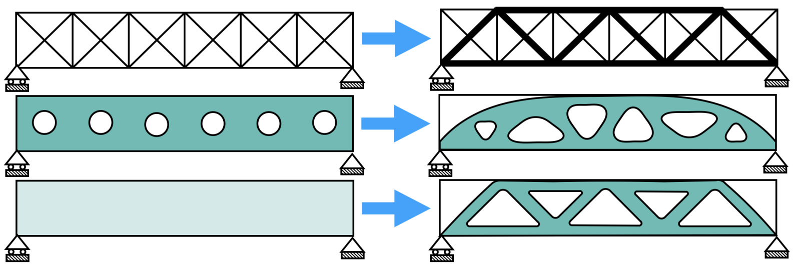

(b) Structural optimization

Figure 1.3: Structural Optimization: *top* is size, *middle* is shape, *bottom* is topology

This thesis project will focus on topology optimization, and which gives answers to the most fundamental engineering question: how to place material within a prescribed design domain in order to obtain the best structural performance? CutFEM allows for changes in topology, for example the creation of holes, which makes it different from shape optimization.

For topology optimization, Bendsoe and Kikuchi (1988) [15] is considered to be a pioneering paper, with the subject of *homogenization approach to topology optimization*. In here, the idea behind the *Solid Isotropic Material with Penalisation (SIMP)* is introduced. This is a material based interpolation where the Young's modulus of the material to the scalar selection field is interpolated like $E = E_0 + \rho^p(E_1 - E_0)$ or more general $E(\rho) = g(\rho_i)E_0$. The penalisation parameter p makes it unfavorable to have intermediate densities in the optimal design. Good overviews on topology optimization are given by Bendsoe and Sigmund (2003) [14] and Sigmund and Maute (2013) [78].

Although topology optimization was introduced for mechanical problems, it is applicable in a large range of fields, for example acoustics [36], electrostatics [93], fluid-structure-interaction in poroelasticity [9], photonics [51], fluid dynamics [19], and thermal transport [8] [3]. Topology optimization often leads to forms which are difficult to manufacture, which is why currently it is mostly used at the concept level of a design process. The growth computational power, need for efficient designs, and improvements in additive manufacturing, are all causing a growth in the potential of topology optimization in different industries.

Density Method vs Level Set Method:
The structural geometry is typically defined via the distribution of two or more material phases in a given design domain. Topology optimization has developed in a number of different directions, including: density (see [15] [94] [66]), level set (see [4] [5] [87]), topological derivative (see [80]), phase field (see [21]). The two main approaches with continuous design variables are described as follows and displayed in Figure 1.4 and 1.5:

- *Density approach*: The material distribution is discribed locally and fictitious porous materials are introduced to continuously transition between two or more materials. The focus is on the voxels, where it is decided what the structural properties are.

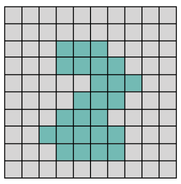


Figure 1.4: Density approach

- *Level set approach*: The geometry of material interfaces is explicitly described via the iso-contours of a level set function. The focus is on the boundary, and from every element it is known on which side of the boundary it is. See also Dijk et al. (2004) [84].

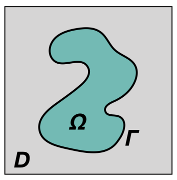


Figure 1.5: Level set approach: D is the design domain, Ω is the object domain, and Γ is the boundary

In this thesis project the level set approach is used, in order to use an explicit boundary representation needed for FSI, but also multiple implementations from the density approach will be utilized. Within the level set approach, there are three main methods to discretize the physical model. They are shown in Figure 1.6. In this thesis project the immersed boundaries on a fixed grid are used.

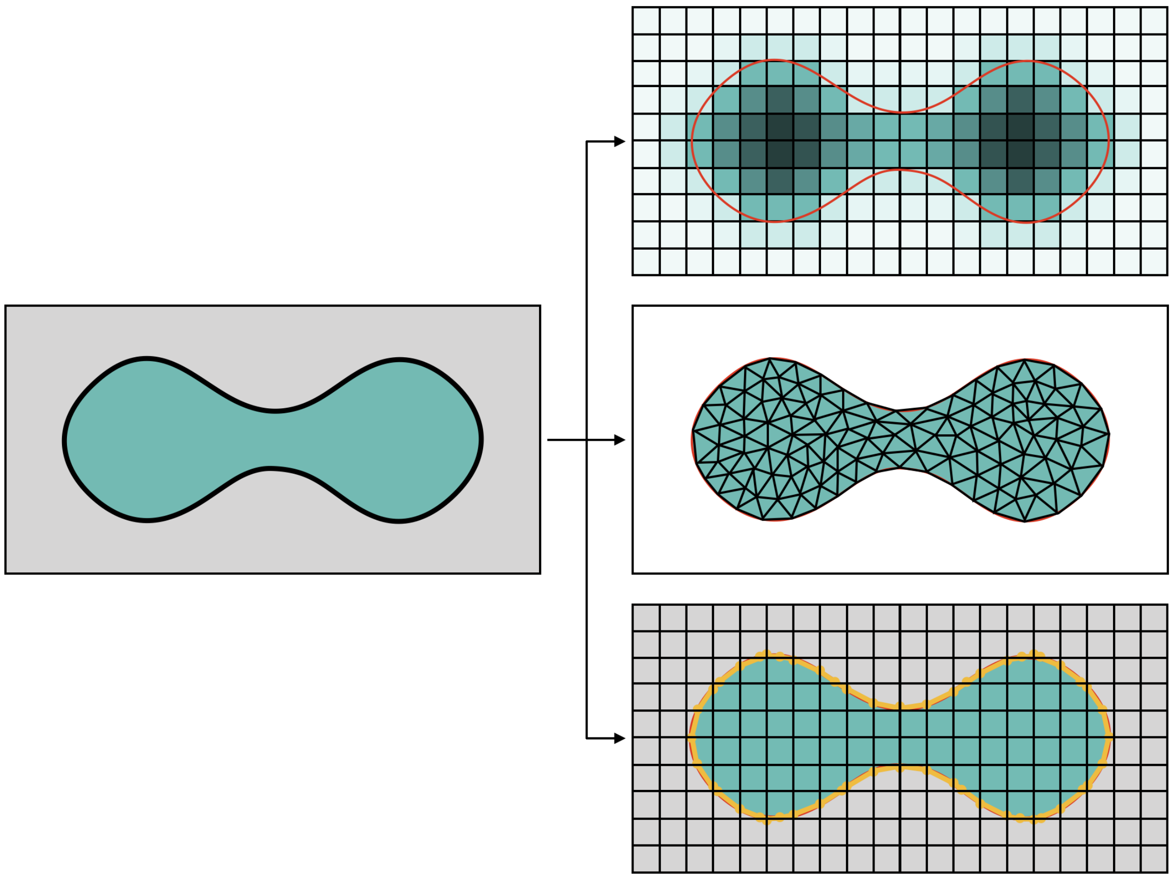


Figure 1.6: Three methods to discretize a physical model (based on [84]). Left: the object to be described. Top: fixed grid with density model / Ersatz material, the density indicates the amount of material. Middle: Body fitted model, where only the material domain is discretized. Bottom: fixed grid with immersed boundaries, the boundaries are enforced locally at the interface

There are different ways to describe the material phases within the design domain. This can be done with or without intermediate values. This is shown in Figure 1.7. In this thesis project the design field representation from the density method is used, so with intermediate density values.

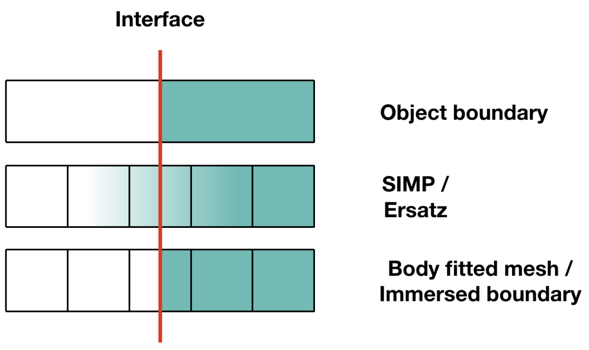
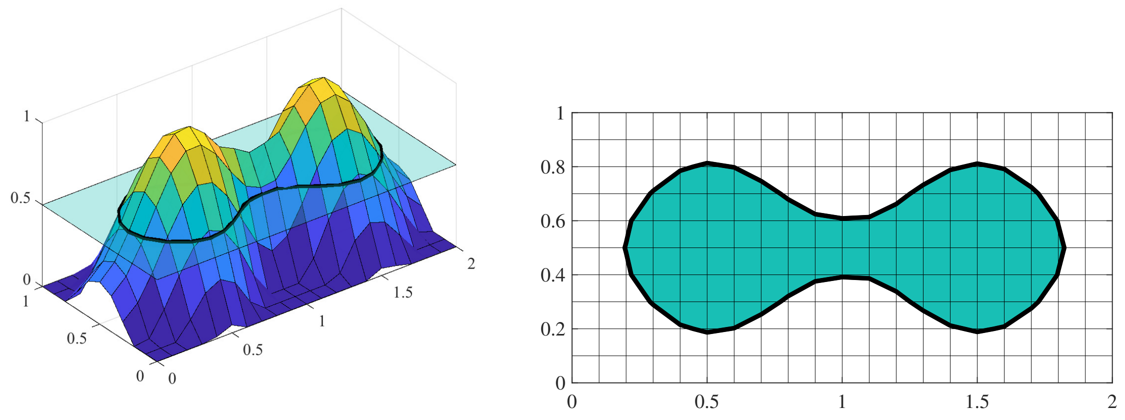


Figure 1.7: Geometry discretization methods

The combination of the immersed boundary, with the intermediate density values can be seen in Figure 1.8.

In Figure 1.8a, the density field is shown. The level set is at a value of 0.5, and all densities above are solid and all densities below are fluid. The created object model is shown in 1.8b.



(a) The density field, with the level set at 0.5 (b) The level set model

Figure 1.8: From density field to level set model

CutFEM:
CutFEM can be traced back to the eXtended Finite Element Method. The *eXtended Finite Element Method (XFEM)* is an immersed boundary technique that does not require a mesh that conforms to the phase boundaries and thus allows for discontinuities in structures. This method was built on the concept of partition of unity developed by Babuška and Melenk (1996) [63]. Its original use was to model crack propagation as described in Belytschko and Black (1999) [13]. For multiphysics problems with a coupled interface, the independent classical non-interface-fitted FEM approximations can be used for the participating phases. Cut elements were as a fictitious domain method introduced by Burman and Hansbo (2012) [26]. The *Cut Finite Element Method (CutFEM)* from Burman et al. (2014) [27] is the boundary technique where embedded boundaries or interfaces with different phases are cut-off. So the enrichment with of shape functions by the Partition-of-Unity method is not needed. Where in common Topology Optimization, the phase of elements are valid for the whole element, in CutFEM, the phase of are defined at every corner initially, and later in the Gauss points. A possible advantage of CutFEM is to make the discretization as independent as possible of the geometric description and minimize the complexity of mesh generation, while retaining the accuracy and robustness of a standard finite element method.
A good overview on XFEM is given Fries and Belytschko (2010) [40]. Research into CutFEM is done in for example [85] [28] [61].

Fluid Structure Interaction:
FSI can can be traced back to the origins of aerodynamics and more specific aeroelasticity. *Aeroelasticity* deals with the behaviour of an elastic structure in an airflow where there is signification interaction between the two. Friedrich Bessel studied the motion of a pendulum in a fluid in 1828 [17]. Research on aeroelasticity was driven by the developments in the aviation sector. In 1903 Professor Langley managed to control the aeroelastic loads created by lifting surface distortion, and in 1916 the first development in flutter was accomplished by scientist F. W. Lanchester.

In practice FSI mostly refers to numerical aeroelastical simulations. This is often related to coupling of Computational Fluid Dynamics (CFD) and Finite Element Method (FEM). Althought FSI was introduced for the aircraft industry, it is applicable in a large range of fields, for example airfoils [35] [37], engines [71], compressors [91], moving containers [56], the human blood flow system [42] [41], and the human lung system [82]. So, improving the performance of topology optimization on an FSI problem could be beneficial for a lot of fields.

1.2. Research objective

The research objective was: *"To investigate the potential for the usage of 3D CutFEM in topology optimization problems, compared to the use of finite elements (FE) with conventional geometry describing method."* The goal was to develop a code to perform topology optimization with CutFEM and identify in which cases it can be more beneficial to use CutFEM than classical methods.

In order to reach this objective, a topology optimization model based on the CutFEM method had to be created and validated. The first goal is to create a model that can perform a finite element analysis using CutFEM, which should be tested and validated. The second goal is to use this model to perform optimization. This optimization step is done by applying existing topology optimization methods to the CutFEM model. Finally, the model has to be validated, by comparing optimized structures to well known problems. In order to investigate the potential for the usage of CutFEM, different factors will be considered, among which the observed phenomena during the optimization process, and the outcome of the structural optimization.

The model will be created with PETSc software [11], so that it can be scaled for bigger structures. The used software is explained in Appendix A.

1.3. Thesis Structure

In Chapter 1, the background and need for this project is described. The usage of the Cut Element, which is the novelty of this project, is described in Chapter 2. The optimization functions that are used in this project, which are mostly taken from existing methods, are described in Chapter 3. The results of the tests are presented in Chapter 4, and discussed in Chapter 5. Chapter 6 contains the explanation and results for an implementation of CutFEM in a fluid solver. The final remarks, in the form of conclusions and recommendations, are given in Chapter 7.

2
Cut Elements

In this chapter the method to describe the surface with cut elements will be described. First, the background information on surface tracking is given. Next, the ambiguity in the marching cubes is described, in order to make clear how this is handled. This is followed by the the description of how the marching cubes are implemented in the model. Finally, the procedure to go from surface description in tetrahedrons to physical stiffness is explained.

2.1. Finite Element Analysis with Gauss Points

2.1.1. Finite Element Method

Hooke was in 1678 the first one to publish on displacement and forces, Bernouilli formulated the fundaments of elasticity, Euler got to solve one specific case which was illustrated by multiple examples, and Cauchy was the first to first to suggest the main ideas of the general theories of elasticity and hyper-elasticity valid for infinitely small strain. The formulations of elements for structural mechanics rely on long-established tools of stress analysis. In this section, the system equations will be derived that are used for the finite element method. For more information on the mechanics of materials, the reader is refered to for example Timoshenko (1951) [83], Cook (1974) [32], or Slaughter (2002) [79].

Potential energy:
To start, the potential energy in the domain Ω consists of the internal potential energy, and the external potential energy:

$$\Pi_{tot} = \Pi_{in} + \Pi_{ext} \tag{2.1}$$

Now the virtual work principle will be used. In order to find a stationary point, a set of displacements $\delta \mathbf{u}$ has to be found that gives a minimum in the potential energy. So the derivative with respect to this variable, should give that the change in potential energy is zero. That can be done as follows:

$$\delta\Pi_{tot} = \delta\Pi_{in}(\mathbf{u}, \delta\mathbf{u}) + \delta\Pi_{ext}(\delta\mathbf{u}) = 0 \tag{2.2}$$

Internal energy:
The internal energy on the body domain Ω can be described with the strain ϵ and stress σ:

$$\Pi_{in} = \{\boldsymbol{\epsilon}\}\{\boldsymbol{\sigma}\} \tag{2.3}$$

The derivative of the internal energy on the domain Ω can be described as follows:

$$\delta\Pi_{in}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} \{\delta\boldsymbol{\epsilon}\}\{\boldsymbol{\sigma}\} dV \tag{2.4}$$

The following equation gives two more known relations. Hooke's law is a law of physics that states that the force (F) which is needed in order to extend or compress a spring by a distance x is scaled linearly with respect to that distance. This is the *stress strain relation*, with the constitutive matrix C (often used with the Youngs modulus). Secondly, there is the *strain displacement relation* which uses the principle of interpolation (of displacement d with matrix B) anywhere in an element.

$$\begin{aligned}\{\boldsymbol{\sigma}\} &= [\mathbf{C}]\{\boldsymbol{\epsilon}\}\\ \{\boldsymbol{\epsilon}\} &= [\mathbf{B}]\{\mathbf{d}\}\end{aligned} \tag{2.5}$$

This gives that the derivative of the internal energy can be written as:

$$\delta\Pi_{in}(\mathbf{u},\delta\mathbf{d}) = \sum_e \int_\Omega \{\delta\mathbf{u}\}^T[\mathbf{B}]^T[\mathbf{C}][\mathbf{B}]\{\mathbf{d}\}dV \tag{2.6}$$

External energy:
The external energy can be described with the body force b and the traction force t of the Neumann boundary Γ_N:

$$\Pi_{ext} = \{\mathbf{b}\} + \{\mathbf{t}\} \tag{2.7}$$

The derivative of the external energy on the body domain Ω and the boundary domain Γ can be described as follows:

$$\delta\Pi_{ext}(\delta\mathbf{u}) = \int_\Omega \{\delta\mathbf{u}\}^T\{\mathbf{b}\}dV + \int_{\Gamma_N}\{\delta\mathbf{u}\}^T\{\mathbf{t}\}dS \tag{2.8}$$

Now, the *shape functions* are used in order to interpolate the displacements u to the displacements at any point in the element d.

$$\{\mathbf{u}\} = [\mathbf{N}]\{\mathbf{d}\} \tag{2.9}$$

This gives that the derivative of the external energy can be written as:

$$\delta\Pi_{ext}(\delta\mathbf{d}) = \int_\Omega \{\delta\mathbf{d}\}^T[\mathbf{N}]^T\{\mathbf{b}\}dV + \int_{\Gamma_N}\{\delta\mathbf{d}\}^T[\mathbf{N}]^T\{\mathbf{t}\}dS \tag{2.10}$$

Combined:
Combining the internal and external energy again gives:

$$\int_\Omega \{\delta\mathbf{d}\}^T[\mathbf{B}]^T[\mathbf{C}][\mathbf{B}]\{\mathbf{d}\}dV = \int_\Omega \{\delta\mathbf{d}\}^T[\mathbf{N}]^T\{\mathbf{b}\}dV + \int_{\Gamma_N}\{\delta\mathbf{d}\}^T[\mathbf{N}]^T\{\mathbf{t}\}dS \tag{2.11}$$

For the finite element method, this is summed up for all elements. So, discretization gives:

$$\sum_e \int_{\Omega_e} \{\delta\mathbf{d}_e\}^T[\mathbf{B}_e]^T[\mathbf{C}][\mathbf{B}_e]\{\mathbf{d}_e\}dV_e = \sum_e \int_{\Omega_e} \{\delta\mathbf{d}_e\}^T[\mathbf{N}_e]^T\{\mathbf{b}\}dV_e + \sum_e \int_{\Gamma_{N,e}}\{\delta\mathbf{d}_e\}^T[\mathbf{N}_e]^T\{\mathbf{t}\}dS_e \tag{2.12}$$

In here, $\{\delta\mathbf{d}_e\}^T$ can be crossed out. $\{\mathbf{d}_e\}$ can be taken out of the integral. Next, the element stiffness matrix $\mathbf{k}_e$ and the element load vector $\mathbf{f}_e$ are given by:

$$\begin{aligned}\int_{\Omega_e}[\mathbf{B}_e]^T[\mathbf{C}][\mathbf{B}_e]dV_e &= [\mathbf{k}_e]\\ \int_{\Omega_e}[\mathbf{N}_e]^T\{\mathbf{b}\}dV_e &= [\mathbf{f}_e]\end{aligned} \tag{2.13}$$

An assembly of $\{\mathbf{d}_e\}$, $\mathbf{k}_e$, and $\mathbf{f}_e$ gives that the finite element equation can be written as follows:

$$\mathbf{KD} = \mathbf{F} \tag{2.14}$$

2.1.2. Gauss Points

The stiffness of an object can be approximated and described in a stiffness matrix. This stiffness matrix can be calculated by using Gauss Quadrature. Gauss points are used to represent where material is present, and with that the stiffness of the element. The stiffness can be calculated by the integration over an isoparametric space with Gaussian quadrature, as shown in Equation 2.15. In here, $[J]$ is the Jacobian matrix, in order to change coordinates, like a projection to the isoparametric cube.

$$[\mathbf{k}_e] = \int_{\Omega_e} [\mathbf{B}_e]^T [\mathbf{C}] [\mathbf{B}_e] dV_e = \sum_{-1}^{1}\sum_{-1}^{1}\sum_{-1}^{1} [J] [\mathbf{B}_e]^T [\mathbf{C}] [\mathbf{B}_e] d\xi d\eta d\zeta \approx \sum_{GP} [\mathbf{B}_e]^T [\mathbf{C}] [\mathbf{B}_e] dV_e \tag{2.15}$$

How can Gauss points represent a Cut Element. Figure 2.1 shows different configurations of Gauss points in 2D. One element can be described by 4 Gauss points, see Fig. 2.1a. The Gauss points can also be calculated for a transformed element, see Fig. 2.1b. Gauss points can also be used in triangles (or tetrahedrons) which combined can make describe another shape, see Fig. 2.1c. In a Cut Element, each side can be described by their own Gauss points, where the weight of the Gauss points contains the information on the material, see Fig. 2.1d. Cut Elements can often not be described by rectangular cuboids, and can therefore be described nicely with tetrahedrons, see Fig. 2.1e. In Equation 2.15, the stiffness matrix is can be calculated with the sum of all the Gauss points. Then, all Gauss points have their own weight, which is in line with the amount of material that they represent. The material properties of the Gauss point are included in $[\mathbf{C}]$ (in the case of this project the Youngs modulus).

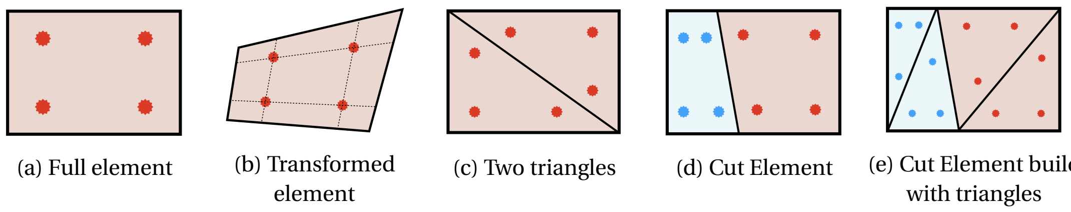


Figure 2.1: Gauss Points in elements

2.1.3. Process

Firstly, in order to make sure the the CutFEM procedure is only applied at the elements with multiple phases in the nodes, the Cut Elements are marked. This prevents unnecesary computions in element with a single phase.

Next, the process of finding the Gauss points in the Cut Element is shown in Figure 2.2. The starting point is the Cut Element where the boundary is known. How the boundary is found will be described in Section 2.3.2. Next, the element is tetrahedralized, so that it can be described by tetrahedrons only. This process will be described in Section 2.16. Next, the boundary points have to be found. Therefore, all the tetrahedrons are turned into standard tetrahedron elements, by changing the coordinates, so that the Gauss points can be assigned. These Gauss points are mapped back into the global coordinate system. This process will be described in Section 2.3.4. Now the Cut Element is described in Gauss points and the structural properties are known.

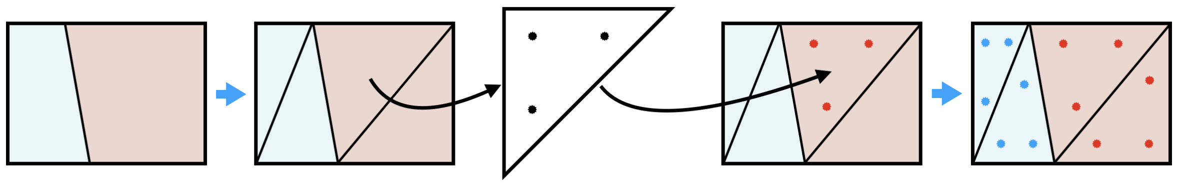


Figure 2.2: The scheme for finding the Gauss points.

2.2. Surface Tracking with Marching Cubes

In CutFEM the method for surface tracking is an important field to consider. Wenger (2013) [89] can be used as a good reference work on isosurfaces. The four approaches for isosurface construction are: (1) partitioning volumetric data into two-dimensional slices, construct isocontours in each slice, and then "stitch" together the slices using triangles, (2) partitioning space into cubes and associate each cube with a scalar value, (3) marching cubes, (4) and dual contouring. For the advantages and dissadvantages of each method, the reader is referred to Wenger. There, the marching cubes are named to be the most popular and this is also the method that will be used in 3D CutFEM. *Marching cubes* were introduced by Lorensen and Line (1987) [59] to create triangle models of constant density surfaces from 3D medical data, see Figure 2.3. The algorithm can be used for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field.

Marching cube (MC) were introduced by Lorensen and Line (1987) [59] to create triangle models of constant density surfaces from 3D medical data. The algorithm can be used for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field. It is probably the most commonly used method for isosurface extraction in scientific visualization ([88]).

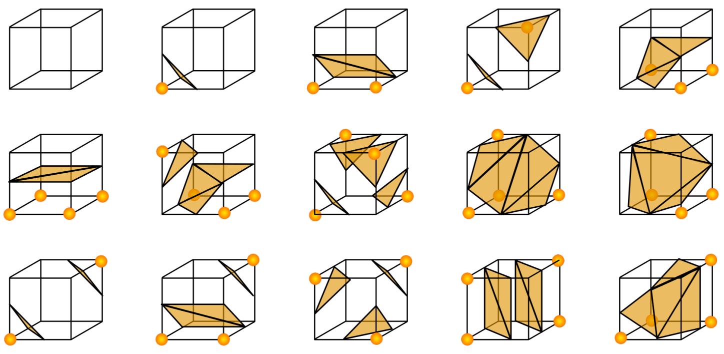


Figure 2.3: The originally published 15 cube configurations, from [89]

One important phenomenon that should be mentioned is ambiguity. This can occur in two manners which will be explained below. The idea behind the used convention is that it does not matter too much what convention is used, as long as it is consistent.

2.2.1. Ambiguity in material connection

A cube with all fluid nodes at one facet, and all solid nodes at the other facet can just be cut in two equal halves. However, if the solid nodes are not next to each other (connected by one direct edge only) the question can arise whether the solid nodes are actually connected by material. Figure B.1 shows an example of this. Chernyaev was in 1996 the first to address this problem and showed that there are actually 33 topologically different configurations [29]. This can also be seen in Figure B.1in Appendix B. His paper did not propose one single configuration on how to handle the ambiguity.

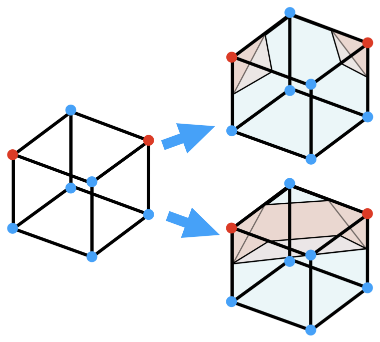


Figure 2.4: The ambiguity in Marching Cubes. The red space can be connected or can not be connected.

The convention used in this paper is that the fluid part is always connected. In Figure 2.5, it is shown that the solid parts would not be connected, but for the same layout the fluid parts would be connected. In Figure B.1 in Appendix B all possible ambiguity cases are shown. In there, if a solid node is represented by the marked vertex, then the cubes in this project are described by configuration `II`. Elements with multiple cuts are called dubbelcut elements. This convention was chosen so that dubbelcut elements would have a lower stiffness. Note that this convention causes a lower stiffness for dubbelcut elements, but also a lower amount of material usage.

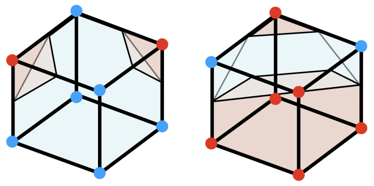


Figure 2.5: Used convention for material connection, the fluid parts are connected. Blue nodes are fluid, red nodes are solid, light blue is fluid space, light red is solid space

2.2.2. Ambiguity with an unequal cut

A cube with all fluid nodes at one facet, and all solid nodes at the other facet can just be cut in two equal halves. However, if it is required that the cut location is not in the middle between the solid and the fluid node, the question can arise where the cutting edge is exactly. If the 4 cut locations are not in one plane, the cutting edge has to be described by for example 2 triangles, as can be seen in Figure 2.6a. This ambiguity can cause problems if cutting edge of the solid and part are described separately (by tetrahedralization). It can happen that both parts will try to have the biggest shape and this does not match, as can be seen in Figure 2.6b.

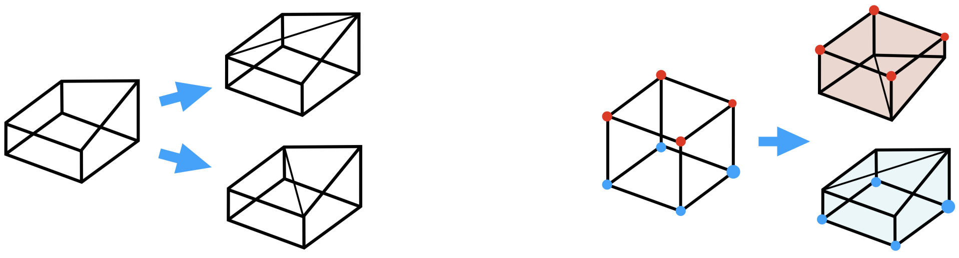

(a) Possible ambiguity in Marching Cubes for an edge that is not in one plane (b) Cutting an element if the edge surface is not described could lead to ambiguity

Figure 2.6: Ambiguity with an unequal cut

The convention used in this project is that the cut is always in the same direction, as shown in Figure 2.7. It does not matter whether the fluid of solid part will be bigger. The triangles used to describe the edge are always connected to the same points at the cutting edge. This convention was chosen so that the cut does not change with a small perturbation of a node value. The risk of this convention is that it is directional so that a different stiffness is obtained dependent on the location of a perturbed node value.

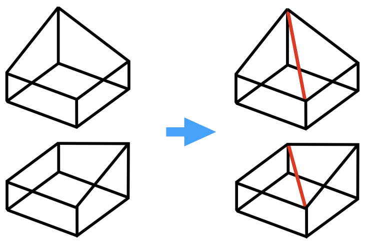


Figure 2.7: Convention in Cut Elements. For one type of cut, the cutting edge is always in the same direction, independent of phase or cutting locations. Note that the stiffness for these elements will be different.

2.3. Calculation of Cut Element stiffness matrix

In this section, it will be described how the stiffness matrix is computed for a Cut Element. First, the preparation steps and the process will be explained. Next, the it will be described how to cut the element. Next, the steps in the tetrahedralization of the Cut Elements are explained. And finally, the process of mapping of the Gauss points from the tetrahedrons to the Cut Element is shown.

2.3.1. Process

The global process with which stiffness matrix of Cut Element is found, can be seen in Figure 2.8. First, the edges that will be cut are marked (marked with a cross). Based on the phase values at the nodes, the cut point locations can be calculated. Now all the information is known to describe the cut elements in tetrahedrons. This process is done with a programm called TetGen, which will be explained in the next section. The solid part and the fluid part are separately put into TetGen. This is because solid parts and fluid parts are treated differently, as described in Section 2.2.1, and in order to keep track of the phases. First the information on the point locations is put in. Next, the information on the outside facets is given. This information is needed because the cutting edge has to be set, as described in Section 2.2.2. TetGen gives back the geometry of all the tetrahedrons that can describe the Cut Element. Now, the Gauss point information can be calculated, so the location and the weight. Combining the Gauss points of all tetrahedrons gives the stiffness of the Cut Element.

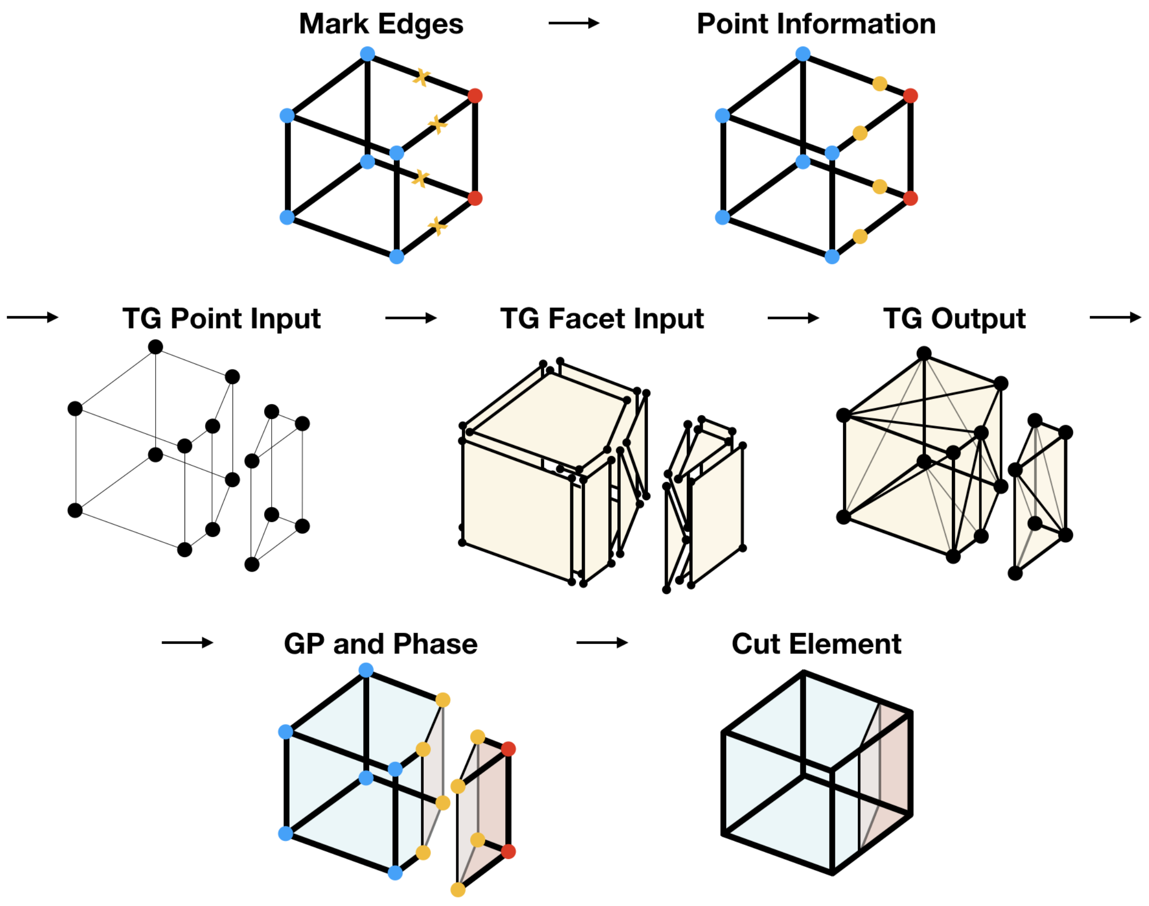


Figure 2.8: Process for tetrahedralizing. TG is TetGen, GP is Gauss points.

2.3.2. Determining the cut

In order to determine the cutting edge of the Cut Element, two things have to be known. The edges that have to be cut, and the exact cut location on that edge. The edges that have to be cut are determined with the cube index. The exact cut location are is determined with the phase value of the nodes.

Cube index:
A cube has 8 nodes, which all can be 2 different phases. This results in $2^8 = 256$ unique cubes. Some conventions have to be used to be consistent in the cube numbering. Firstly, the Figure 2.9 part a) shows the convention on the numbering of the nodes. This cube can be displayed in any direction, but the connection between the node numbers is important. Secondly, all nodes can be of state 0 or state 1. Thirdly, the numbers 0 until 255 will be used, and those can be written in a binary manner. Cube number 1 would be `00000001` and cube number 185 would be `10111001`. Now, by looping over all 8 nodes, and identifying the state, the cube index can be found.

Next, a lookup table can be used. Such a table can contain pre-computed information for all 256 cubes, which edges are cut, and how many parts the cut element is split up in.

```
      4--------5      *---4----*        4---12---5
     /|       /|     /|       /|       /|       /|
    / |      / |    7 |      5 |     15 |      13|
   /  |     /  |   /  8     /  9     /  16    /  17
  7--------6   |  *----6---*   |    7----14--6   |
  |   |    |   |  |   |    |   |    |   |    |   |
  |   0----|---1  |   *---0|---*    |   0---8|---1
  |  /     |  /  11  /    10  /    19  /    18  /
  | /      | /    | 3      | 1      | 11     | 9
  |/       |/     |/       |/       |/       |/
  3--------2      *---2----*        3---10---2
```

Figure 2.9: The convention on the numbering a) the node numbering b) the edge numbering c) new convention for both nodes and edges

Cut location:
In most other applications, the cut is exactly in the middle of two nodes, as the 'phase' is described in a discrete manner, either 0 or 1. For the CutFEM application, the phase is continuous (in a the range `[0, 1]`). Therefore, the cut location is not in the middle between two nodes, but based on phase values. This is done linearly, according to Equation 2.16 which is illustrated in Figure 2.10. For most applications, the cut value is set to be exactly between the maximum and minimum phase values.

$$x_{cut} = x_{node1} + (x_{node2} - x_{node1}) \cdot \frac{\phi_{node1} - \phi_{levelset}}{\phi_{node2} - \phi_{node1}} \tag{2.16}$$

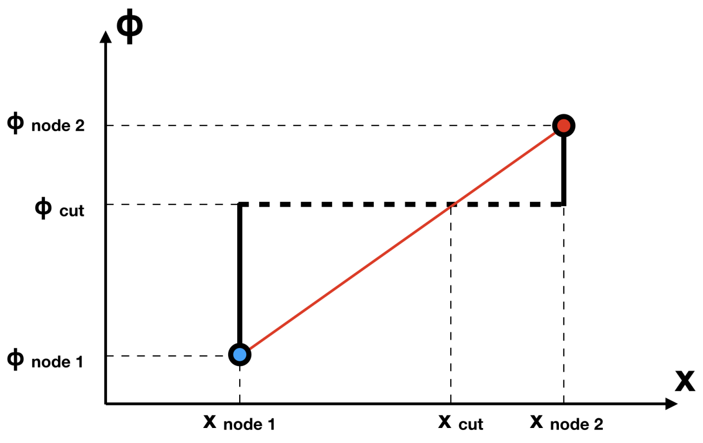


Figure 2.10: Cut location.

2.3.3. Tetrahedralization

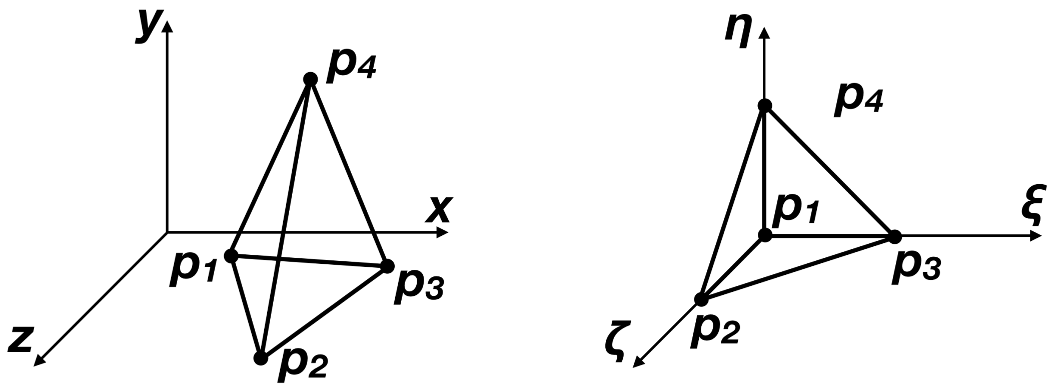
The tetrahedralization is performed by external software TetGen. TetGen is a program to generate tetrahedral meshes of any 3D polyhedral domains by employing a form of Delaunay triangulation (see Si 2013 [73] and Si 2015 [74]). The *Delaunay triangulation* is named after Boris Delaunay (1934) [33], and maximizes the minimum angle of all the angles of the triangles in the triangulation of a given set of discrete points. This is done by ensuring that no point is inside the circumcircle of any triangle in the triangulation.

All parts (one fluid, and one or more solids) are put in separately into TetGen, in order to make sure the cutting edge is obeyed, and keep track of the phase of the tetrahedron that come out. The first input are the coordinates of all points. The obtaining of this information was described above. The second input is the information on the outer facets of the part. That is the number of facets, and a list with the point numbers that form each facet. This is obtained with a lookup table. The lookup tables are described in Section B.2 in Appendix B. The last input are the settings that TetGen should use. The used settings are *"pp/0.0000001YT0.00000000000000000001Q"*. The exact meaning of these settings is described in Section B.3 in Appendix B. This input makes sure that the output tetrahedrons obey the outer boundary (in order to prevent the ambiguity in section 2.2.2). Furthermore it prevents error messages from TetGen about unreliable tetrahedron shapes. On element level, the accuracy of the tetrahedrons is considered of less importance then the continuation of the code.

The output from TetGen is a list with point coordinates, and a list with the point numbers that form each tetrahedron. This information is saved, and the phase information for the tetrahedron is added, as it is known whether the tetrahedralized part was solid or fluid.

2.3.4. Gauss point projection

The calculation of the Gauss points of the tetrahedrons can be significantly simplified by carrying out a coordinate transformation to a reference system, as can be seen in Figure 2.11. Each point (x, y, z) of the tetrahedron in the original coordinate system can be mapped to a corresponding point (ξ, η, ζ) in the reference coordinate system (with outer size $1x1x1$).



(a) Original coordinate system (b) The reference tetrahedron

Figure 2.11: Tetrahedral finite element

Next, shape functions are needed in order to describe the tetrahedron. The *shape functions* are the functions which interpolates the solution between the discrete values at the mesh nodes. If one would for example like to know the displacement, at any given point within the tetrahedron, this can be calculated with the displacement of the 4 corner points, which all have their own contribution. It is important that these contributions do not cause any scaling, so at any given point the sum needs to be 1:

$$\sum_{i}^{N} N_i(\xi, \eta, \zeta) = 1 \tag{2.17}$$

The nodal basis functions for the tetrahedron in the reference coordinate system are given by:

$$
\begin{aligned}
N_1^t(\xi,\eta,\zeta) &= 1-\xi-\eta-\zeta \\
N_2^t(\xi,\eta,\zeta) &= \xi \\
N_3^t(\xi,\eta,\zeta) &= \eta \\
N_4^t(\xi,\eta,\zeta) &= \zeta
\end{aligned}
\tag{2.18}
$$

The shape function interpolation for the original coordinate system is now as follows:

$$
\begin{aligned}
x &= \sum_i^N N_i(\xi,\eta,\zeta)x_i = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta + (x_4 - x_1)\zeta \\
y &= \sum_i^N N_i(\xi,\eta,\zeta)x_i = y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta + (y_4 - y_1)\zeta \\
z &= \sum_i^N N_i(\xi,\eta,\zeta)x_i = z_1 + (z_2 - z_1)\xi + (z_3 - z_1)\eta + (z_4 - z_1)\zeta
\end{aligned}
\tag{2.19}
$$

In here x, y, and z are the coordinates in the original coordinate system of any point in the tetrahedron, and for example x_1, y_1, and z_1 are the coordinates of point P_1 in Figure 2.11a. Filling in the corner points of the tetrahedron (at x_1, y_1, z_1) and Gauss point locations of the reference tetrahedron (at ξ, η, ζ) gives the coordinates of the Gauss points in the original coordinate system (at x, y, z). The Gauss point locations of the reference tetrahedron can be found in Table 2.1. 4 Gauss points are used, as the tetrahedrons used in this project are linear.

Table 2.1: Gauss points in a tetrahedron (see for example Cook et al. 1981 [32])

Point	ξ	η	ζ
1	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$
2	$\frac{5+3*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$
3	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5+3*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$
4	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5-1*\sqrt{5}}{20}$	$\frac{5+3*\sqrt{5}}{20}$

Now the coordinates of the Gauss points of all tetrahedrons are known in the original coordinate system of the cut element. Next, in order to perform the integration from Equation 2.15 Gauss points have to be mapped back into a isoparametric cube. Because the original coordinate system of the cut element that was put into TetGen is in the domain `[0, dx]`, and the integration is done on a cube with the domain `[-1, 1]`. This mapping is done linearly.

$$
\mathbf{x}_{map} = \mathbf{x} * \frac{2}{dx} - 1 \tag{2.20}
$$

Before conclusion, the weight of the Gauss points should be considered. First, the weight for the size of the cut element itself is given by the determinant of the Jacobian. Next, the weight of each point is relative to the volume of the tetrahedron V_{tet} in the cut element V_{elem}. The Gauss points in the reference tetrahedron all have weight 0.25, while the Gauss points in a isoparametric cube have weight 1. The domain of the isoparametric cube is 8 times larger than the domain of the reference tetrahedron. Therefore, the weight of each Gauss point is given by:

$$
W_i = \det[J(\xi,\eta,\zeta)] * \frac{V_{tet}}{V_{elem}} * \frac{0.25}{1} * \frac{8}{2} \tag{2.21}
$$

Now all the features for the finite element analysis model are there. This is the first goal of the research objective. This model is tested and the results are shown in Section 4.1.

3 Methods for Optimization

In this chapter the methods are described which are used to perform the actual optimization. The goal is to answer the question of how the optimal solution changes as the problem parameters are changed. In order to answer this question without having to solve the optimization problem again, sensitivities are used.

First, the general optimization problem is explained, after which the specific topology optimization problem is explained. Next, the calculation of the objective and constraints are explained. This is followed by the calculation of the filter. To perform the actual optimization, the Method of Moving Asymptotes (MMA) is used. Finally, a dilated and eroded method is used to create a better design.

3.1. Optimization

The optimization problem is the problem of finding the best solution from all feasible solutions. Firstly, the *objective* must be identified, this is a quantitative measure of the performance of the system. This objective could be for example profit, time, or a combination of variables that can be represented by one single number. The objective either has to be optimised, which can either be maximised or minimised. As the standard form is to *minimize*, it might be needed to either multiply the objective with -1 or divide 1 over the objective, if that provides the required optimisation. The objective is dependent on *variables*. The goal is to find the variables that optimise that objective. That is why these can also be called *unknowns*. The variables are *restricted* or *constrained*, which limits the set of variables that is allowed for. See Figure 3.1 for a visualisation.

Now, the standard form a an optimization problem which is subject to constraints on the variables can be described as follows (see Boyd and Vandenberghe 2004 [22] or Nocedal and Wright 2006 [68]):

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}}: \quad & f(x) \\ \text{subject to}: \quad & g_i(x) \leq 0, \qquad i=1,\dots,m \\ & h_i(x)=0, \qquad i=1,\dots,p \end{aligned} \tag{3.1}$$

in which:

- x is the vector of *variables*, also called *unknowns* or *parameters*
- $f:\mathbb{R}^n \to \mathbb{R}$ is the *objective function*, which is to be minimised over the n-variable vector x. $\mathbb{R}$ is a set of all real numbers.
- $g_i(x) \leq 0$ are called *inequality constraints*, which define certain inequalities that the vector x must satisfy
- $h_j(x) = 0$ are called *equality constraints*, which define certain equations that the vector x must satisfy

In some cases, a third type of constraint is considered, the *box constraint*. This is a double inequality constraint, where a variable has to be in between.

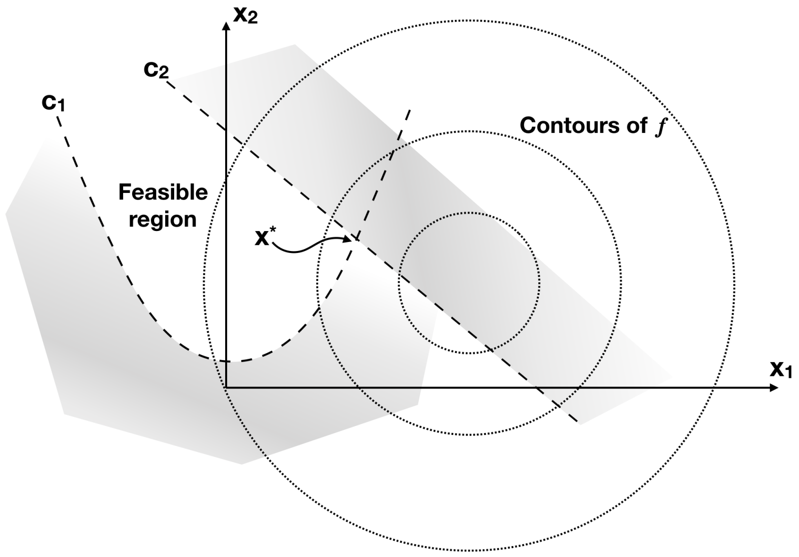


Figure 3.1: Geometrical representation of an optimization problem, the constraints c set the feasible region, x_1 and x_2 are the variables, f shows the objective function contours, and x^* is the optimal combination of variables.

Now, the Lagrangian function for the optimisation problem in Equation 3.1 is as follows:

$$\mathscr{L}(x,\mu,\lambda) = f(x) - \mu_i g_i(x) - \lambda_i h_i(x) \tag{3.2}$$

In here, μ and λ are the Lagrange multipliers for the inequality and equality constraints, respectively. The Lagrangian can be seen a the original objective function in combination with a weighted sum of the constraint functions.

The *Karush–Kuhn–Tucker (KKT)* conditions are the necessary conditions for a solution in nonlinear programming to be optimal. The KKT conditions are the foundation for many of the optimization algorithms, and can be defined as follows:

$$\begin{aligned} \text{Stationary:} \quad & \frac{\partial \mathscr{L}}{x_j} = \frac{\partial f}{\partial x_j} - \mu_i \frac{\partial g_i}{\partial x_j} - \lambda_i \frac{\partial h_i}{\partial x_j} = 0 \\ \text{Primal feasibility:} \quad & g_i \leq 0 \\ \text{Primal feasibility:} \quad & h_i = 0 \\ \text{Dual feasibility:} \quad & \mu_i \geq 0 \\ \text{Complementary slackness:} \quad & \mu_i g_i = 0 \end{aligned} \tag{3.3}$$

3.1.1. Minimum Compliance

For a topology optimization problem, different interpretations are possible for the objective function Ψ, for example maximize stiffness, minimize mean compliance, or minimize stored mechanical energy. The design vector x can be for example dimension, topology, shape, or material.

The goal of the optimization problem is to maximize stiffness. A suitable measure for stiffness is compliance, so the used objective function is minimum compliance. Compliance is the property of a material of undergoing elastic deformation or change in volume when subjected to an applied force. In other words, how much does the structure move along with the force on it. It is equal to the reciprocal of stiffness. For a simple truss this is given by $\mathbf{F}^T\mathbf{u}$. For more complex structures the compliance is given by multiplying the displacement with the load: $C = \{D\}^T\{F\} = \{D\}^T\{K\}\{D\} = \sum_e \{d^e\}^T [k^e] \{d^e\}$. This leads to the following objective function:

$$\min_s \quad \Psi(u(s)) = \min_s \quad f^T u(s) = \min_s \quad u(s)\widetilde{K}(s)u(s) \tag{3.4}$$

The optimization problem is as follows:

$$\begin{aligned} \underset{s\in\mathbb{R}^n}{\text{minimize}}: \quad & \Psi(\mathbf{s},\mathbf{u}) = \mathbf{u}^T\mathbf{K}(\mathbf{s})\mathbf{u} \\ \text{subject to}: \quad & \mathbf{K}(\mathbf{s})\mathbf{u} = \mathbf{f} \\ & \frac{V(s)}{V^*} - 1 \leq 0 \\ & 0 \leq s_i \leq 1 \end{aligned} \tag{3.5}$$

In here, the equality constraint is the finite element equation from Equation 2.14. The inquality constraints are the upper and lower bounds of the design variable s_i, so the phase ϕ, which goes from 0 to 1. The other inequality constraint, should make sure that the material fraction of the structure $V(s)$ is less than the allowed volume fraction V^*.

3.1.2. Process

The global process with which the optimization will happen can be seen in Figure 3.2. First, the input parameters and initial design are defined in the initialisation. The design is filtered in order to improve the layout. Next, a finite element analysis is performed on this design, which measures the performance of this design. A sensitivity analysis is performed, in which the influence of the design parameters is measured. For this project that is the phase value at the nodes. The sensitivities for both the objective function and the constraint function are calculated. Then, the algorithm will calculate how the design can be optimized. Now, the design will be updated. The optimization cycle can start again with this new design. If the design does not improve anymore, so the final design is reached.

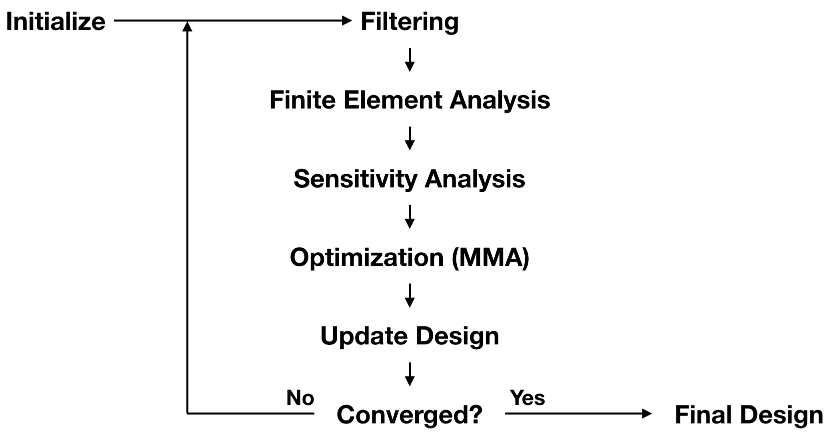


Figure 3.2: Scheme for optimization

3.2. Objective and constraints

This section will elaborate on the calculation of the objective and the constraints. For both, also the sensitivities are calculated. The adjoint method behind the sensitivities is explained in Appendix C.

The calculation of the objective function `fx`, and the volume constraint `gx` can directly be done after the system has been solved. The calculation of the objective function consists of a matrix multiplication of the `Uvec` and the `Kmat`, followed by a multiplication in a vector dot product with `Uvec`. The objective function is calculated as follows:

$$\texttt{fx} = UKU \tag{3.6}$$

The volume constraint is calculated as follows:

$$\texttt{gx} = \frac{V(s)}{V^*} - 1 \tag{3.7}$$

The sensitivities are calculated for the design variables, so the phase ϕ. This is done with the adjoint method, which is explained in Appendix C. For the calculation of the objective function sensitivities `dfdx` and the volume constraint sensitivities `dgdx`, a small perturbation is done in the phase of each node separately and the effect is checked, see Figure 3.3. Only the influence in 1 element is calculated, and this is added to the total sensitivity vector. This is only done in the cut elements, as a perturbation in other elements would not have an influence. The sensitivities are calculated with $Sens = \frac{f^{pert} - f}{h}$, where f^{pert} is the perturbed value, f the original value, and h the perturbed value.

In the code it is important that the tetrahedralization of the element does not change when a node is perturbed. When the element is initially tetrahedralized, the geometry connection and geometry coordinates are saved. One node is then perturbed and only the new geometry coordinates are changed.

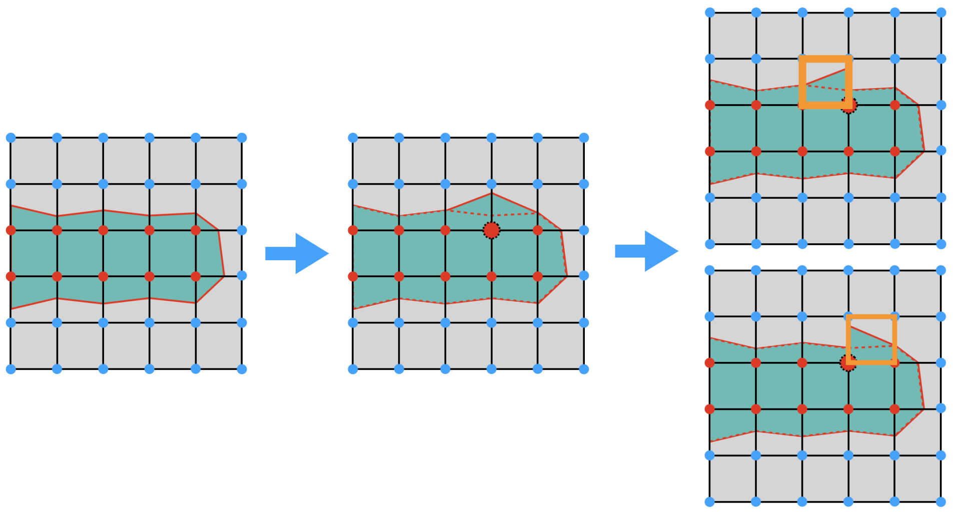


Figure 3.3: The adjoint method. *Left* is the design, *middle* is the new design if a node value is perturbed, *right* is the adjoint method which only calculates the stiffness in one element at the time, instead of calculating the stiffness of the whole structure

In the calculation of the objective function sensitivities `dfdg[i]` is related to the sensitivity in one node, which is calculated with by summing up the contributions of all neighbouring elements. For the calculation of the sensitivities of the objective function, a minus sign is needed, this is explained in Appendix C.

$$\texttt{dfdx[i]} = -\sum_e \mathbf{u}_e^T \frac{([\mathbf{k}_e^{pert}] - [\mathbf{k}_e])}{h} \mathbf{u}_e \tag{3.8}$$

As the volume constraint is scaled by the allowed volume, the sensitivities of the volume constraint also have to be scaled with the allowed volume. The volume constraint sensitivities are calculated as follows:

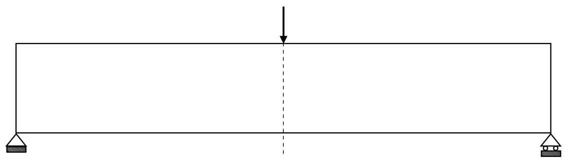
$$\texttt{dgdx[i]} = \frac{\sum_e \frac{V_e^{pert} - V_e}{h}}{V^*} \tag{3.9}$$

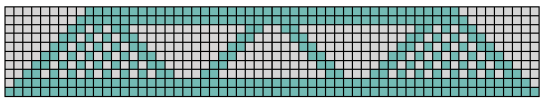
The performance of the sensitivities is checked with a finite difference test and is presented in Appendix D.

3.3. Filter

As stated in Bendsoe and Sigmund (2003) [14], there are some issues that significantly influence the computational results. One example for the density method is the *checkerboard*. This means that regions of alternating solid and fluid elements are formed ordered in a checkerboard like fashion. This pattern leads in the computation to a lower objective, but can be unphysical or undesirable for manufacturing. The checkerboard problem was first discussed in Bendsoe et al. (1993) [16], Jog et al. (1993) [52] and Rodrigues and Fernandes (1993) [70], and it detailed analysis can be found in Diaz and Sigmund (1995) [34] and Jog and Haber (1996) [53]. In order to prevent the checkerboard, techniques like *higher-order finite elements*, *patches*, and *filtering* can be used. For a full comparison, the reader is referred to a review by Sigmund and Peterson (1998) [75].

Also in the level set method, the checkerboard issue can occur (see Dijk et al 2013 [84]).


(a) The design problem


(b) The checkerboard solution

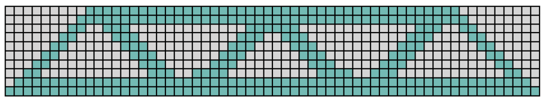

(c) The solution with rough discretization

Figure 3.4: Checker-boarding

Level set topology optimization with CutFEM also has to deal with specific numerical instabilities. Makhija and Maute (2004) [60] have described this for XFEM. The given reason for these instabilities can be explained with Figure 3.5, where two pieces of material are separated by void. If the material is forced into a displacement away from each other, the reaction force should be zero, as the bars are not physicaly connected. However, the computation stiffness does show a connection, and a reaction force will be present.

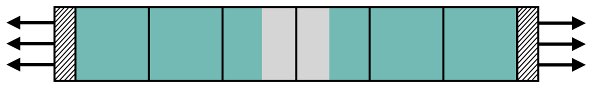
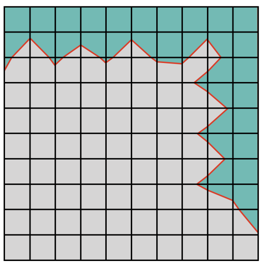


Figure 3.5: Example problem

In the design of a structure, this can lead to issues in the design which are artificialy good, but not physical. Two examples are shown in Figure 3.6.


(a) Artificial stiffness across void domains is computed

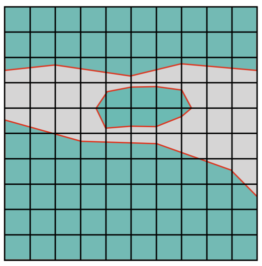
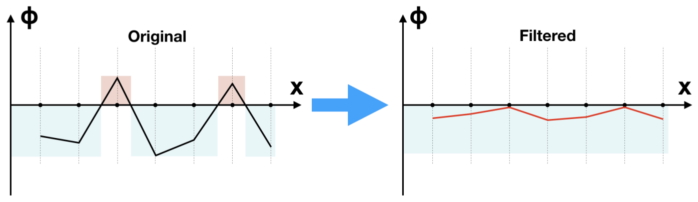

(b) “Free-floating” material is created

Figure 3.6: Checker-boarding issues

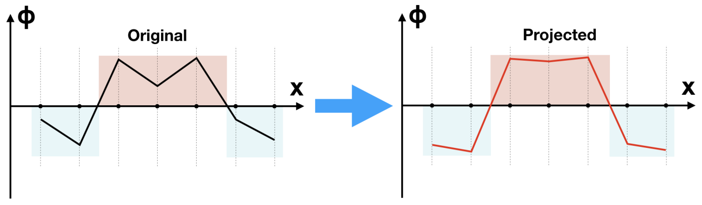
One solution that helps avoiding these issues is filtering. There is both *sensitivity filtering* and *density filtering*. The idea of the sensitivity filter is to use blurring techniques borrowed from image processing (see Pratt 1991 [69]). Element sensitivity values are modified to be weighted averages of their neighbors within a mesh-independent radius r_{min}. This method was introduced in Sigmund (1994) [76]. It is an efficient way to tackle the problem of mesh-dependency. The density filter defines the physical density of an element ρ as a weighted average of the design variables in a neighborhood of radius r_{min}. Density filters were first used in Bruns and Tortorelli (2001) [24] and were analysed in Bourdin (2001) [20]. Unlike the perimeter control and gradient control methods which have an extra (explicit) constraint on the density distribution, the filtering methods only limits the variations in sensitivity and material in elements. Furthermore it is easy to implement, and its implementation can even stabilize convergence. One disadvantage is that it is based on heuristics (not guaranteed to be optimal, but sufficient enough).

Now, the filtering is performed with three procedures, all with their own functionality:

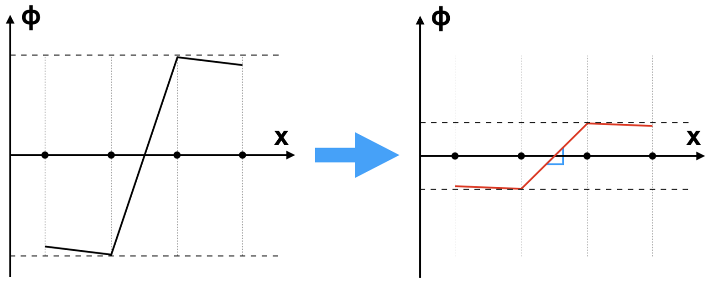
- Filtering: the node phase values become the weighted averages of their neighbors within a mesh independent radius r_{min}, see Figure 3.7a.
- Projecting: the average values close to the level set are projected with the Heaviside filter into more extreme values, see Figure 3.7b.
- Mapping: the values are mapped into a range that is adjusted to the physical distance between nodes, in order to have a smooth level set function, see Figure 3.7c.



(a) The original values (black) can be smootened (grey) which does not suppress small localized features, and filtered (red) which does suppress the small localized features.



(b) The projection brings the values closer to the maximum value



(c) Mapping is done in order to prevent ill-conditioned values

Figure 3.7: The three procedures of the filtering

That leads to the following variables which are added:

- `sVals` or S is the original. The values are in the range `[0, 1]`. This is the design that is being updated.
- `sFilt` or $\widetilde{S}$ is filtered. The values are in the range `[0, 1]`.
- `sProj` or $\overline{S}$ is projected. The values are in the range `[0, 1]`.

- sPhys or S_{phys} or S_p is physical (and is mapped). The values are in the range [-$\frac{dx}{2}$, $\frac{dx}{2}$]. This design is referred to as the physical design S_{phys} because the filter causes the original design S to loose its physical meaning. For example the volume constraint calculations will be done on sPhys rather then sVals. Therefore, always the filtered design S_{phys} should be presented.

In equation form, the filtering is presented as follows:

$$S \xrightarrow{filtering} \widetilde{S} \xrightarrow{projecting} \overline{S} \xrightarrow{mapping} S_p \tag{3.10}$$

All of the procedures mentioned before have an influence on the gradients (which will be covered in the next chapter). This means that the gradients have to be filtered/projected/mapped back. The gradients which are calculated are $\frac{\partial \phi}{\partial S_p}$, but the gradients which are needed are $\frac{\partial \phi}{\partial S}$. This requires the following procedure:

$$\frac{\partial \phi}{\partial S} = \frac{\partial \phi}{\partial S_p} \frac{\partial S_p}{\partial \overline{S}} \frac{\partial \overline{S}}{\partial \widetilde{S}} \frac{\partial \widetilde{S}}{\partial S} \tag{3.11}$$

3.3.1. Filter of the phases

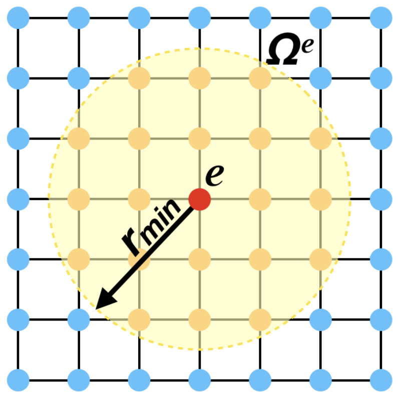
The computation of the parts in the filtering scheme in Equation 3.10 will be explained here.

Filter:
The filter function $S = Filter(\overline{S})$ uses a weighted average over different elements within a distance, as shown in Figure 3.8. In order to account only for the node values within the set radius H_{ei} is the weight factor or filter kernel for which the set of elements i for which the center-to-center distance $\Delta(e,i)$ to element e is smaller than the filter radius r_{min}:

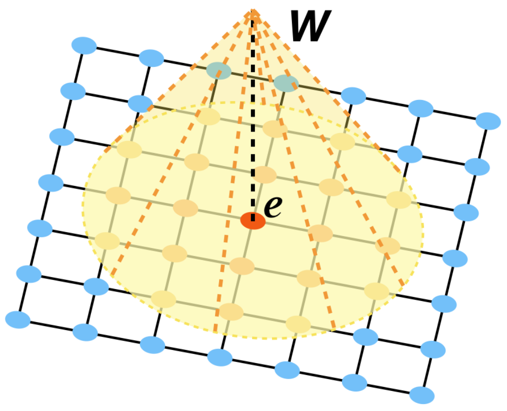
$$H_{ei} = \max(0, r_{min} - \Delta(e,i)) \tag{3.12}$$

The weighted average is computed in Equation 3.13. In this equation, H_{ei} are the weight coefficients, ϕ_i are the node values within domain Ω_w^e, and $\sum_{i\in N_e} H_{ei}$ are the normalization constants. As this is a linear operation, it can also be implemented as a matrix product, as shown on the right side. The coefficient matrix H establishes a relationship between all elements (within the bounded support of filter kernel). It is a matrix where one row belongs to one node. This row contains the weight values that its surrounding nodes have on this node. The column of the surrounding node belong also to one specific node each. As the sum of all these weights can differ there has to be divided by the sum of all weights. Hs can be seen as a vector with the for each node the sum of all weights on that node.

$$\widetilde{\phi}_e = \frac{\sum_{i\in N_e} H_{ei}\phi_i}{\sum_{i\in N_e} H_{ei}} \quad \Rightarrow \quad \widetilde{S} = \frac{H * S}{Hs} \tag{3.13}$$



(a) Nodes located inside the domain Ω_w^e are used in used in the projection scheme for element e



(b) The weight function W for the linear projection scheme

Figure 3.8: Filter

Heavyside:
In order to impose length scale on the optimized designs, the Heaviside projection was introduced in topology optimization by Guest et al. (2004) [48]. The original Heaviside projection was a step function giving 0 for input smaller than a threshold input value, and 1 for input equal or larger than a threshold input value. A smooth Heaviside function is displayed in Figure 3.9. A new Heavyside projection function was then introduced by Xu et all. (2010) [92], in order to alleviate bad convergence due to volume preservation issues. It is a generalization of the dilation projections by Guest et al. (2004) [48] and the erosion projections by Sigmund (2007) [77]. The Heaviside function of ϕ can be written as:

$$\overline{\phi}\begin{cases}\eta\left[e^{-\beta(1-\phi/\eta)}-(1-\phi/\eta)e^{-\beta}\right] & 0<\phi<\eta\\ (1-\eta)\left[1-e^{-\beta(\phi-\eta)/(1-\eta)}+(\phi-\eta)(1-\eta)e^{-\beta}\right]+\eta & \eta<\phi<1\end{cases} \tag{3.14}$$

Rewriting gives that the following Heaviside filter used in this project is as follows, with β is the steepness of the projection, and η is the threshold value:

$$\overline{S}=\frac{tanh(\beta\eta)+tanh(\beta(\widetilde{S}-\eta))}{tanh(\beta\eta)+tanh(\beta(1-\eta))} \tag{3.15}$$

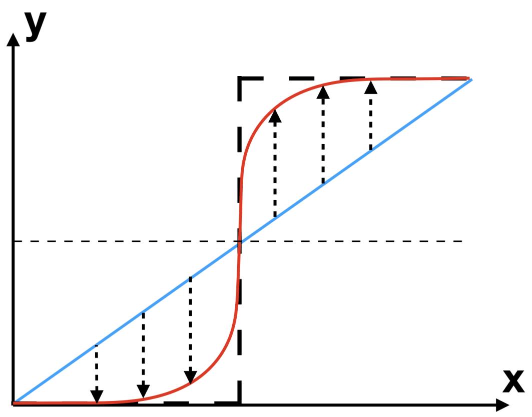


Figure 3.9: A Heaviside projection function, where x is projected to y. The original values (in blue) are projected (in red) away from the cut value. The dashed black line is the step function.

Mapping:
Finally, the mapping function $S_p = Mapping(\overline{S})$ is computed according to linear mapping. The resulting equation is as follows, where the S_{min} and S_{max} are the mesh size $-\frac{dx}{2}$ and $\frac{dx}{2}$ respectively.

$$S_p = S_{min}+\overline{S}*(S_{max}-S_{min}) \tag{3.16}$$

3.3.2. Filter of the gradients

After the analyses, the obtained gradients $\frac{\partial\phi}{\partial S_p}$ are filtered back. The computation of the part in the chainrule in Equation 3.11 ($\frac{\partial S_p}{\partial\overline{S}}$, $\frac{\partial\overline{S}}{\partial\widetilde{S}}$, $\frac{\partial\widetilde{S}}{\partial S}$) are explained below. Note that these are computed by taking the derivatives of the equations from the section above.

Mapping:
The gradient is calculated as follows:

$$\frac{\partial S_p}{\partial\overline{S}}=(S_{max}-S_{min}) \quad\Rightarrow\quad \frac{\partial\phi}{\partial\overline{S}}=\frac{\partial\phi}{\partial\overline{S}}(S_{max}-S_{min}) \tag{3.17}$$

Heavyside:
Similarly, the Heaviside function is obtained as follows:

$$\frac{\partial \phi}{\partial \tilde{\tilde{S}}} = \beta * \frac{1 - \sqrt{tanh\left(\beta * (\frac{\partial \phi}{\partial \tilde{\tilde{S}}} - \eta)\right)}}{tanh(\beta\eta) + tanh(\beta(1-\eta))} \tag{3.18}$$

Filter:
Finally, the influence of the filter is handled in the gradients as follows:

$$\frac{\partial \tilde{S}}{\partial S} = \frac{H}{Hs} \qquad \Rightarrow \qquad \frac{\partial \phi}{\partial S} = \frac{H \cdot \frac{\partial \phi}{\partial \tilde{S}}}{Hs} \tag{3.19}$$

3.4. Method of Moving Asymptotes (MMA)

Now that there is a design, and there are the sensitivities of the objective and the volume, the design has to be updated. One method to do that is *sequential quadratic programming (SQP)*. This was introduced in 1963 by Wilson [90], after which Fletcher proposed in 1970 to solve constrained minimization problems via a sequence of quadratic programming subproblems [38]. A survey of SQP methods is given by Boggs and Tolle (1995) [18]. The main idea behind SQP is to model the optimization problem $\mathbf{x}^k$ at a given iteration k and then use this solution to construct an approximation $\mathbf{x}^{(k+1)}$. With every iteration, the solution is to should closer to the the global minimum. One main disadvantage of SQP is that these methods do not take into account that structural optimization problems have specific characteristics. One of these characteristics is that stresses and displacements are functions of $1/A_i$ (with A_i being the cross-sectional area). In the CONLIN (Convex Linearization) method, developed by Fleury in 1986 [39], this was solved as some functions are linearized in the variables A_i and some are linearized in the variables $1/A_i$. However, the CONLIN method can converge slow and sometimes not at all.

The Method of Moving Asymptotes (MMA) was introduced by Svanberg in 1987 [81] for structural optimization of nonlinear programming problems. MMA uses intervening variables, in order to control the "conservatism", which solves the problem of slow convergence in the CONLIN method. Also, the interior-point method causes that the worst-case solution is better than for eg the simplex method (see Nocedal and Wright 2006 [68]). Very important is that the MMA algorithm is good with handling problems with a large number of design variables with less constraint functions.

In the MMA, at each iteration, a convex subproblem is generated to approximate the true nonlinear optimization problem. This is according to the principle that the optimization problem is convex, if the objective function and the negative of the inequality constraints are convex and the equality constraints are met. If the optimal solution satisfies the KKT conditions from Equation 3.3, the global minimum can be found [68]. The optimization problem is bounded by assymptotes which can be updated every iteration. See Figure 3.10 and 3.11 for a visual representation.

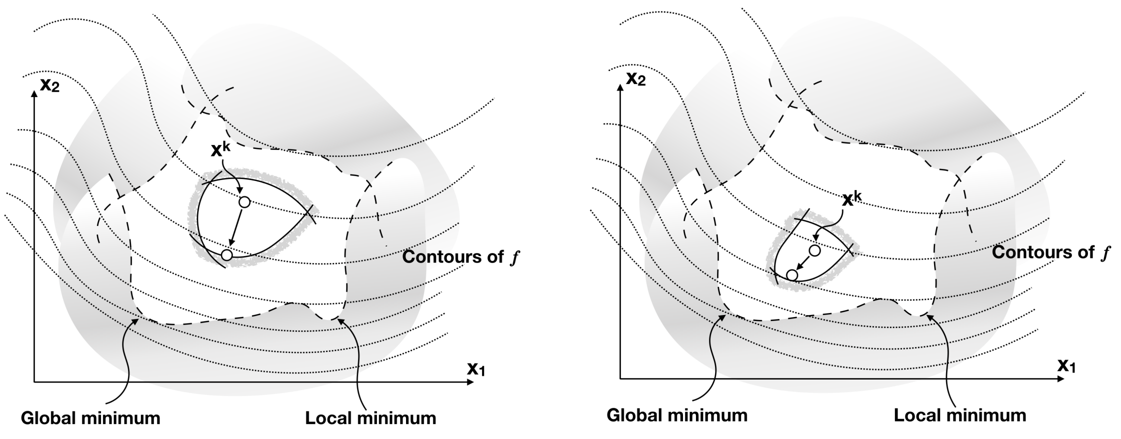
The MMA subproblem is an enrichment of the optimization problem (in Equation 3.1, with 2 extra variables, see Equation 3.20. α_j and β_j are the lower and upper bounds of the design variable. The "elastic" variable $y_i \geq 0$ is introduced in order to make sure that there is always a feasible solution. The extra variable z is introduced in order to solve non-smooth problems such as min-max problems. At every iteration, the design variable is bounded by a move limit, in order to prevent overshooting, and therefore making the optimization more stable. The used move limit in this project is 0.2, which was observed to give a stable optimization.

$$\begin{aligned}
\underset{x\in\mathbb{R}^n, y\in\mathbb{R}^m, z\in\mathbb{R}}{\text{minimize}}: \quad & f_i(x) + a_0 z + \sum_{i=1}^{m} \left(y_i c_i\right) \\
\text{subject to}: \quad & g_i(x) - a_i z - y_i \leq 0, \qquad \text{for} \qquad i = 1..m \\
& y_i \geq 0, \qquad \text{for} \qquad i = 1..m \\
& z \geq 0 \\
& \alpha_j \leq x_j \leq \beta_j, \qquad \text{for} \qquad j = 1..n
\end{aligned} \tag{3.20}$$

The used MMA Class is replicated from the MMA Class presented in Aage et al. (2015) [2]. This had been created according to the description given in Aage and Lazarov (2013) [1]. The current design is updated with as inputs the current design, the objective, constraint, and sensitivities, and the move limit.

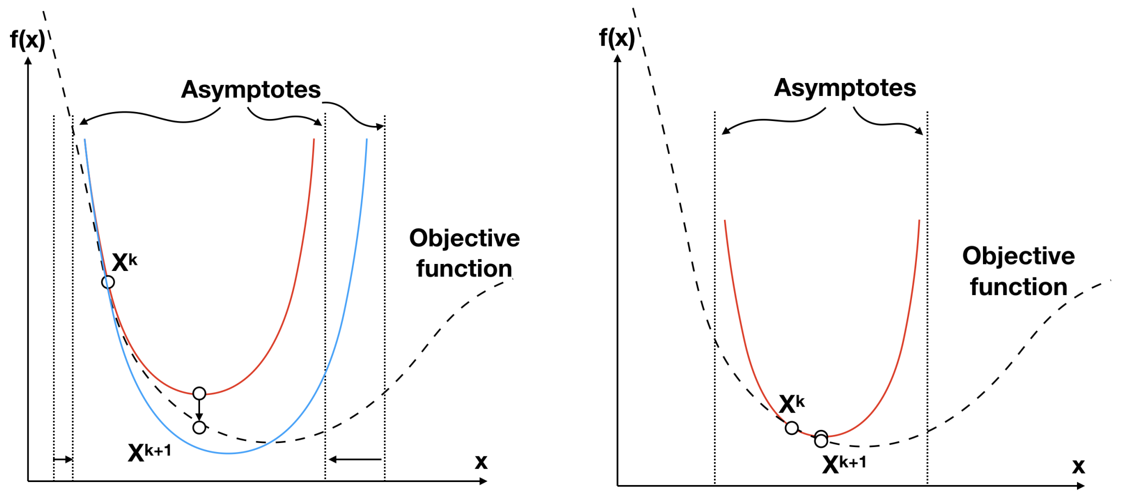
Because the MMA code performs best for numbers within a certain range, a scaled objective is used as input for this code. The initial scaled objective is set to be 10.0.

The MMA is an *interior-point method*, which requires that at all iterations, the inequality constraints in the problem are satisfied. This high priority to obey the constraint functions, can cause that material is drastically removed. To slow down this process, the volume constraint should manually be adjusted, like a volume constraint limit. This is done by setting the allowed volume 5% lower than the amount of material present in the structure. This is done every 10 iteration. The design can often obey this volume constraint within 1 iteration. Next, there are 9 iterations left to improve the design for this volume constraint. Also, the MMA function uses information from the previous 3 iterations, which would cause worse results if the volume constraint is updated every iteration.



(a) Iteration 1. (b) Iteration 2.

Figure 3.10: The MMA follows the Sequential Convex Programming approach where a sequence of approximations, controlled with move limit strategy, results in finding the minima of the global defined problem. This figure shows two design variables and three constraint functions.



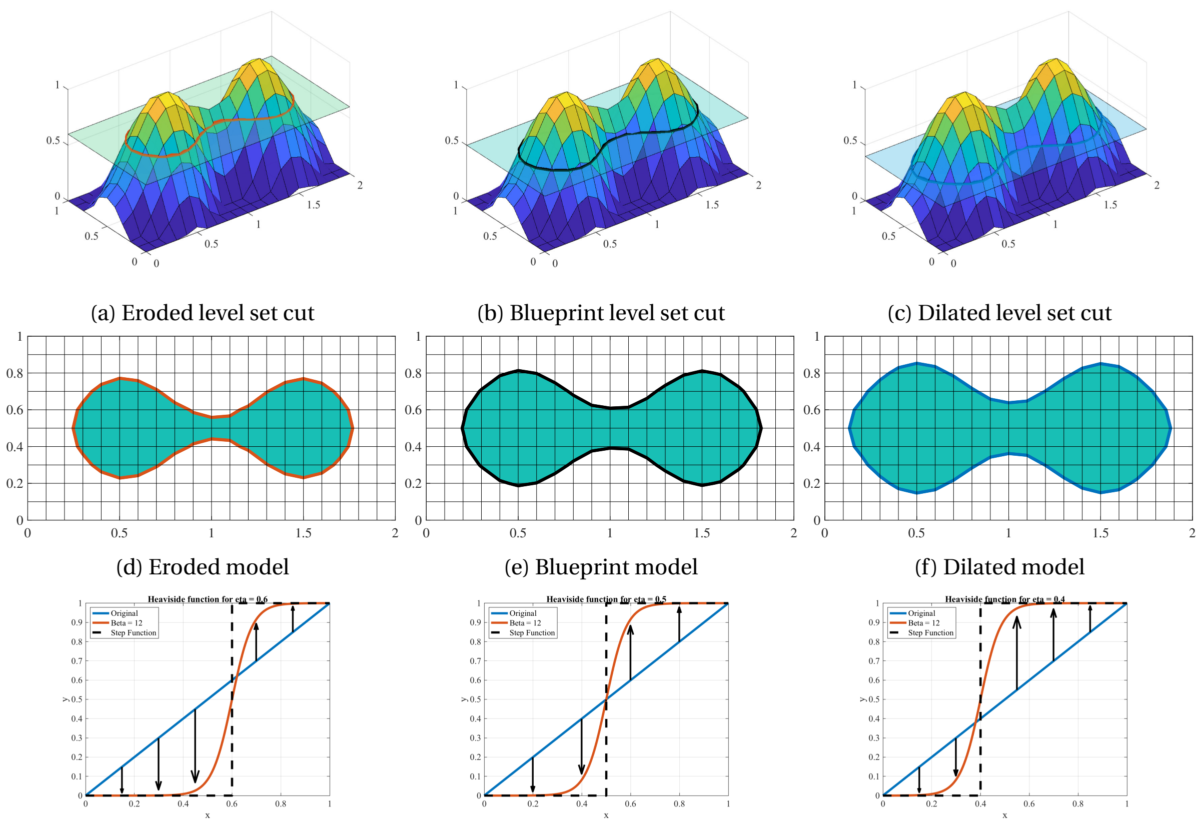
(a) Iteration 1. The asymptotes of the convex solution (in blue) are moved to new asymptotes (in red)

(b) Iteration 2. This time there is no need to adjust the asymptotes.

Figure 3.11: The working of the MMA function. The shape of the objective function is not known. The asymptotes are adjusted every iteration.

3.5. Robust design

Next, an optional feature is added which can improve the physical design. Figure 3.12 shows how a different level set cut value can lead to different model design. An *eroded*, *blueprint*, and *dilated* design can be created. This principle is utilized in the code. The threshold value η in Equation 3.15 is adjusted which leads to a different mapping. This adjustment is called the threshold offset $\Delta\eta$.



(a) Eroded level set cut (b) Blueprint level set cut (c) Dilated level set cut

(d) Eroded model (e) Blueprint model (f) Dilated model

(g) Heaviside function for Eta = 0.6 (h) Heaviside function for Eta = 0.5 (i) Heaviside function for Eta = 0.4

Figure 3.12: A different cut value can lead to a different level set design

Each of the *eroded*, *blueprint*, and *dilated* designs perform different. The *eroded* design has the lowest weight, but the worst objective. The *dilated* design has the lowest objective, but the highest weight. The *blueprint* design is in between. This is shown in Figure 3.13.

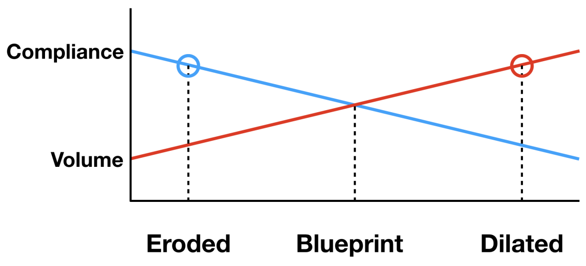


Figure 3.13: The performance of the three different designs. To have a margin, the worst design is taken. This is not necessarily linearly, as projected here.

This difference in performance causes that each of the *eroded*, *blueprint*, and *dilated* designs serve a different purpuse. The *eroded* design is used in the FEM analysis and the sensitivity analysis. The *dilated* design is used in the volume constrain analysis. The *blueprint* design is used as the final result. This is shown in Table 3.1.

Table 3.1: Deflection of testcase 1

Design	Utilization
Eroded	Objective
Blueprint	Final design
Dilated	Constraint

Utilizing these three different designs causes the final outcome to be safe within the volume constraint, and safe within the objective performance. For example small and shart edges which are present in the *eroded* model are more thick and robust in the *blueprint* model.

Finally, the *blueprint* volume and *dilated* volume can differ substantially after many iterations. Having a big threshold offset $\Delta\eta$, would mean that the final design has much less material available. The volume continuation scheme presented in Wang et al. (2011) [86] could be used to update the target volume fraction, see Equation 3.21. The dilated volume V_d and the blueprint volume V_b are calculated and the used volume fraction constraint V^* is adjusted. The result is that the volume fraction of the *blueprint* design matches the set volume fraction for the design $V^{*,final}$. This is done from the moment that volume constraint is satisfied for the first time, and is updated every 20 iterations.

$$V^* = \frac{V_d}{V_b} V^{*,final} \tag{3.21}$$

4 Results of tests

In this chapter, the CutFEM code will be tested. First, the sole FEM code will be tested and the results will be compared to existing commercial software. As the commercial software is considered to be reliable, this will work as both validation and verification. Next, the model for performing topology optimization using CutFEM will be tested. Two benchmark problems will be used for this. The results will be compared with a classical topology optimization method using SIMP. Finally, in robust design implementation is tested, which is presented in Section 4.3.

Furthermore, as long as the website is online, more results are visible at `http://www.student.dtu.dk/~s173509/`

4.1. Finite Element Analysis

4.1.1. Setup

In order to test if the FEM code works, some simple tests were done where the results were compared with an existing commercial software. The test cases should be be simple, so that one specific functionality is tested. Next, the CutFEM elements should be big enough to have an influence on the structural behaviour. Furthermore, the test cases should be performed in every direction, on a varying amount of processors, and with varying cuts in the CutFEM elements. Finally, a mesh convergence test should be done, although this should not influence the comparison with the commercial software.

Comsol Software:
The commercial software chosen to compare the PETSc model results with is COMSOL Multiphysics. This is a cross-platform finite element analysis, solver and multiphysics simulation software. The validation the performance of Comsol can be seen in *Large Deformation Analysis of a Beam* document in [31] where the *Straight Cantilever GNL Benchmark* model from Becker (2000) [12] is studied.

Test input:
A beam will be tested, which is fully clamped on one side, and has a face load on the other side. The boundary conditions and load are actually applied at nodes instead of faces. Also CutFEM elements can be clamped as usual, even though there is no node to fix at the boundary between phases. The elements are linearly connected so can not bend through. To simulate a face load on a discretized nodal field, the load on the edges are half of the size of the loads on the nodes in the middle. The sum of the total load is calculated in the PETSc model and used as input in COMSOL. The dimensions for the beam are always $3 \times 1 \times 1$, with, unless mentioned, element size $0.1 \times 0.1 \times 0.1$. For consistence, the face load is applied at an area of 0.6×0.6.

In COMSOL, the test setup and mesh are mimiced as well as possible. The *Discretization* for the displacement field is set to be *Linear*. The *Geometric Nonlinearity* is set to *Force Linear Strains*. Geometric Nonlinearity is the non linear relation between stress and strain, which can occur due to large deformations. As the used PETSc model can only do this linearly, this is forces to be linear in COMSOL. Finally, it should be kept in mind that in both models linear elements are used, so the curvature might not be accurately modelled, also known as 'locking'.

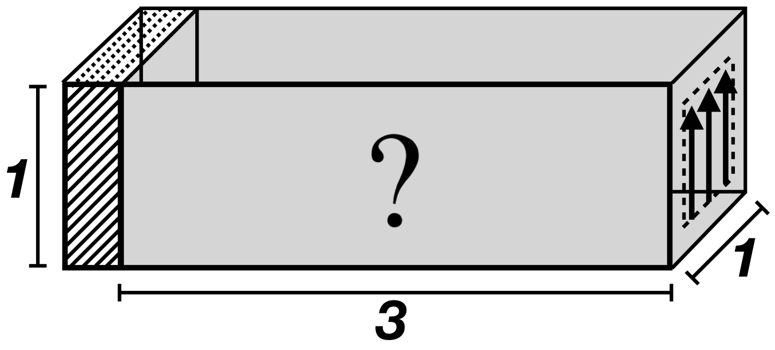
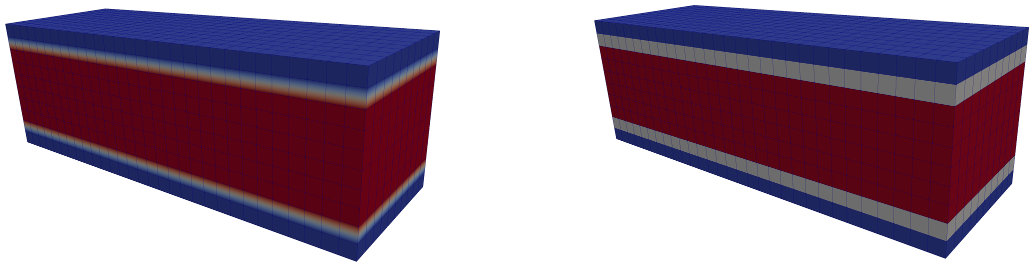


Figure 4.1: Design domain an boundary conditions for the Cantilever beam problem. Nr of elements = 192x64x64; v = 0.33 ; E = 100 ; Face loading with load per node = 0.04625.

The results from the test that will be compared are the displacement. The displacement distribution is not compared, only the maximum total displacement and the displacement in y direction at location $x = xmax$. Note that in the figures, only the solid part is shown, and a scaling factor is used for the displacement.
The general Petsc model can be seen in Figure 4.2. In Figure 4.2a it can be seen that a phase value is assigned to all nodes so that a solid beam is in between two fluid parts. In Figure 4.2b it can be seen that the elements are marked a value which represents fully solid, fully fluid, or CutFEM.



(a) Phase value at nodes. Red nodes are solid, blue nodes are fluid

(b) Marked elements to represent element type. Red elements are solid, grey elements are Cut Elements, blue elements are fluid

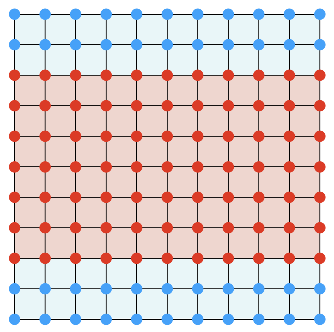
Figure 4.2: Petsc model input

4.1.2. Results

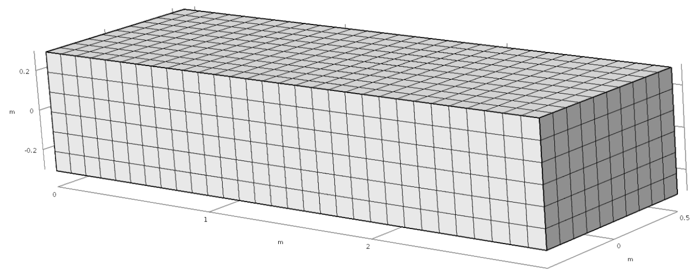
In this section, the tests and the results will be described. 2 more tests were performed, which gave similair results and are presented in Appendix E.

Beam without CutFEM:

In this test, the working of the CutFEM elements will be nullified. This shows the performance of the PETSc code without CutFEM elements. This is done to check if this most simple model. Also it gives and indication of how much the PETSc model differs from COMSOL for the results with CutFEM elements. The used models can be seen in Figure 4.3. The results can be seen in Figure 4.4 and Table 4.1.



(a) Front view of the input for sPhys. Red is solid, blue is fluid, Cut Element are set to perform as fluid elements.



(b) Design input in Comsol

Figure 4.3: Design input

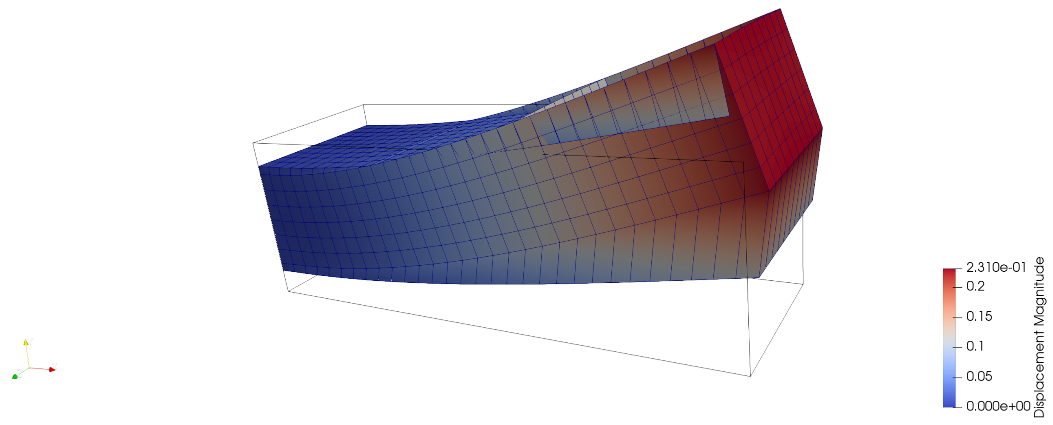


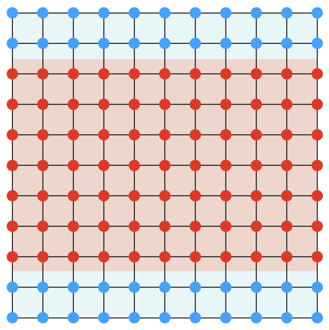
Figure 4.4: Deflected beam without CutFEM

Table 4.1: Deflection of beam without CutFEM

Direction	PETSc[m]	Comsol[m]	Difference [%]
Displacement Y	0.2285	0.23251	1.7
Displacement tot	0.2310	0.23502	1.7

As can be seen, there is a difference in the displacement of approximately 1.7%. The weird shape of the beam in Figure 4.4 can be explained by the fact that the solid part did deflect, while the fluid part stayed in its original position.

Beam with CutFEM:
In this test, the beam is will have two rows of CutFEM elements. This shows the performance of the stiffness matrix of the CutFEM elements. The many elements have a significant influence on the stiffness matrix and behaviour of the beam. The test is similar to the previous test, but now the CutFEM elements will perform as designed. The used models can be seen in Figure 4.5. The results can be seen in Figure 4.6 and Table 4.2.


(a) Front view of the input for sPhys. Red is solid, blue is fluid.

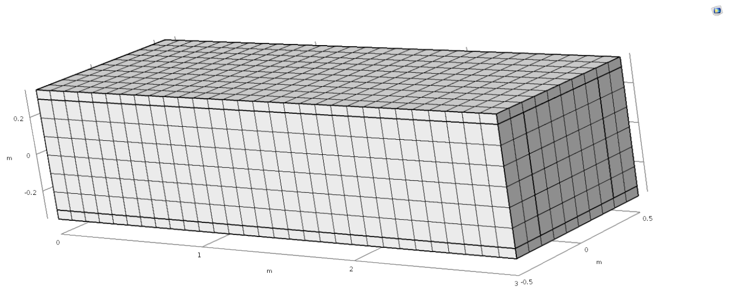

(b) Design input in Comsol

Figure 4.5: Design input

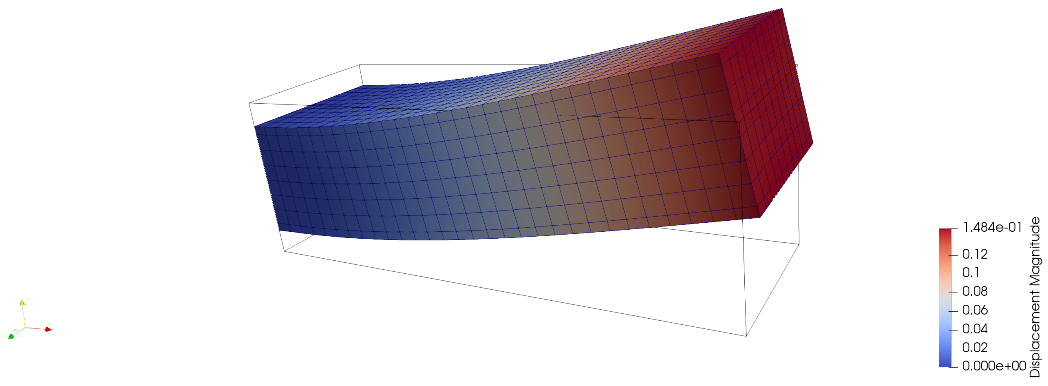


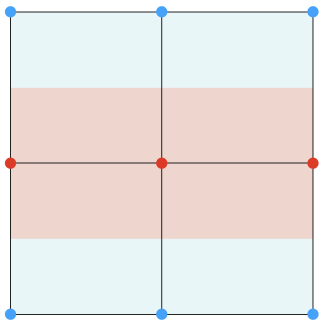
Figure 4.6: Deflected beam with CutFEM

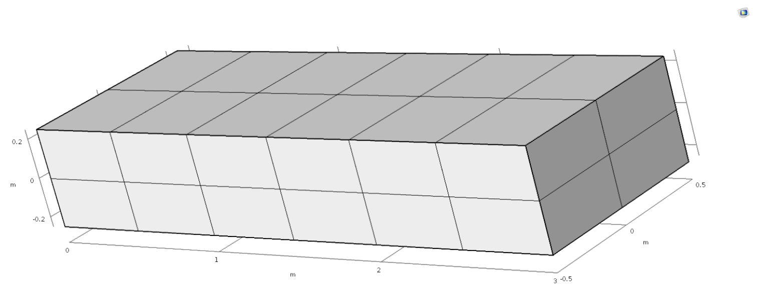
Table 4.2: Deflection of beam with CutFEM

Direction	PETSc[m]	Comsol[m]	Difference [%]
Displacement Y	0.1463	0.14575	0.377
Displacement tot	0.1505	0.14727	2.19

As can be seen, there is a difference in the displacement similair to the beam without CutFEM.

Beam with only Cut Elements:
In this test, the beam be made up of only CutFEM elements. This is the most extreme example of the performance of the stiffness matrix of the CutFEM elements. The mesh size is decreased as much as allowed for by the code. This shows the performance of the CutFEM elements without any influence of fully solid elements. Note that the elements are too big to obtain reliable results, but that does not matter since the main objective is to compare it to commercial software. The used models can be seen in Figure 4.7. The results can be seen in Figure 4.8 and Table 4.3.


(a) Front view of the input for sPhys



(b) Design input in Comsol

Figure 4.7: Design input

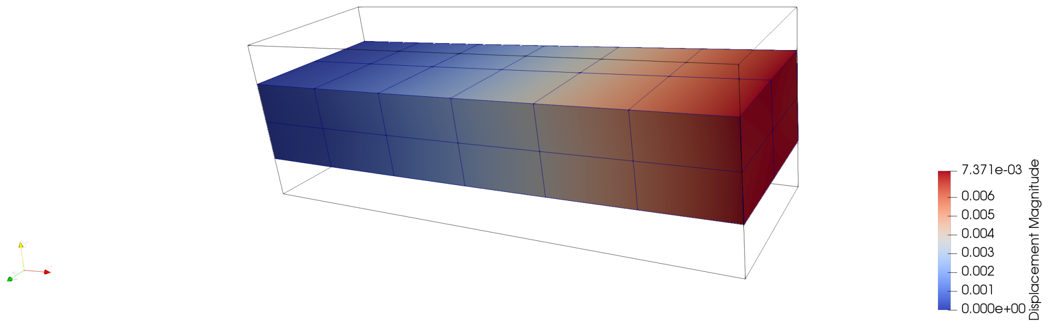


Figure 4.8: Deflected beam with only Cut Elements

Table 4.3: Deflection of beam with only Cut Elements

Direction	PETSc[m]	Comsol[m]	Difference [%]
Displacement Y	0.007330	0.0072730	0.78
Displacement tot	0.007371	0.0073288	0.58

As can be seen, the solid part of the CutFEM elements can work just like a normal fully solid element.

4.2. Optimization

In this section, the tests and results for the topology optimization problems are presented. Two problems were tested. Test 1 will create a cantilever beam, test 2 will create a MBB (*Messerschmitt-Bölkow-Blohm*) beam. Both problems were run by topology optimization using CutFEM from this project and by classical topology optimization using the SIMP method. The PETSc code that was used for the SIMP method is presented in Aage et al (2015) [2] In this section, the robust method is not used.

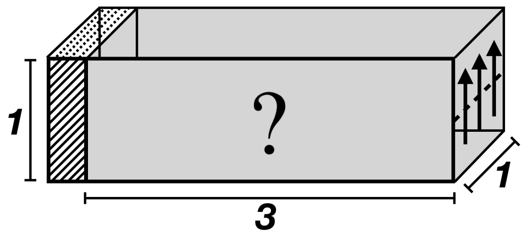
The results from the test that will be checked is the behaviour of the design, and the behaviour of the code variables. The behaviour of the code variables is for example checking the objective variable which should be minimized over time, or the volume constraint which should aim to approach and stay under 0.

4.2.1. Setup

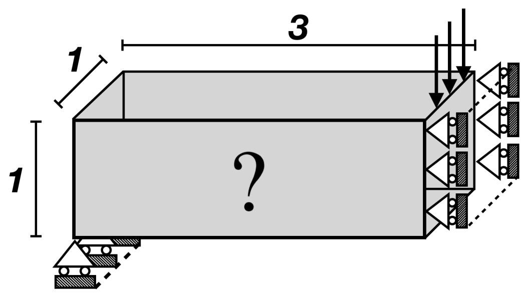
Test loadings:

Test 1 is a cantilever beam. The design domain can be seen in Figure 4.9a. The beam is fully clamped on one side, and has a line load on the other side.

In this test, a MBB-beam is tested. This problem is used to optimize the design of one half of a symmetric bridge. The design problem is shown in Figure 4.9b. The bridge is fully clamped clamped in x direction on one side, has a facet loading on the other side, and this facet has a boundary limitation to move out of plane.



(a) Cantilever beam problem.
Line loading with load per node = 10.0.



(b) MBB-beam problem.
Face loading with load with load per node = −0.1

Figure 4.9: Design domain an boundary conditions. Nr of elements = $192x64x64$; $\nu = 0.33$; $E_{min} = 1^{-7}$; $E_{max} = 100$; Allowed volume = 0.14 ; Convergence for change < 0.01

Test input:

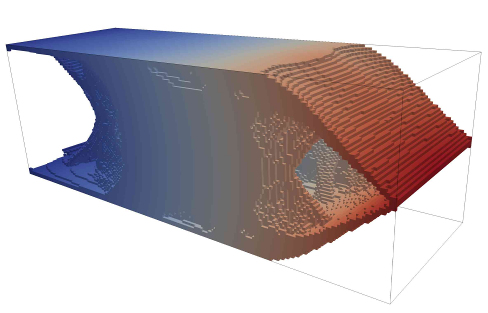
The 4 different inputs are described here:

- The *SIMP rmin = 0.016* test uses the SIMP method with a density filter with a stencil size of 1 element. Using a filter with stencil size of 0 elements, resulted in worse objective. No Heaviside projection is used, as this converged too fast.
- The *SIMP rmin = 0.08* test uses the SIMP method with a density filter with a stencil size of 5 elements. Also Heaviside mapping with increasing β is used, to a final value of β is 500.
- The *CutFEM* test uses the CutFEM method. A filter is used with the Heaviside projection. The filter radius is $r_{min} = 0.08$, so a stencil size of 6 elements. The sensitivity perturbance $\epsilon = 10^{-5}$, which gave good results in the FD check. The used TetGen settings are *"pp/0.0000001YT0.00000000000000000001Q"* which is described in Section B.3 in Appendix B. This test uses a personalized initial design. This initial design consists of a solid beam of material in most of the design area. In order for the sensitivites to be calculated in the Cut Elements, there is fluid on the outside and fluid holes are created within the beam. Finally, at the areas close to the loading and boundary conditions there is solid material to carry the loads.
- The *CutFEM cont* test uses the CutFEM method, with exactly the same input parameters are the *CutFEM* test, but continues with the final design from the SIMP method as its initial design. The SIMP

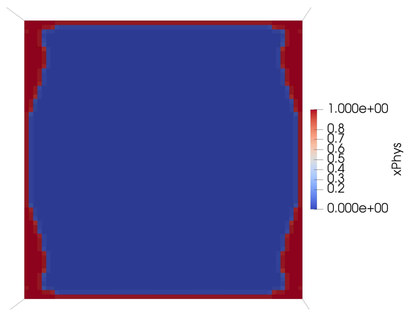
method was run with a PDE filter, and filter radius is $r_{min} = 0.08$. The PDE filter uses phase values at the nodes, so the final design for the SIMP here could be used as the initial design for the CutFEM.

4.2.2. Results

In this section, the results for the cantilever beam problem are presented. The results for the MBB problem are presented in Section F.1, as these results are similar to the cantilever beam results. First, the results for the design with the *SIMP rmin = 0.016* and the *SIMP rmin = 0.08* tests can be seen in Figure 4.10 and 4.11. Figures 4.10b and 4.11b show a slice of the phase at 0.5 of the beam. Next, the initial and resulting designs for the *CutFEM* test and the *CutFEM cont* test can be seen in Figure 4.12 and 4.13. For all the steps in between, see Section F.2.

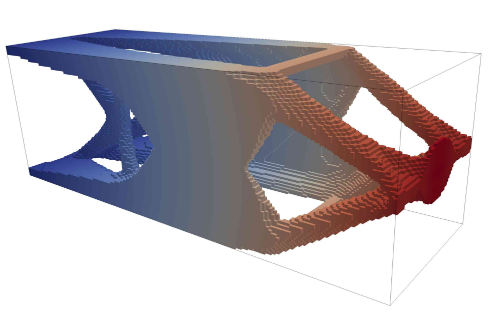


(a) Optimized structure. The color represents deflection.

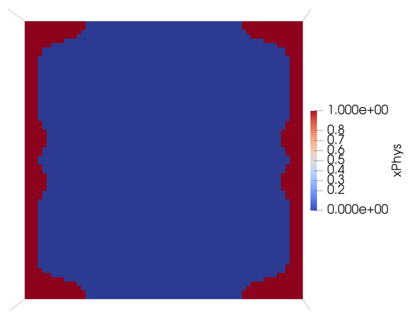


(b) Cutout at 0.5 of the beam.

Figure 4.10: Optimized structure for the *SIMP rmin = 0.016* test of the cantilever beam optimization.

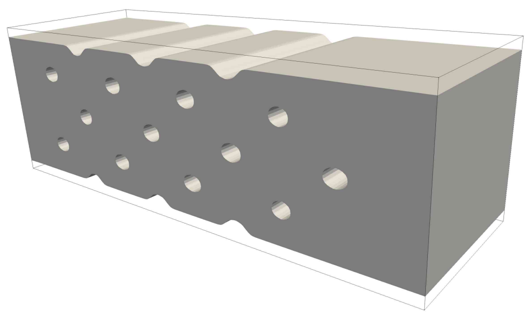


(a) Optimized structure. The color represents deflection.

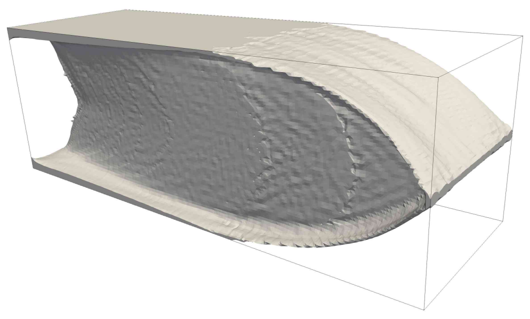


(b) Cutout at 0.5 of the beam.

Figure 4.11: Optimized structure for the *SIMP rmin = 0.08* test of the cantilever beam optimization, with Heaviside projection with final $\beta = 500$.

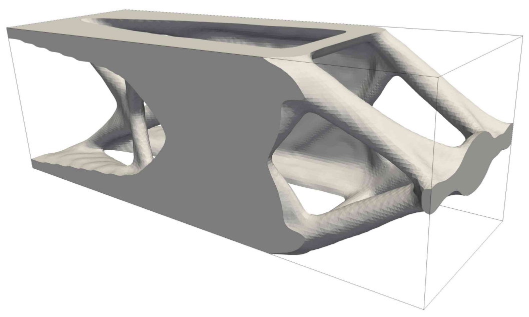


(a) Initial design.

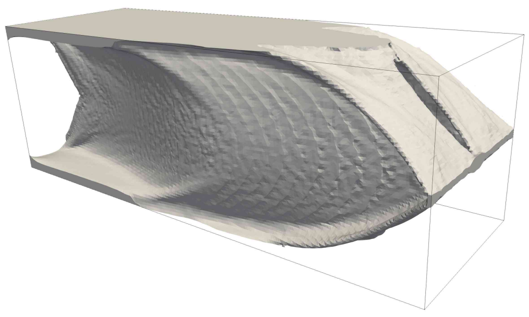


(b) Optimized structure, iteration 1000.

Figure 4.12: Structures for *CutFEM* test of the cantilever beam optimization



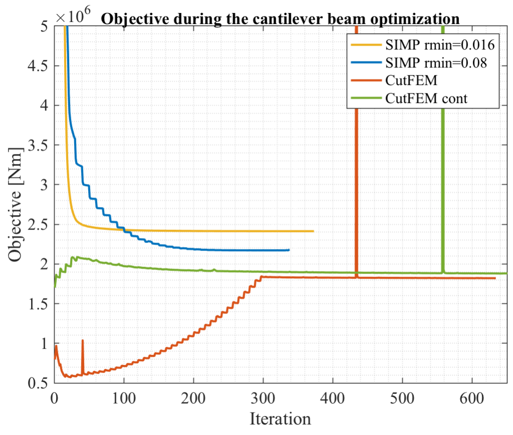
(a) Initial design.



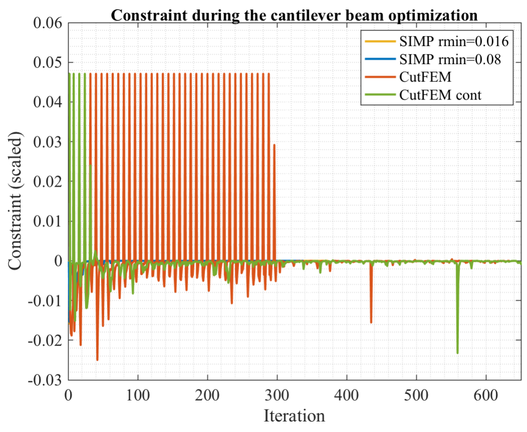
(b) Optimized structure, iteration 1000.

Figure 4.13: Structures for *CutFEM cont* test of the cantilever beam optimization

Also, the development of the different parameters have been checked throughout the optimization. The development of the objective and the constraint at every iteration can be seen in Figure 4.14. The development of the calculation time, the number of iterations to solve, and the number of cut elements in the structure can be seen in Figure 4.15

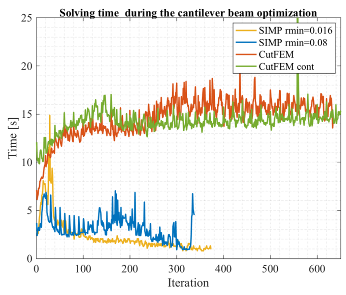


(a) Objective

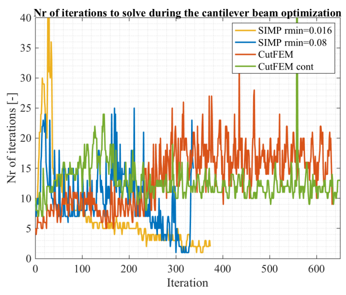


(b) Constraint

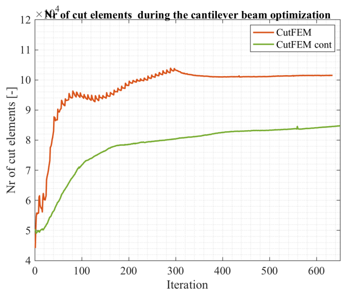
Figure 4.14: Function variables at every iteration.



(a) Time per iteration



(b) Number of iterations to solve



(c) Number of cut elements in structure

Figure 4.15: Function variables at every iteration

4.3. Robust design

In this section, the tests and results for the robust topology optimization problems are presented. The same two problems as in the previous section were tested, again for both the CutFEM and the SIMP method. This time the robust design approach is used.

4.3.1. Setup

Exactly the same test setup is used as in the previous section. This time a projection threshold value is used, as explained in Section 3.5. Different thresholds offset values $\Delta\eta$ were tested, $\Delta\eta = 0.00$ (no threshold offset), $\Delta\eta = 0.01$, $\Delta\eta = 0.02$, $\Delta\eta = 0.03$, and $\Delta\eta = 0.05$.

4.3.2. Results

In this section, the results for the cantilever beam problem are presented. The results for the MBB problem are presented in Appendix G, as these results are similar to the cantilever beam results.

Firstly, Figure 4.16 shows the eroded and the dilated design on top of each other. This shows that the dilated design is indeed bigger than the eroded design.

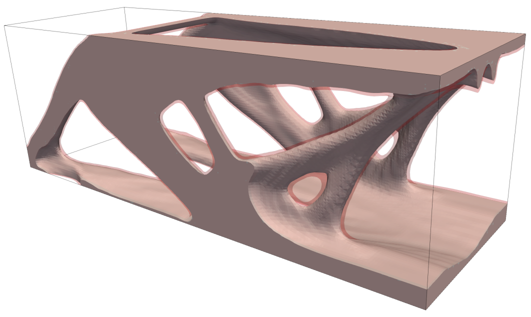
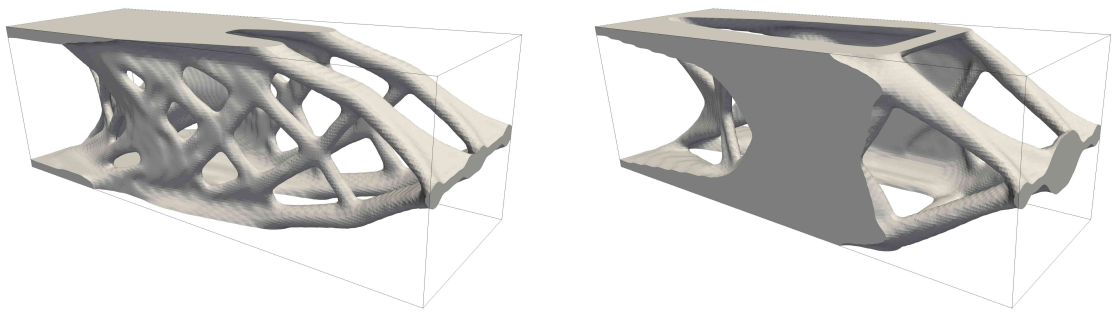


Figure 4.16: The difference between the eroded design and the dilated structure. The eroded design is white, the dilated design is red transparant.

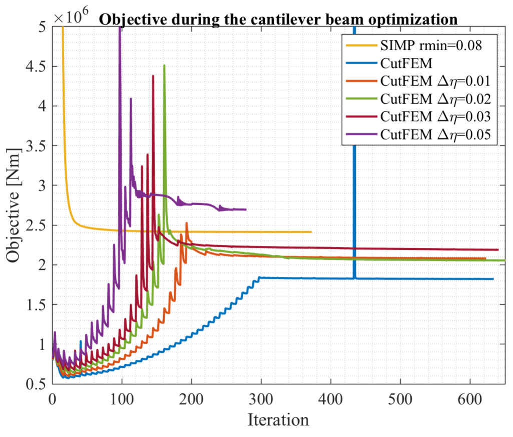
The optimized robust structure for a threshold ofset value of $\Delta\eta$ = 0.03 are shown in Figure 4.17. The optimized robust structures for all other threshold offsets are shown in Appendix G.2.



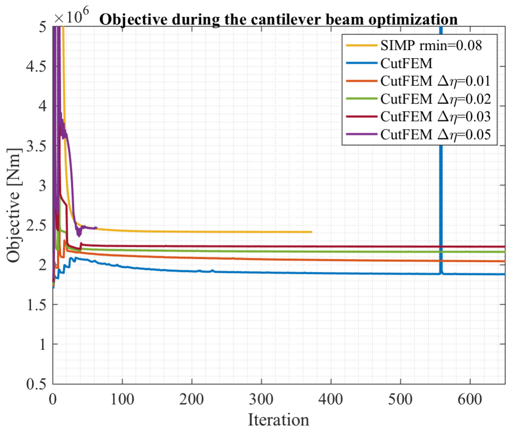
(a) *CutFEM* test (b) *CutFEM cont* test

Figure 4.17: Optimized robust structure for the *CutFEM* test and the *CutFEM cont* test of the cantilever beam problem. A threshold offset value of $\Delta\eta$ = 0.03, at iteration 1000.

Now, results for different threshold values can be compared. The objective values, can be seen in Figure 4.18. The objective values for the MBB problem are presented in Section G.1.



(a) *CutFEM*



(b) *CutFEM cont*

Figure 4.18: Objective functions for different threshold offset values for the MBB beam tests

5
Discussion on the performance of topology optimization using CutFEM

In Chapter 4, the performance of topology optimziation using CutFEM was tested. In this chapter, this performance these results are discussed. This is done in order to identify when the usage of CutFEM is more beneficial than classical FEM in topology optimization applications.

5.1. Performance of Cut Elements in FEM

Firstly, the results presented in Section 4.1 show that the outcome of a model described with Cut Elements is similar compared to a model described with classical elements. A small difference in result was already present in the test without Cut Elements, see Table 4.1, and a similar difference is present in the tests with Cut Element, see Table 4.2 and 4.3. Such a small difference in results can be caused by different things, for example the solver or hidden corrections by Comsol. The finite element method is already by itself a method to approximate real life results.

Next to the results presented above, other configurations have also been checked. For example, the load has been applied in different directions, and the phase values have been set so the the cut is no exactly in the middle of the CutFEM element. These tests showed the same performance of the Cut Elements. Furthermore, a check was done on the stiffness matrices. The stiffness matrix for a fully solid element was computed in the classical way (so with 8 Gauss points). Also the stiffness matrix for a fully solid element, was computed with the code used for Cut Elements (resulting in 24 Gauss points). These stiffness matrix were exactly the same.

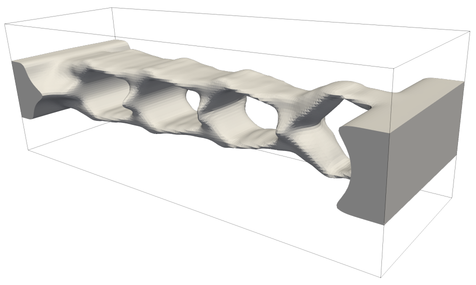
From these test it can be concluded that Cut Elements can be used to accurately perform a finite element analysis.

5.2. Performance of topology optimization using CutFEM

From the CutFEM test which are described in Section 4.2, a few things can be noticed. First, the performance of the CutFEM method will be discussed, focusing on what happens during the optimization process. Next, the performance of the 4 test inputs will be compared, focusing on the measurable results.

5.2.1. The optimization process

During the iterations in the beginning of the optimization, the most important task is to lose material in order to obey the volume constraint. The function that gradually lowers the allowed volume fraction, works well. Every 10 iterations, the volume fraction is lowered, and it often takes 1 iteration to obey the new volume constraint. This can be seen in Figure 4.14, where every 10 iteration the objective jumps up and the constraint has one peak. Without the function that gradually lowers the allowed volume fraction, material is cut out drastically, for which an example is shown in Figure 5.1.


(a) The cantilever beam structure at iteration 30

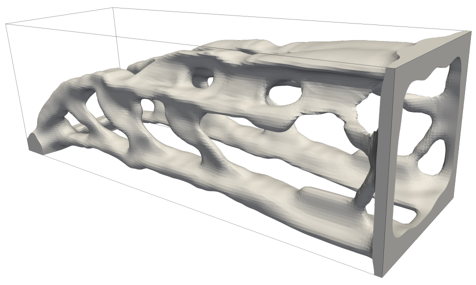

(b) The MBB beam structure at iteration 30

Figure 5.1: Structure during optimization, without gradually lowering the allowed volume

Once the final allowed volume fraction is reached, it was observed that the design is not tempted to change. It might even happen that the optimization is actually stuck in a local minimum. In the SIMP method, beams can grow towards each other, because they are always connected with a very low density. In the CutFEM method that does not happen. Sometimes structures will disconnect which can cause a jump in the objective. For more information on these jumps, the reader is referred to Jenkins and Maute (2015) [50].

The final result for the CutFEM method gives a design layout which looks similar to the design layouts of other methods. As can be seen in Figure 4.12b and 4.13b, the design looks smooth and can contain thin plates, because it does not have to follow a grid.

In the *CutFEM cont* test, the test which uses the final design from the SIMP method as initial design, the optimization is less drastic. The optimization process mostly consists of changing the beams into thin plates, because there are no restrictions. From an early stage in the process, the design looks like the final design. Also, as shown in Figure 4.14, the objective function changes less for the *CutFEM cont* test than for the *CutFEM* test.

The performance of the features which are new in this project, seem to perform well. Firstly, the CutFEM method has more opportunities now that it is used in 3D. In 2D, CutFEM performs a little bit like shape optimization, because the sensitivities are only calculated on the boundaries. In 3D, CutFEM can also create holes in the middle of a flat plate. The third direction makes that the sensitivities are calculated everywhere on the boundary of the plate, see Figure 5.2. This makes that the performance of 3D CutFEM is less dependent on the initial design compared to 2D CutFEM. Note how the optimized structures in Figure 4.12 and 4.13 look very similar even though the initial design is very different.

Secondly, the conventions for the ambiguous situations, that were introduced in Section 2.2.1 and Section 2.2.2, seem to not have caused any problems. No undesired behaviour was noticed because the fluid nodes are always connected, and the solid nodes can lose their connection. Neither does the design show directional behaviour, because an element can be more stiff in one direction.

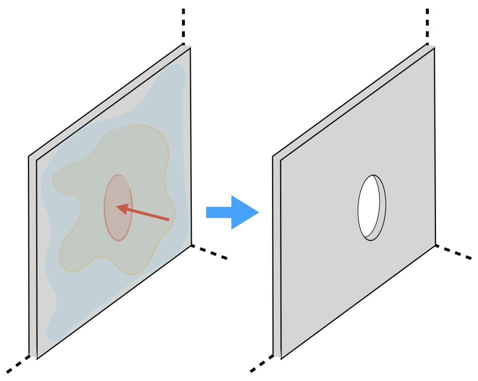


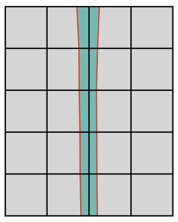
Figure 5.2: The changing of topology in a 3D plate. Sensitivities are calculated everywhere on the boundary of the plate (left), after which a hole can be formed (right). In 2D the sensitivities are not calculated in the middle of a plate.

5.2.2. Quantitative comparison

From all the tests which are described in Section 4.2, a few things can be noticed. Firstly however, it should be mentioned that the tests SIMP methods seemed to have trouble with converging to a local minimum. The test with a lower radius (*SIMP rmin = 0.016*) resulted in a worse objective function than the test with higher radius (*SIMP rmin = 0.08*). This was not expected, because a lower filter radius should more freedom in the design layout. Trying to solve this, for example by using a Heaviside filter for the low radius test, or by having no filter at all did not solve this.

From the results of the tests in Figure 4.14a it can be seen that the SIMP method uses less time and iterations than the CutFEM method to converge. In fact, it was observed that the CutFEM method could continue for several thousand iterations, so the convergence requirement is important. From Figure 4.15a and 4.15b, it can be seen that for every iteration the system of SIMP method is solved faster and less iteration are required. Furthermore, the CutFEM method also needs an initial design, which requires time to be created. Finally from Figure 4.15, it can been seen that the system solving time does still depend more on the amount of iterations to solve than on the number of Cut Elements in the system.

From the results of the tests in Figure 4.14a it can be seen that the CutFEM method can lead to a better objective function. Firstly, this might be because in the CutFEM method thin plates can be created, unrestricted to the grid see Figure 5.3a. Secondly, in the CutFEM method material can be used more effectively, see Figure 5.3b. It can be seen that it takes a lot of iterations before the *CutFEM* test reaches the objective function value around which it will approxamtely stick. The *CutFEM cont* test design does not change much from the initial design, and the objective function does also not change that much during the process.


(a) Thin plates can be created, unrestrictred to the grid

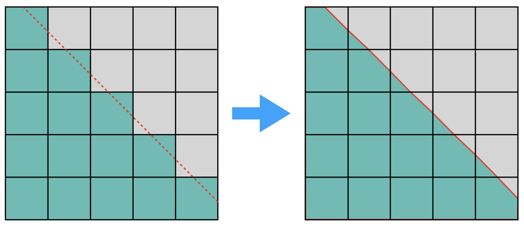

(b) The SIMP method (left) can not use material as efficient as the CutFEM method (right)

Figure 5.3: Structural design phenomena for CutFEM

5.3. Performance of the robust design approach

From the tests which are described in Section 4.3, a few things can be noticed. It can be seen that the function performs as expected, and the dilated design layout is bigger than the blueprint and the eroded design layouts. It is harder to draw conclusions here, as this functionality is related to the manufacturability of the structure, which is harder to measure.

First, the design of the structures will be compared, based on the Figures in Section G.2. It can be seen that the CutFEM method performance with a threshold offset value of $\Delta\eta = 0.01$, is different from the performance without threshold offset value. The design does not contain extremely thin plates anymore. From a threshold offset value of $\Delta\eta = 0.02$, almost all thin structural elements have disappeared. From a threshold offset value of $\Delta\eta = 0.03$, plates seem to make place for beams, as can be seen in Figure 4.17a. From a threshold value of $\Delta\eta = 0.05$, the CutFEM method perform significantly different. Without an adequate initial design, the optimization seems to end up in a local minimum.

In the *CutFEM cont* test, the test which uses the final design from the SIMP method as initial design, the differences is less drastic but still present. The optimized structure without threshold offset value already contains less thin structural elements. Still, the plates grow thicker with increasing threshold offset value. From a threshold offset value of $\Delta\eta = 0.03$, the topology is significantly changed less from topology of the initial design.

Secondly, the quantitative performance can be evaluated. As can be seen in Figure 4.18, does the objective function increase with increasing threshold offset value. Until a threshold offset value of $\Delta\eta = 0.03$, the optimization process is still similar to the optimization process without threshold offset. Notice how, the impact of the threshold offset is more present in the *CutFEM* test than in the *CutFEM cont* test. From a threshold offset value of $\Delta\eta = 0.05$, the optimization process is very different, and the objective value goes up.

6 Preliminary results on CutFEM with fluids

Before going to the final remarks, one more part of the project should be addressed. As mentioned in the introduction, CutFEM might have potential to be useful in FSI problems. A start has been made in using CutFEM in combination with fluids. This has lead to some some preliminary results. The implementation, and the outcome will be presented in this chapter.

The Brinkman penalization method can be used to simulate a flow through a domain, where the solid is modelled as a porous medium with low permeability. This volume penalization method was proposed by Arquis and Caltagirone in 1984 [10], and can be applied to the incompressible flow regime (Kevlahan and Ghidaglia 2001 [55]), with moving obstacles (Kadoch et al. 2012 [54]) and to the compressible flow regime (Liu and Vasilyev 2007 [58]). In the Brinkman method, the whole domain is modelled as permeable, but the friction term is set very low for fluid, or very high for solid.

This friction term is set in the Gauss points of each element. For Cut Elements it is possible to create the element Brinkman matrix similar to the way the stiffness matrix is calculated. The element Brinkman matrix is calculated as follows:

$$\underset{24\times 24}{\boldsymbol{\alpha}} = \int_{\Omega_e} \alpha_e \underset{24\times 3}{\mathbf{N}_u^T} \underset{3\times 24}{\mathbf{N}_u} \, dV \tag{6.1}$$

In here, $\mathbf{N}_u$ is the shape function matrix, corresponding to the location of the Gauss point. α is the element impermeability. For CutFEM, this term should not be dependent on the element, but on the phase that the Gauss point represents. This element Brinkman matrix is then added to the system matrix $\mathbf{M}$, which also contains the convection matrix, the diffusivity/viscosity matrix, and the Boussinesq coupling matrix.

This idea has been implemented. A Navier-Stokes fluid solver with Brinkman penalization was used. The variables required for CutFEM were added, for example phase variables at the nodes. Next, it was implemented that the Brinkman matrix is calculated with the CutFEM procedure. The performance was investigated with a test. In the test a fluid tunnel was created in a domain, as shown in Figure 6.1. The dimensions for the domain are always $3 \times 1 \times 1$, with element size $0.1 \times 0.1 \times 0.1$. The results of the test can be seen in Figure 6.2.

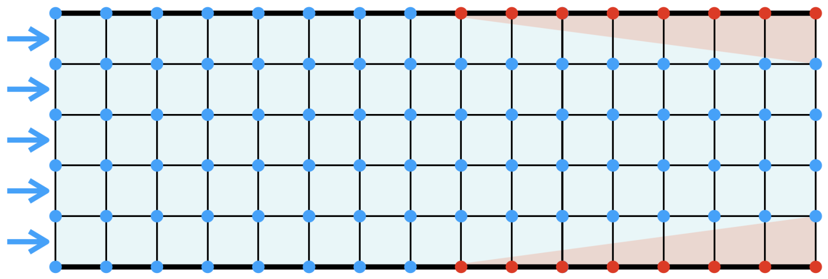
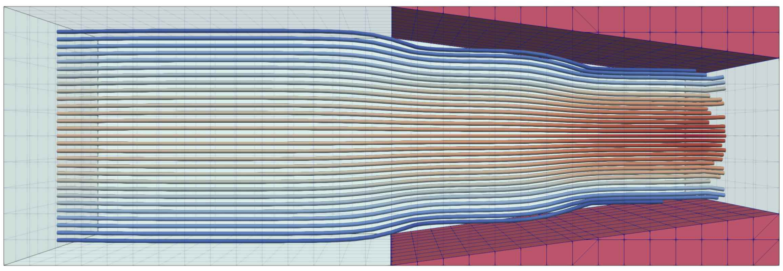
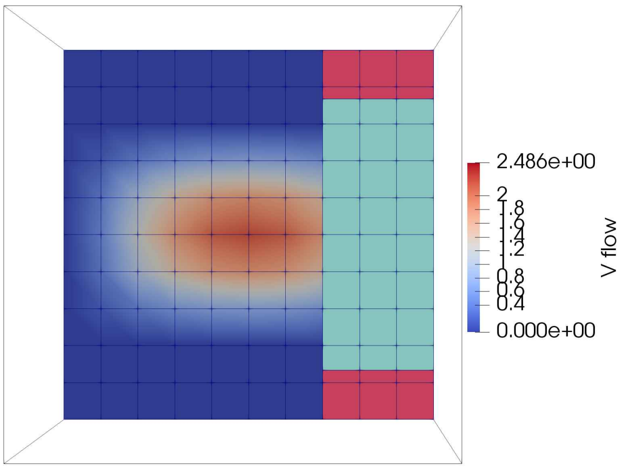


Figure 6.1: Input for sPhys. Not the real mesh size. Red is solid, blue is fluid.
The inlet is on the left, the outlet on the right. The fluid can not leave on the side.



(a) Flow streamlines in the domain



(b) A cutout of the domain. The left part is the velocity of the flow.
The right part represents the phase state (red is solid, blue is fluid).

Figure 6.2: Results of fluid test

As can be seen in the results, the flow has no velocity within the Cut Elements. This means that using CutFEM in combination with the Brinkman method does not work in this setting. This can be explained, as the Brinkman method works with a soft boundary but for a whole element, while the CutFEM method works with a hard boundary. Therefore, the next step for using CutFEM in a FSI problem, is with a method that imposes Dirichlet boundary conditions. In 1971, Nitsche introduced the idea to handle the Dirichlet boundary conditions weakly without using Lagrange multipliers [67]. Based on the results above it is believed that this implementation can lead to the best performance of CutFEM with FSI.

7 Final remarks

In this chapter, the final remarks are given. First the conclusions are given, followed by the recommendations.

7.1. Conclusions

The research objective of this project was: *"To investigate the potential for the usage of 3D CutFEM in topology optimization problems, compared to the use of finite elements (FE) with conventional geometry describing method."*.

The first goal was to develop a code that can perform a finite element analysis with CutFEM. When this code was tested, the results indicate that CutFEM is a reliable method to perform finite element analysis.

The second goal was to develop a code that can perform topology optimization with CutFEM. When this code was tested, the results indicate that CutFEM is a suitable method to perform topology optimization. Looking at the optimization process, it was seen that smooth structures were created which are not bounded to the grid. Many iteration steps are needed, as changes are only made on the boundary and because it should be prevented that material is removed too rapidly. It was found that the CutFEM method can easier change the topology in 3D than in 2D. Therefore, the initial design is of less importance. Ambiguities in 3D marching cubes do not cause problems if one convention is kept. The obtained objective function can be better than the objective function for the SIMP method. This is however at the cost of computation time, and manufacturability.

The usage of the threshold offset is a suitable method in order to improve the manufacturability. This can cause significant differences in the performance and design. The resulting designs may be easier to manufacture, but at the cost of the objective function. If the threshold offset is increased, the topology of the initial design is increasingly important to the outcome. Doing this inaccurate could potentially lead to poorly optimized designs.

In conclusion, it can be said that there is a high potential in topology optimization using CutFEM. But, because of the increase in computational time, the CutFEM method can not replace existing methods, such as the SIMP method. It is thought that a good practical usage for CutFEM could be to perform optimization with the initial design computed by the SIMP method. Next, the extra benefits that the Level Set approach in CutFEM has to offer can be used to utilize CutFEM in for example Fluid Structure Interaction problems.

7.2. Recommendations

Based on the project and the whole process there are a few recommendations.

Firstly, a number of things have not been thoroughly researched in this project. For example the consequences of the used conventions for the ambiguity in the cubes can be checked. In order to prevent directional problems, it is recommended to make the cutting edge also dependent on the cut location.

Next, the comparison with the SIMP method is only done on a high level. Only two problems were tested, with the same settings and the same initial design. It is recommended to study the influence of variables which might influence the results, for example the allowed volume fraction. Also, it might be able to quantify the manufacturability of the structural designs. Now the exact influence of the threshold offset value can be measured. With all of this, a complete comparison can be made, where the advantages of each method are measured.

Finally, and most importantly, it is recommended to perform topology optimization using CutFEM in combination with a fluid load. The hard boundary in CutFEM is one of the main advantages of this method. Therefore, it is recommended to implement CutFEM in a fluid solver with Nitsches method for the Dirichlet boundary conditions. A hard boundary is present in the Cut Element, and a FSI problem can be solved. The next step is to perform topology optimization using CutFEM on a FSI problem. This could possibly lead to great improvements of the design of structures under a fluid load.

A
Software

A.1. PETSc

In this section, the method for performing the topology optimization on multiple processors is explained. First, the background information on all the methods used is given. Next, all the settings used in the model are explained.

A.1.1. Background

In order to build the model that can perform CutFEM, a software has to be chosen. This starts with the selection for the hardware requirements.

Parallel Computing:
Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously (see Almasi and Gottlieb 1989 [6]). A large problem is divided into smaller problems, which can then be solved at the same time. The rise in usage and availability was caused by multiple factors. Firstly, the effectiveness of heat dissipation impose physical limits on the speed of a single computer. Secondly, the cost of advanced single-processor computers have increased more rapidly than their power. Thirdly, the price over performance ratio becomes more favorable if the required computational resources can be found instead of purchased, which causes existing workstations to be used. For more information, the reader is refered to Gropp et al. (2000) [46].

There are multiple models within parallel computing: data parallelism, shared memory, message passing, remote memory operations, threads, and different hybrid models. In the *message-passing* model, the processes have only a local memory but they are able to communicate with other processes by sending and receiving messages (see Figure A.1). There are some advantages of this model. Firstly, the model works well with separate processors connected by a (fast or slow) communication network. Secondly, because the memory references are very explicitly controlled, it is easier to locate and debug erroneous memory reads and writes. Thirdly, a way to explicitly associate specific data with processes is provided, and therefore the compiler and cache-management hardware can function to their maximum performance. That is why message passing is used in this project.

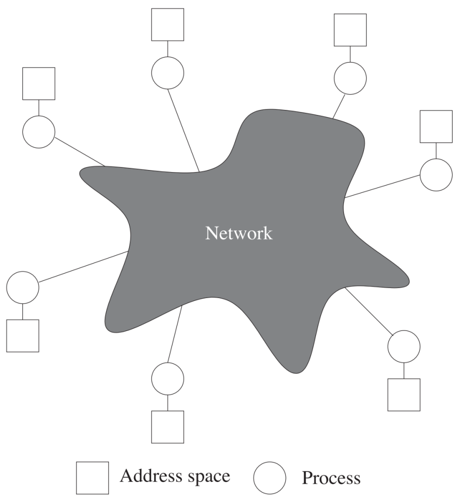


Figure A.1: The Message Passing model, from [46]

Message Passing Interface:
Message Passing Interface (MPI) is a standardized and portable message-passing standard to function on a wide variety of parallel computing architectures. It started with discussions by researchers in 1991, and resulted in 1994 in version 1.0 of MPI, and also in the founding of the MPI Forum. MPI is a communication protocol for programming parallel computers, together with syntax and semantic specifications for how its features must behave in any implementation. Or in the words of Gropp (2000) [46], it is an attempt to collect the best features of many message-passing systems that have been developed over the years, improve them where appropriate, and standardize them. This has resulted in high performance, scalability, and portability. Note that MPI is not sanctioned by any major standards body.

In message-passing, all processes (ranks) have their local data. In structural analysis it is also important to have information about neightbouring points. *Ghost points* are the elements of the array that are used to hold data from other processes. This can be seen in Figure A.2.

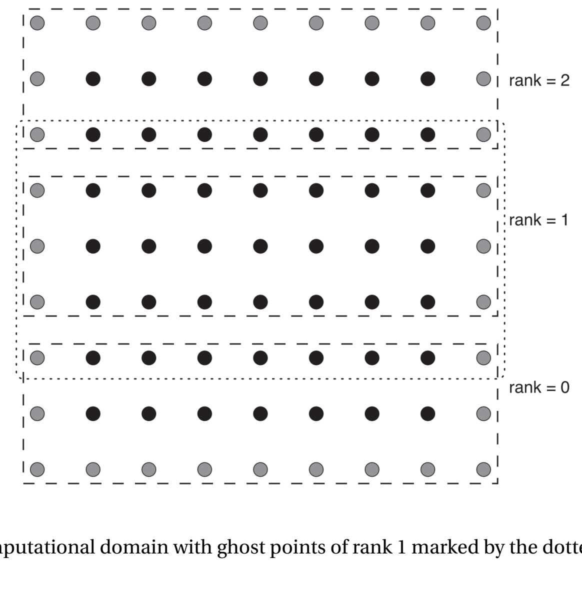


Figure A.2: Computational domain with ghost points of rank 1 marked by the dotted line, from [46]

MPICH:
MPICH is a free and open source software, portable implementation of all MPI version (see Bridges et al. 1995 [23]). It was originally developed by the Argonne National Laboratory and Mississippi State University

during the MPI standards process starting in 1992, in order to provide feedback to the MPI Forum on implementation and usability issues [47]. The CH part of the name comes from "Chameleon". Chameleon, symbol of adaptability to the environment and therefore portability, was the portable parallel programming library/layer, used to provide portability to the existing message-passing systems (see Gropp and Smith 1993 [44]).

PETSc:
The *Portable, Extensible Toolkit for Scientific Computation (PETSc)*, is a suite of data structures and routines for the parallel solution modelled by partial differential equations (see Balay et al. 2019 [11]). PETSc uses the Message Passing Interface (MPI) standard for all message-passing communication. PETSc consists of a variety of libraries, which manipulate a particular family of objects (for example parallel matrix and vector assembly routines that allow the overlap of communication and computation). PETSc is used for this project, because it hides the MPI from the user, and because of the previous work done by Aage et al. (2015) [2].

A.1.2. Settings
In order to use PETSc, the MPICH software has to be used. There is no need to directly program the message passing with MPI, but it is only useful to be familiar with the basic concepts of message passing and distributed memory computing. Firstly, the procedure to begin a MPI job must be known to execute a PETSc program. In this project, this is done by this:

```
mpirun -np <number of processes> <program name and arguments>
```

`mpirun` is a shell script that will cover up for all the differences in starting jobs and running machines. The only input is the number of processes, and the name of the program that has to start. During the writing of the code for a project, it is advised to use multiple processes, to take out MPI problems as soon as possible.

Note that both `mpirun` and `mpiexec` can be used to begin a MPI job. MPI-1 did not specify a standard on how to start the programm. MPICH used `mpirun`. From MPI-2, the recommended standard is `mpiexec`, which is also found in all documentation. However, the command `mpirun` does exactly the same and is still supported. See Gropp et al. (1999) [45] for more background information.

Next, the functions are listed that are used in the main file. These are all PETSc functions, and all have been used to build the CutFEM topology optimization model.

- `#include<>`
  This is used by the preprocessor to include "header files" which have fragments of code. This does not work for prototypes of functions or in-line functions.

- `int main(int argc, char **argv)`
  The purpose of this return value is to return an integer representing the application software status, so also an exit status to the operating system. The arguments *argc* and *argv* are the command line arguments which are delivered in all C and C++ programs, and usually include the number of arguments and the program name itself.

- `PetscInitialize(int *argc,char ***argv,char *file,char *help);`
  This initializes PETSc and MPI, by defining the the command line arguments *argc* and *argv*. By default this file is called `.petscrc` in the user's home directory, but this can be changed with `PETSC_OPTIONS`. `PetscInitialize` calls the MPICH function `MPI_Init()`.

- `PetscErrorCode`
  This is a datatype that is used for returning an error code from almost all PETSc functions. The error code is set to nonzero if an error has been detected. This is also the return value of most routines.

- `PetscFinalize();`
  This routine handles options to be called at the end of the program. `PetscFinalize` calls the MPICH function `MPI_Finalize()`.

A.2. Solver

The procedure that is used for the solver is as follows:

1. `SetUpSolver()`
   This is a self written function, in which the following functions are called:
   - `KSPCreate(MPI_Comm comm,KSP *ksp)`
     This is used to create a solver context. In here, `comm` is the MPI communicator, and `ksp` is the solver context that is created. This follows the multigrid implementation presented in Amir et al. (2014) [7].
   - `KSPSetOperators(KSP ksp,Mat Amat,Mat Pmat)`
     This is used to set the matrix of the linear system (`Amat`) and the matrix of the preconditioner (`Pmat`).
   - `KSPSetFromOptions(KSP ksp)`
     This is used to set the KSP options, from the options database. This is for example used to set the maximum number of linear iterations.
2. `KSPSetUp(KSP ksp)`
   This is used to set up the internal data structures, that are later used in the solver. This is for example in order to utilize the preconditioners. The settings for the preconditioners must have been set already.
3. `KSPSolve(KSP ksp,Vec b,Vec x)`
   This is used to solve the linear system. In here, `b` is the right-hand-side, and `x` are the solution vectors of the system.
4. `KSPDestroy(KSP *ksp)`
   This is used to destroy the KSP context, once it is no longer needed.

The options that are set in the `SetUpSolver()` are as follows:

- **Main solver:** `KSPFGMRES` ("fgmres")
  This setting is set with `KSPSetType` and implements the *Flexible Generalized Minimal Residual method.*
- **Preconditioner:** `PCMG` ("mg")
  This setting is set with `PCSetType` and implements multigrid preconditioning. This does require to give extra information in the multigrid settings, for example with `PCMGSetLevels`.
- **Level 0-3 smoother:** `KSPGMRES` ("gmres")
  This setting is set with `KSPSetType` and implements the the Generalized Minimal Residual method.
- **Level 0-3 prec:** `PCGAMG` ("gamg")
  This is the native geometric algebraic multigrid (AMG) preconditioner.
- **Max iterations:**
  With `KSPSetTolerances` the maximum amount of iterations for the convergence testers on the coarse and fine grids are set. For the coarsest grid this is set 30, for the finer grids this is set to 4.

However, there are also other solving methods possible. For the finite difference check, the tolerances have to be set really small and the solving method is set to a direct solver. The external package `MUMPS` can do this. However, the speed and memory consumption scale with $n * n$ so this is not a good option for bigger systems.

B
Tetrahedralization

B.1. Marching Cubes Ambiguity

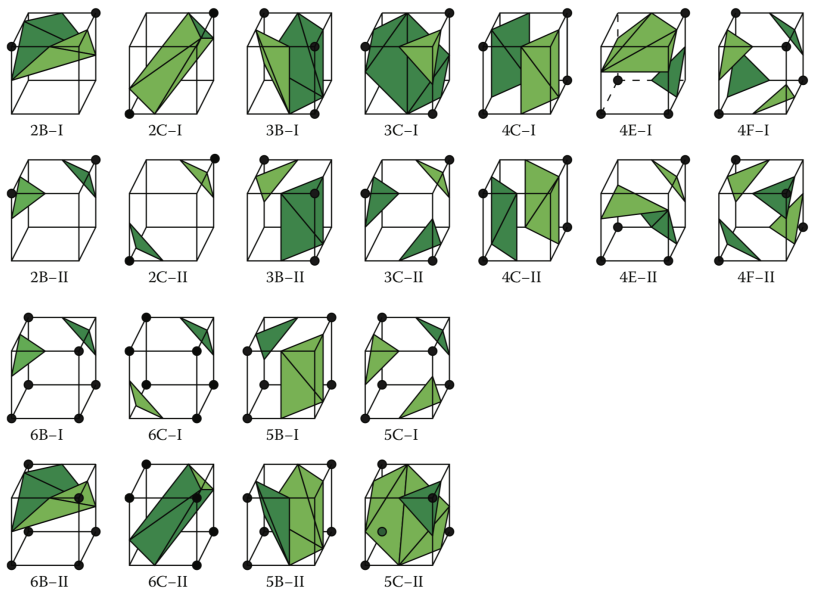


Figure B.1: The ambiguity in Marching Cubes, from [89]

B.2. Lookup tables

In order to know which points and facets to put into TetGen for each cubeindex, multiple lookup tables are used. It is considered faster to calculate the cuts for one element one time and look it up, compared to calculating it for each cut element again. The lookup tables are created with a Matlab code. The 6 tables which are created are: `SolidTable`, `FluidTable`, `FacetSolLenTable`, `FacetSolTable`, `FacetFluLenTable`, and `FacetFluTable`. The 6 tables contain the following are used to describe the following (the full conventions can be seen in section B.4 at the end of this appendix):

• `SolidTable`:	Points that describe the solid parts
• `FluidTable`:	Points that describe the fluid parts
• `FacetSolLenTable`:	Number of points of the facets that describe the solid parts.
• `FacetSolTable`:	Points that describe the facets in solid parts.
• `FacetFluLenTable`:	Number of points of the facets that describe the fluid parts.
• `FacetFluTable`:	Points that describe the facets in fluid parts.

Code:
Before the code is described, it is good to mention the used conventions. The corner points or vertex are referred to as *nodes*. The points which mark the cut on the edge are referred to as *edge points*. Because parts have to be described with both nodes and edge points, a new numbering convention is used. This is based on the often used convention for nodes (0-7) and edges (0-11). Now, the edge points are just the number of the edge plus 8:

```
    4---------5      *---4-----*          4---12---5
   /|        /|     /|        /|         /|       /|
  / |       / |    7 |       5 |       15 |      13|
 /  |      /  |   /  8      /  9       /  16     /  17
7---------6   |  *-----6---*   |      7----14--6   |
|   |     |   |  |   |     |   |      |   |    |   |
|   0-----|---1  |   *---0|---*       |   0---8|---1
|  /      |  /   11 /     10 /       19  /    18  /
| /       | /    | 3       | 1       | 11      | 9
|/        |/     |/        |/        |/        |/
3---------2      *---2-----*         3---10---2
```

Figure B.2: The convention on the numbering a) the node numbering b) the edge numbering c) new convention for both nodes and edge points

Now, the code that was used to obtain the lookup tables is described. In the Matlab code, the convention is used that a node with value 1 is solid and a node with value 0 is fluid:

1. Start
    - Initialize:
      Information on node connection and facet connection
2. Loop over all 256 cubes for the node tables, see Figure B.3
    - Use the cubeindex to obtain the node state information.
    - Group the solid nodes together with their connected nodes:
      Only nodes which are connected directly connected by one edge, are considered connected.
    - Group all unconnected nodes:
      In case of double cut element, there is often one bigger part with multiple connected nodes, and one or more loose nodes.
    - Add the edge points to the relevant groups.
    - Group the fluid nodes and the edge points:
      This is done similar to the solid points.
3. Loop over all 256 cubes for the facet tables, see Figure B.4
    - Use the node information to obtain the facet information:
      The facets refer to the points to which they are connected. This has to be in the right order. Facets can be described by a different amount of points, so the number of points in the facet is also saved. The facets on the inside are always described by triangles because of the ambiguity.
    - For double cut element, the facets should be matched to right part, see Figure B.5.
    - Group the fluid facets:
      This is done similar to the solid points.
    - Reorder the point for TetGen usage:
      The numbering of referencing to the points should match the input number in TetGen, not the numbering from the original cube convention.
4. Write TetGen Input tables and print the cube layout

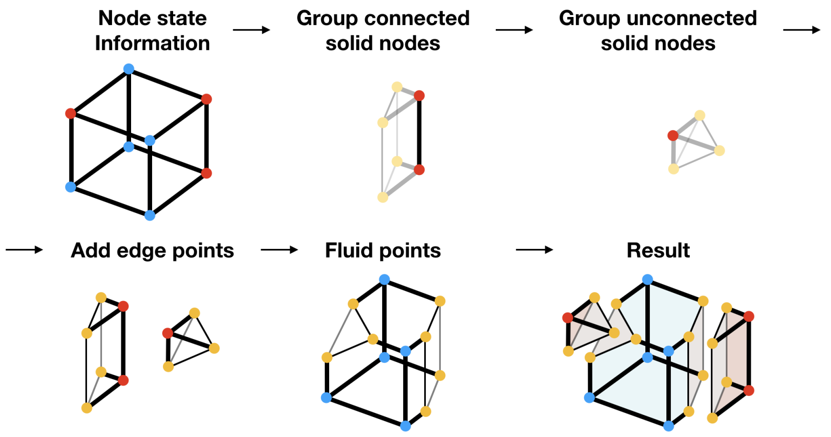


Figure B.3: The process of finding the nodes

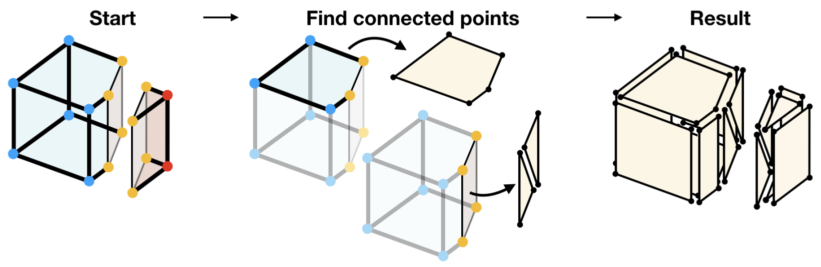


Figure B.4: Finding the facets for points that are connected

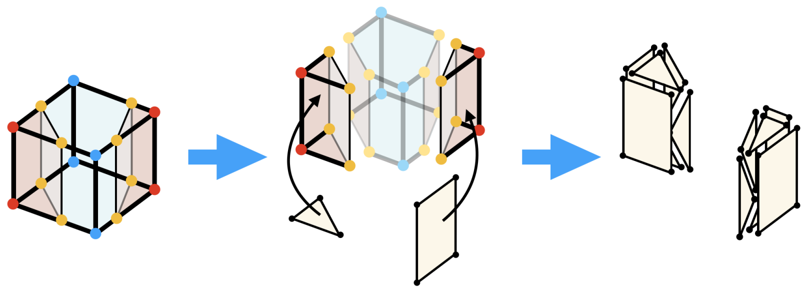


Figure B.5: For elements with a double cut, the facets have to be matched to the right (solid) part

B.3. TetGen

TetGen is a program to generate tetrahedral meshes of any 3D polyhedral domains by employing a form of Delaunay triangulation (see Si 2013 [73] and Si 2015 [74]). The *Delaunay triangulation* is named after Boris Delaunay (1934) [33], and maximizes the minimum angle of all the angles of the triangles in the triangulation of a given set of discrete points. This is done by ensuring that no point is inside the circumcircle of any triangle in the triangulation.

The input of TetGen is either a 3D point set, or a 3D piecewise linear complex (PLC), or a tetrahedral mesh. The *piecewise linear complex (PLC)*, was introduced by Miller et al (1996) [65], and represents a 3D polyhedral domain with a set of "cells" such as vertices, edges, polygons, and polyhedra. This is the input that was used, as this can also contain information on the where the cutting edge is. The output of TetGen is either a Delaunay (or weighted Delaunay) tetrahedralization, or a constrained (Delaunay) tetrahedralization, or a quality tetrahedral mesh.

TetGen1.5.1:
TetGen has been incorporated into other software packages and also PETSc. However, the version downloaded by the PETSc configuration is `TetGen1.4.3`. This version gave at arbitrary moments the following error message:

```
            Assertion failed:  (neightet.tet != dummytet), function carvecavity, file
tetgen.cxx, line 21703.
```

Which refered to the function: "`tetgenmesh::carvecavity`" where in line 21703 the following procedure is performed:

```
            // Find an interior tet.
           assert(neightet.tet != dummytet); // SELF_CHECK
```

For some cut element configurations this occured, but if the same configuration did not per definition lead to this error. The problem did not occur when only 1 processor was used. The problem was fully solved by installing a new TetGen version, `TetGen1.5.1`. This was downloaded, the `libtet` function file was made, and referred to from the makefile of the cutfem code. To prevent mixing of versions, it is advised to use a petsc configuration without tetgen.

Input:
In the code the tetrahedralization of a cut element is done per part, so first the solid part(s), and next the fluid part. This is done separately to make sure that the tetrahedrons do not cross the cutting edge, and to connect the right phase state to the tetrahedrons which come out. The procedure comes down to putting the right information in and taking the right information out of the `tetgenio`. The *tetgenio* is a structure for transferring data into and out of TetGen's mesh structure. The loop over the number of parts (both solid and fluid) is as follows:

1. Initialize TetGen object `tetgenio`
2. Put the points information in `tetgenio`
3. Put the facet information in `tetgenio`
4. Tetrahedralize the part with `tetrahedralize`
5. (optional:) write the output in a file
6. Save the tetrahedron output from the `tetgenio` output

The used TetGen function `tetrahedralize(char *switches, tetgenio *in, tetgenio *out)` is an interface for using TetGen's library to generate Delaunay tetrahedralizations. The *switches* are used to control the behaviour of TetGen and to specify the output. The *in* contains a PLC that has to be tetrahedralized, and *out* is used for storing the generated tetrahedral mesh.

The switches that are used in the computations are as follows:

- **-p** Tetrahedralizes a piecewise linear complex (PLC):
  The boundary description (a surface mesh) of a 3d piecewise linear complex (PLC) is read and a tetrahedral mesh of the PLC is generated. By default, TetGen generates a constrained Delaunay tetrahedralization (CDT) of the PLC. In TetGen this sets the variable `plc`.

- **-Y** Preserves the input surface mesh (does not modify it):
  The input boundary edges and faces of the PLC are preserved in the generated tetrahedral mesh. Steiner points appear only in the interior space of the PLC. In TetGen this sets the variable `nobisect`.

- **-T** Sets a tolerance for coplanar test, the default value is 10^{-8}:
  The vertices which are used to define a facet of a PLC should be exactly coplanar. But this is very hard to achieve in practice due to the inconsistent nature of the floating-point format used in computers. TetGen accepts facets whose vertices are not exactly but approximately coplanar. Furthermore, this tolerance is also used in checking if not all facets are in the same plane. Because small elements are used, points might lie very close to each other. In TetGen this sets the variable `epsilon`.

  The variable was changed to avoid the following error message appeared:
  ```
  Exception:  All vertices are coplanar (Tol = 1e-08)
  ```
  The variable was set to 10^{-19}. During the project, this has been sufficiently low, to not get this error message.

- **-p/** Sets a tolerance for intersection test, the default is 10^{-1}:
  If two subfaces, ($f_1[a,b,c]$ and $f_2[a,b,d]$), share the same edge ($[a,b]$), the dihedral angle between these two facets, should be higher this tolerance. In TetGen this sets the variable `facet_overlap_ang_tol`.

  The variable was changed to avoid the following error message appeared:
  ```
  Found two nearly self-intersecting facets.
  1st:  [5, 6, 1] #0
  2nd:  [5, 6, 3] #0
  The dihedral angle between them is 0.0181848 degree.
  Hint:  You may use -p/# to decrease the dihedral angle tolerance 0.1 (degree).
  ```
  The variable was set to 10^{-5}. During the project, this has been sufficiently low, to not get this error message.

- **-Q** Quiet, no terminal output except errors:
  This is only used when it was ensured that TetGen performs as expected. In TetGen this sets the variable `quiet`.

The input information on the points consists of the number of points and a pointlist with the x, y, and z coordinate of each point. From all points in Figure B.2 the coordinates are calculated in the local coordinate system with domain `[0, dx]`. Only the relevant points for each part are used as input. The information on the facets consists of the number of facets, for each polygon the number of vertices, and a vertexlist which refers to all the points that form a polygon. These points refer to the points that were put in earlier, and they should be ordered correctly.

The output information consists of the number of cells (tetrahedrons), the coordinates of the points, and a list which refers all the points that form a tetrahedron. For all parts this is saved into a variable for the whole cut element. Another variable keeps track of the phase of each tetrahedron.

B.4. Matlab Tables Convention

```
SolidTable Conventions
1st column:          number of separate solid parts
2-5th columns:       number of points per part
6th-21st columns:    node numbers

FluidTable Conventions
1st column:          number of separate fluid parts
2-3th columns:       number of points per part
4th-20th columns:    node numbers

FacetSolLenTable Conventions
1st column:          number of separate solid parts
2-5th columns:       number of facets per part
6th-21st columns:    number of points per facet

FacetSolTable Conventions
1st column:          number of separate solid parts
2-5th columns:       number of points to describe facets
6th-21st columns:    node numbers that describe facets

FacetFluLenTable Conventions
1st column:          number of separate solid parts
2-5th columns:       number of facets per part
6th-21st columns:    number of points per facet

FacetFluTable Conventions
1st column:          number of separate solid parts
2-5th columns:       number of points to describe facets
6th-21st columns:    node numbers that describe facets

Cube Conventions
0-7 are node numbers, 8-19 are edge numbers
```

C
Adjoint method

In this section, the mathemathics behind the adjoint method are explained. This method is used to calculate the sensitivities. As stated in Sharma et al. (2017) [72], in topology optimization there are three methods in which the shape sensitivities can be calculated: the *finite difference method*, the *direct method* (AD), and the *adjoint method* (AD). First, the finite difference method is explained. For more information, the reader is referred to Haug et al. (1992) [49] or Kollman (2010) [57].

Finite Difference method:

In the finite difference method, the shape parameter of interest is perturbed and the response of the model for the perturbed values are reanalyzed. An approximation to the gradient vector $\nabla f(s)$ can be obtained by evaluating the function f at $(n+1)$ points and performing some elementary arithmetic. A popular formula for approximating the partial derivative $\partial f/\partial s_i$ at a given point x is the *forward-difference*, or *one-sided-difference*, approximation, defined as:

$$\frac{\partial \Psi}{\partial s_i}(s) \approx \frac{\Delta \Psi}{\Delta s_i} = \frac{\Psi(s+\epsilon e_i) - \Psi(s)}{\epsilon} \tag{C.1}$$

This method is not used, because it requires the complete system to be solved for every design variable. This method is used to compare the sensitivities of the adjoint method in the finite difference check.

Adjoint method:

Consider a general function Ψ as shown in Equation C.2, that may represent any of the performances measured in a structure, and which is explicit dependent on the design parameters s, and implicit dependent through the solution u of the state equations. The objective of the design sensitivity analysis is to determine the total dependence of these functions on the design, thus to find $\frac{d\Psi}{ds}$.

$$\Psi = \Psi(\mathbf{s}, \mathbf{u}(\mathbf{s})) \tag{C.2}$$

Using the chain rule, the differative of Ψ with respect to s can be calculated as follows:

$$\frac{d\Psi}{d\mathbf{s}} = \frac{\partial \Psi}{\partial \mathbf{s}} + \frac{\partial \Psi}{\partial \mathbf{u}} \cdot \frac{d\mathbf{u}}{d\mathbf{s}} \tag{C.3}$$

Now consider the structural state problem, where both sides are differentiated to s:

$$\mathbf{K}(\mathbf{s})\mathbf{u} = \mathbf{F}(\mathbf{s}) \qquad \rightarrow \qquad \frac{\partial(\mathbf{Ku})}{\partial \mathbf{s}} = \frac{\partial \mathbf{F}}{\partial \mathbf{s}} \tag{C.4}$$

This can be rewritten as follows:

$$\mathbf{K}(\mathbf{s})\frac{d\mathbf{u}}{d\mathbf{s}} = -\frac{\partial}{\partial \mathbf{s}}(\mathbf{K}(\mathbf{s})\tilde{\mathbf{u}}) + \frac{\partial \mathbf{F}(\mathbf{s})}{\partial \mathbf{s}} \tag{C.5}$$

In the above equation, $\tilde{\mathbf{u}}$ indicates that this is kept constant during the process of the partial differentiation.

Now, using that the matrix $\mathbf{K}(\mathbf{s})$ is non-singular Equation C.5 can be solved for $\frac{d\mathbf{u}}{d\mathbf{s}}$ and substituted into Equation C.3 to obtain the final gradients:

$$\frac{d\Psi}{d\mathbf{s}} = \frac{\partial\Psi}{\partial\mathbf{s}} + \frac{\partial\Psi}{\partial u}\mathbf{K}^{-1}(\mathbf{s})\frac{\partial}{\partial\mathbf{s}}\left[\mathbf{F}(\mathbf{s}) - (\mathbf{K}(\mathbf{s})\tilde{\mathbf{u}})\right] \tag{C.6}$$

Right now, one solution of problem Equation C.6 is needed for each design parameter **s**.

To overcome the problem of solving for each design parameter **s**, in the adjoint approach the *adjoint variable* λ is defined as follows:

$$\boldsymbol{\lambda} \equiv \left[\frac{\partial\Psi}{\partial\mathbf{u}}\mathbf{K}^{-1}(\mathbf{s})\right]^T = \mathbf{K}^{-1}(\mathbf{s})\frac{\partial\Psi^T}{\partial\mathbf{u}} \qquad \rightarrow \qquad \mathbf{K}^T\boldsymbol{\lambda} = -\left(\frac{\partial\Psi}{\partial\mathbf{u}}\right)^T \tag{C.7}$$

Equation C.7 can be substituted in Equation C.6 to obtain the following:

$$\frac{d\Psi}{d\mathbf{s}} = \frac{\partial\Psi}{\partial\mathbf{s}} + \boldsymbol{\lambda}^T\left[\frac{\partial\mathbf{F}(\mathbf{s})}{\partial\mathbf{s}} - \frac{\partial}{\partial\mathbf{s}}(\mathbf{K}(\mathbf{s})\tilde{\mathbf{u}})\right] = \frac{\partial\Psi}{\partial\mathbf{s}} + \frac{\partial}{\partial\mathbf{s}}\left[\tilde{\boldsymbol{\lambda}}^T\mathbf{F}(\mathbf{s}) - \tilde{\boldsymbol{\lambda}}^T(\mathbf{K}(\mathbf{s})\tilde{\mathbf{u}})\right] \tag{C.8}$$

As can be seen, in the adjoint method, the linear system of equations is solved as many times as number of response functions.

Note that another way to write the new objective function with the residual **R** is as follows:

$$L = \Psi + \boldsymbol{\lambda}^T\mathbf{R} \qquad \text{with:} \qquad \mathbf{R}(\mathbf{s},\mathbf{u}(\mathbf{s})) = \mathbf{K}\mathbf{u} - \mathbf{F} = 0 \tag{C.9}$$

Now, simply rephrasing Equation C.8 (with **F** independent of **s**) gives that the adjoint equation is as follows:

$$\frac{d\Psi}{d\mathbf{s}} = \frac{\partial\Psi}{\partial\mathbf{s}} + \boldsymbol{\lambda}^T\frac{\partial\mathbf{K}}{\partial\mathbf{s}}\mathbf{u} \tag{C.10}$$

Remember that following relations are known:

$$\begin{gathered}\frac{\partial\Psi}{\partial\mathbf{s}} = \mathbf{u}^T\frac{\partial\mathbf{K}}{\partial\mathbf{s}}\mathbf{u}\\ \boldsymbol{\lambda} = \mathbf{K}^{-1}(\mathbf{s})\frac{\partial\Psi^T}{\partial\mathbf{u}} = 2\mathbf{u}\end{gathered} \tag{C.11}$$

Filling in gives that the sensitivities for minimum compliance can be calculated as follows:

$$\frac{d\Psi}{d\mathbf{s}} = \mathbf{u}^T\frac{\partial\mathbf{K}}{\partial\mathbf{s}}\mathbf{u} - 2\mathbf{u}^T\frac{\partial\mathbf{K}}{\partial\mathbf{s}}\mathbf{u} = -\mathbf{u}^T\frac{\partial\mathbf{K}}{\partial\mathbf{s}}\mathbf{u} \tag{C.12}$$

Where the element sensitivities with respect to **s** can be calculated as follows:

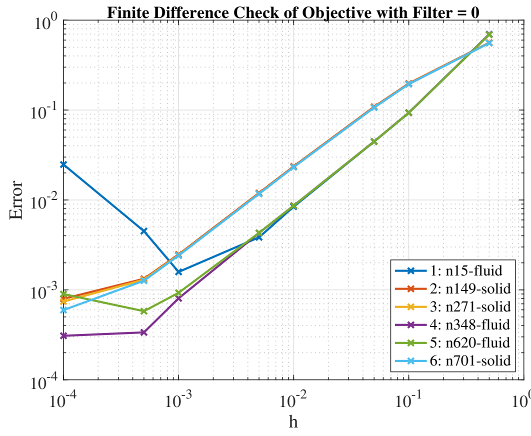
$$\frac{\partial\mathbf{K}}{\partial\mathbf{s}} = \sum_e^n\left(\frac{\partial k_e}{\partial s_i}\right) \qquad \text{with} \qquad \frac{\partial k_e}{\partial s_i} = \frac{k_e - k_e^{pert}}{\delta s_i} \tag{C.13}$$

D Finite Difference Results

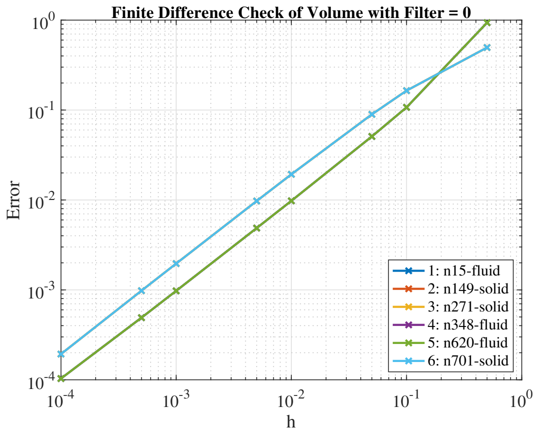
In this Appendix, the Finite Difference (FD) results are presented. The Finite Difference (FD) check is done in order to test the sensitivity function. First, the FD test setup will be described. Next, the results will be presented. Note that the goal of the FD check is not to compare results with external data, but rather to check the convergence and consistency of the sensitivities.

Setup:
In the FD test, the sensitivities first are calculated as usual. Next, the phase state of a node will be perturbed (by a high perturbation value) which also gives a FD sensitivity value. The original and FD sensitivities can be compared, where the difference is called the *error*. The perturbation value of the FD, referred to as *h*, is then decreased. If the error also decreases there is convergence.
The input structure for the tests is taken from the *Beam with CutFEM* problem in Section 4.1.2. The sensitivities should be check on random nodes, so both solid and fluid, both on top and at the bottom of the beam, both in the centre and at the side. The perturbation for the sensitivities is, unless mentioned, set to $epsilon = 1.0 * 10^{-5}$, which is low enough for a reliable FD check. The perturbation for the FD sensitivities changes, as explained.

Three tests were performed, for three different filter settings. The `filter = 0` test uses no filter. The `filter = 1` test uses the density filter. The `filter = 2` test uses the density filter and Heaviside projection. The radius is set to $rmin = 0.21$, which results in a stencil of 3 elements. The Heaviside steepness variable β is set to $\beta = 12$.
Results:
The results can be seen in Figure D.1, D.2, and D.3. The rank and node number are mentioned in the legend.

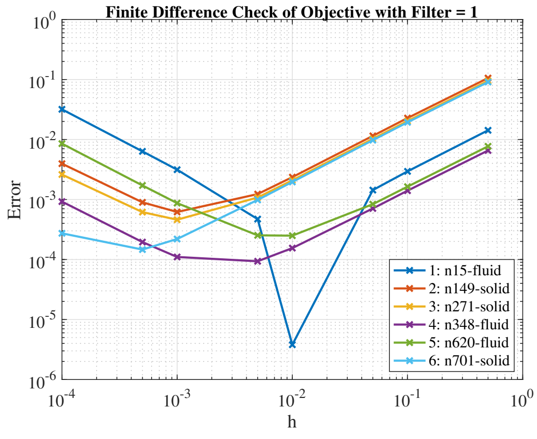


(a) Objective sensitivities

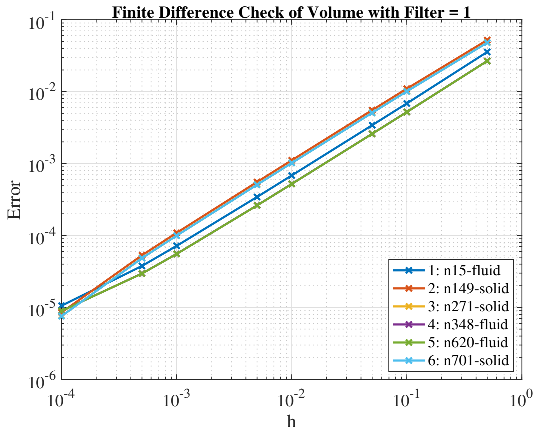


(b) Volume sensitivities

Figure D.1: Finite Difference check, for no filter

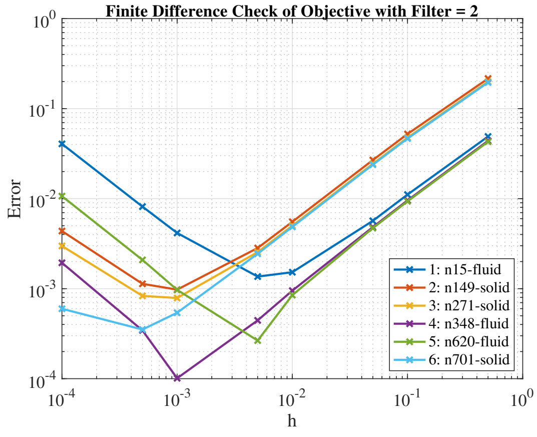


(a) Objective sensitivities

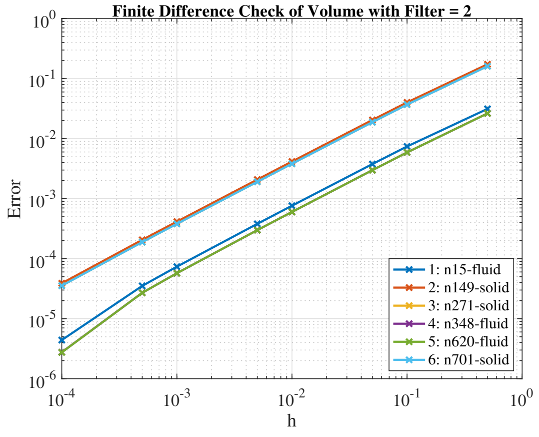


(b) Volume sensitivities

Figure D.2: Finite Difference check, for a density filter



(a) Objective sensitivities



(b) Volume sensitivities

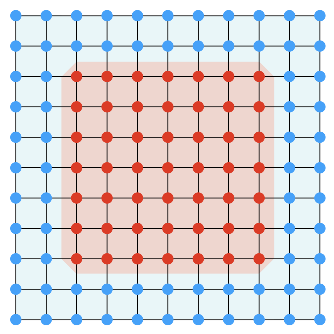
Figure D.3: Finite Difference check, for a density filter and Heaviside projection

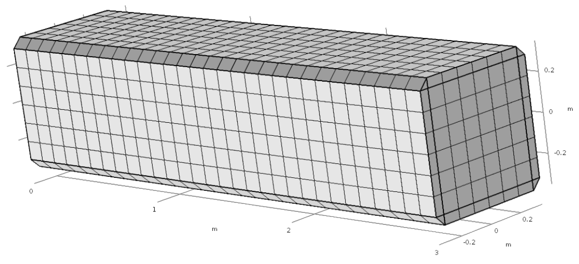
As can be seen, again error is linearly related to the perturbation. It can be concluded that the sensitivities function works.

E Finite Element Analysis Results

Here, two more tests from Section 4.1 are presented.

Beam with CutFEM 2:
Again a beam is tested with rows of Cut Elements, this time on all sides of the beam. This shows the performance of the stiffness matrix of the CutFEM elements in multiple directions. The used models can be seen in Figure E.1. The results can be seen in Figure E.2 and Table E.1.


(a) Front view of the input for sPhys



(b) Design input in Comsol

Figure E.1: Design input

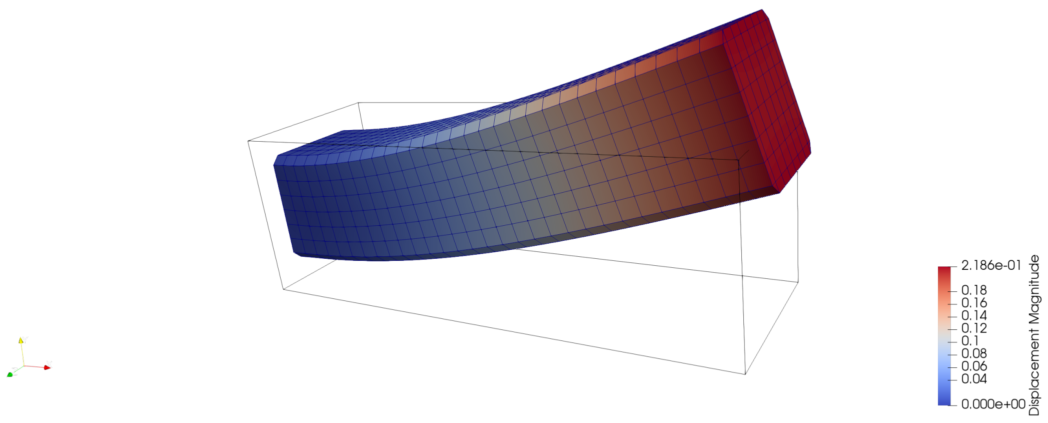


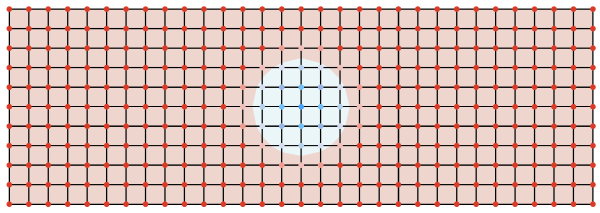
Figure E.2: Deflected beam without CutFEM

Table E.1: Deflection of beam with CutFEM

Direction	PETSc[m]	Comsol[m]	Difference [%]
Displacement Y	0.02155	0.21542	0.037
Displacement tot	0.02186	0.21768	0.42

As can be seen, there is a difference in the displacement similair to the beam without CutFEM.

Beam with hole:
In this test a beam is tested which has got a hole in the middle. This shows the performance of the PETSc code with different CutFEM element shapes. Note that the CutFEM elements are of less influence than the previous test. The used models can be seen in Figure E.3. The results can be seen in Figure E.4 and Table E.2.


(a) Side view

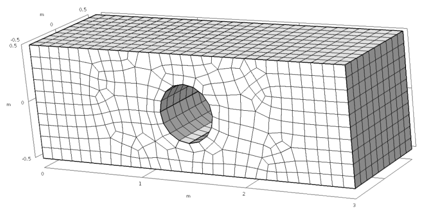

(b) Design input in Comsol

Figure E.3: Design input

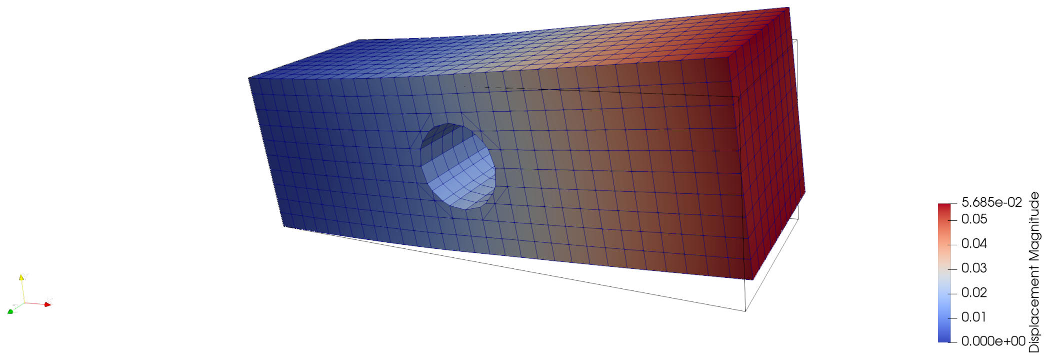


Figure E.4: Deflected beam with hole

Table E.2: Deflection of beam with hole

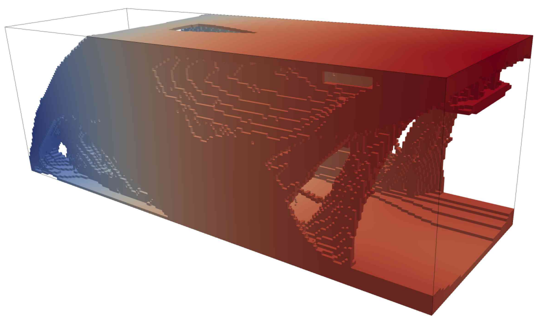
Direction	PETSc[m]	Comsol[m]	Difference [%]
Displacement Y	0.05568	0.055493	0.34
Displacement tot	0.05685	0.056885	0.062

As can be seen, the CutFEM elements also work when they have different shapes.

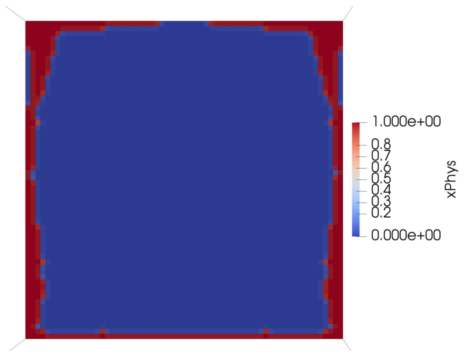
F Optimization Results

F.1. MBB problem

In this section, the results for the MBB problem are presented. For the input, see section 4.2.1. The resulting structed for this problem with the *SIMP rmin = 0.016* and the *SIMP rmin = 0.08* tests can be seen in Figure F.1 and F.2. Figures F.1b and F.2b show a slice of the phase at 0.5 of the beam. The initial and resulting structure for this problem with the *CutFEM* and the *CutFEM cont* tests can be seen in Figure F.4. For all the steps in between, Section F.2.

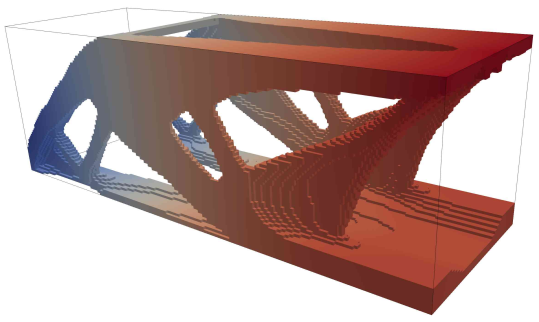


(a) Optimized structure. The color represents deflection.

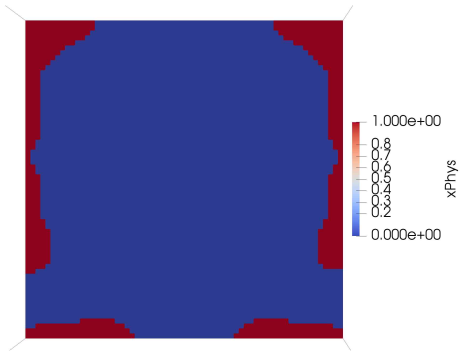


(b) Cutout at 0.5 of the beam.

Figure F.1: Optimized structure for the *SIMP rmin = 0.016* test of the cantilever beam problem.

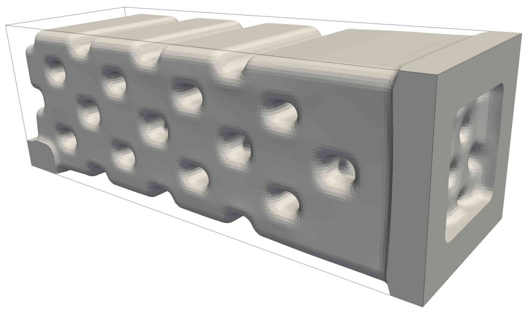


(a) Optimized structure. The color represents deflection.



(b) Cutout at 0.5 of the beam.

Figure F.2: Optimized structure for the *SIMP rmin = 0.08* test of the MBB beam problem, with Heaviside projection with final $\beta = 500$.


(a) Initial design.

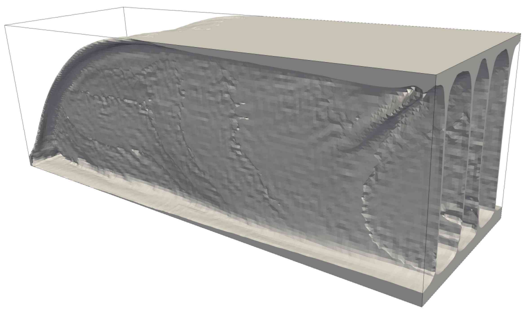
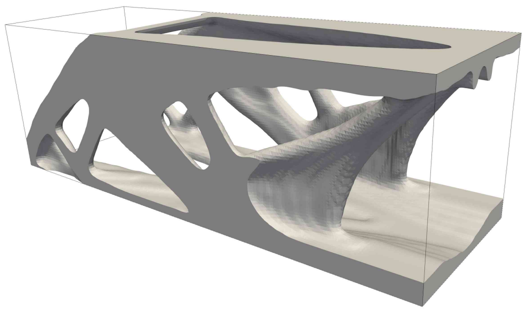

(b) Optimized structure, iteration 1000.

Figure F.3: Structures for *CutFEM* test of the MBB beam optimization.


(a) Initial design.

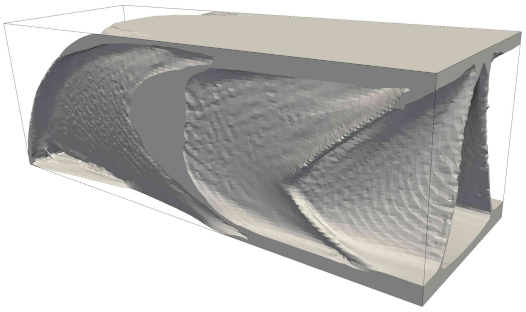
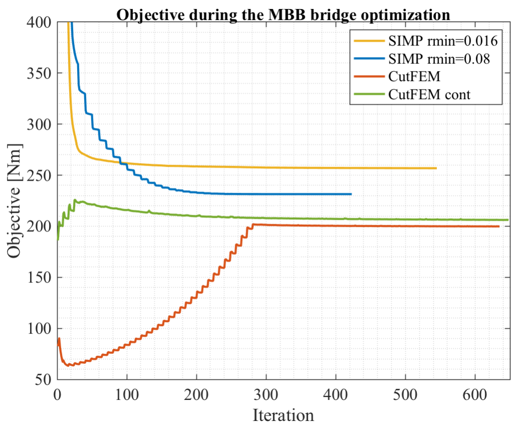
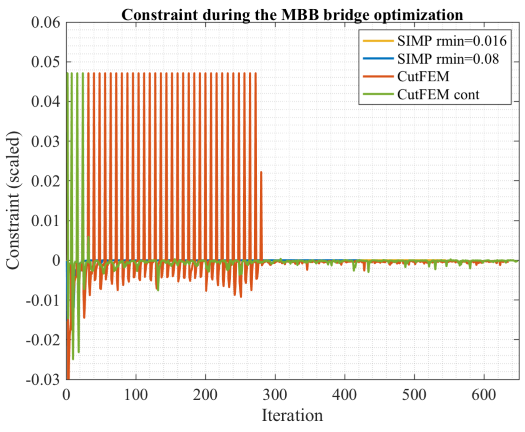

(b) Optimized structure, iteration 1000.

Figure F.4: Structures for *CutFEM cont* test of the MBB beam optimization.

Also, the development of the different parameters have been checked throughout the optimization. The development of the objective and the constraint at every iteration can be seen in Figure F.5. The development of the calculation time, the number of iterations to solve, and the number of cut elements in the structure can be seen in Figure F.6

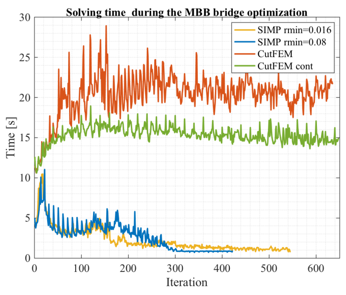


(a) Objective

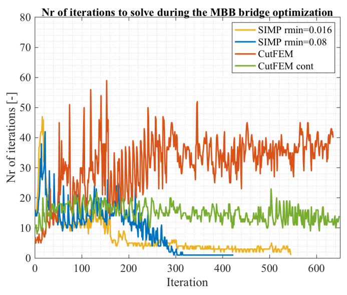


(b) Constraint

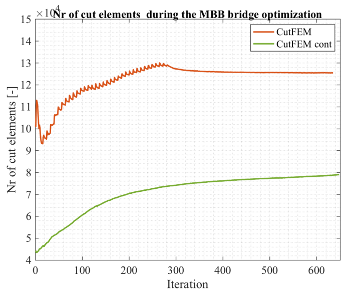
Figure F.5: Function variables at every iteration.



(a) Time per iteration



(b) Number of iterations to solve

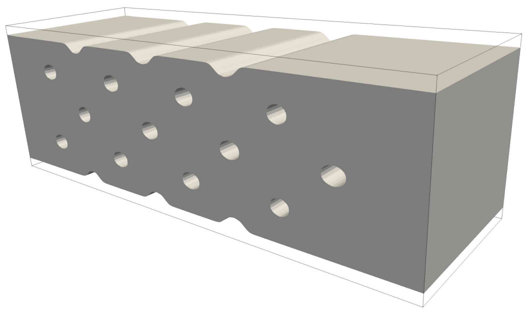


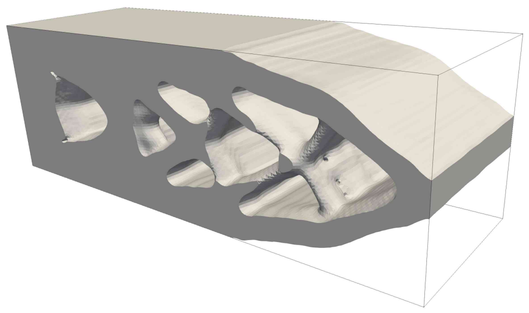
(c) Number of cut elements in structure

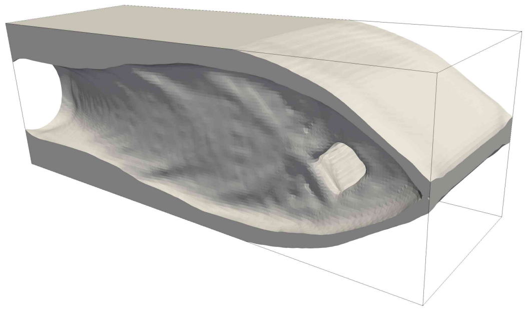
Figure F.6: Function variables at every iteration.

F.2. Structure progress during optimization

Cantilever beam:


(a) Iteration 0 (initial design)


(b) Iteration 25


(c) Iteration 60

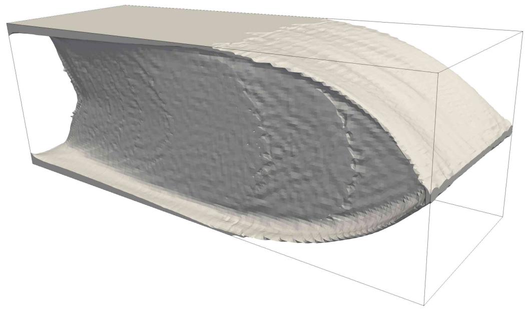
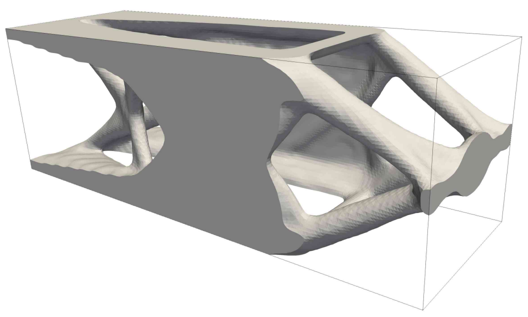
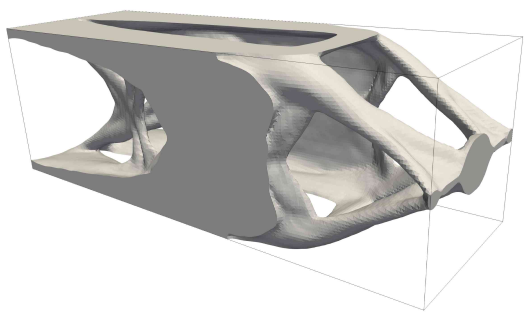
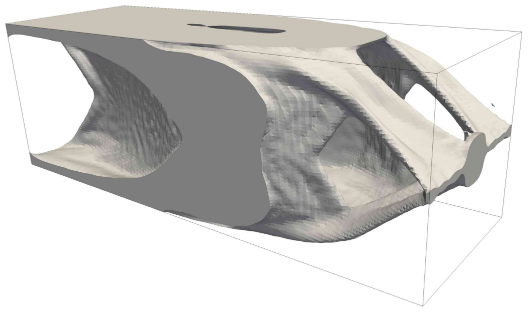

(d) Iteration 1000

Figure F.7: Structure for the *CutFEM* test of the cantilever beam problem during optimization. For input, see section 4.2.1


(a) Iteration 0 (initial design)


(b) Iteration 30


(c) Iteration 120

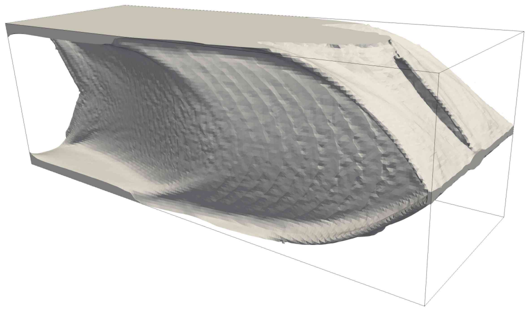
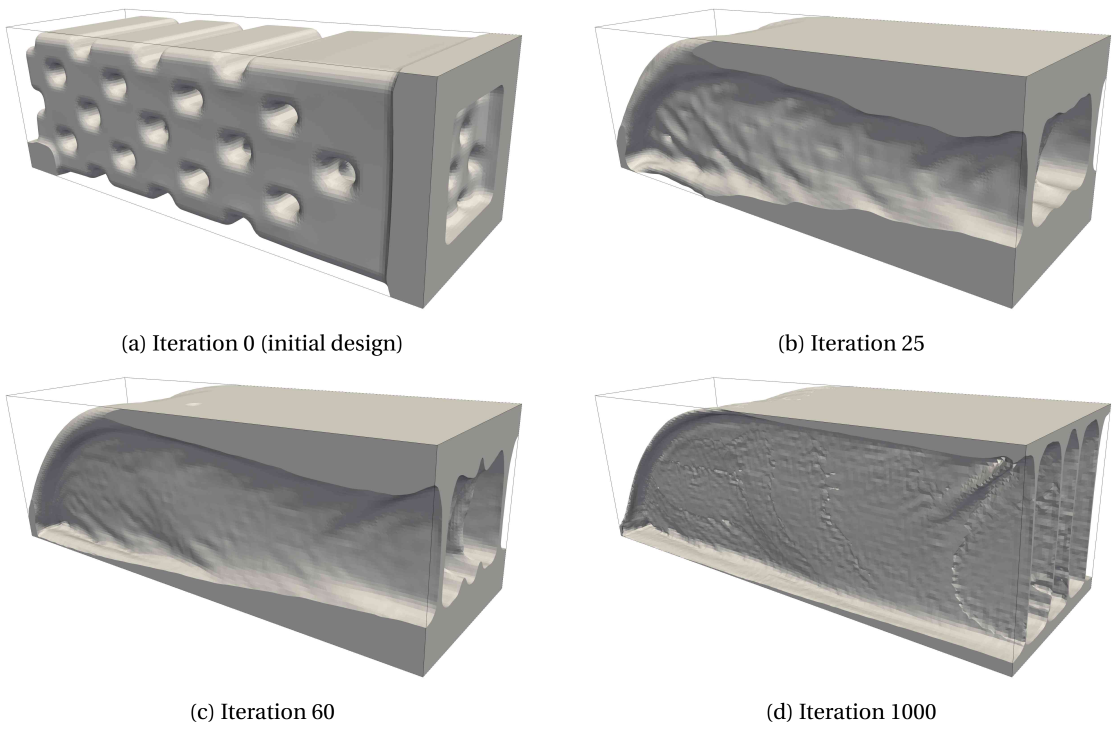

(d) Iteration 1000

Figure F.8: Structure for the *CutFEM cont* test of the cantilever beam problem during optimization. For input, see section 4.2.1

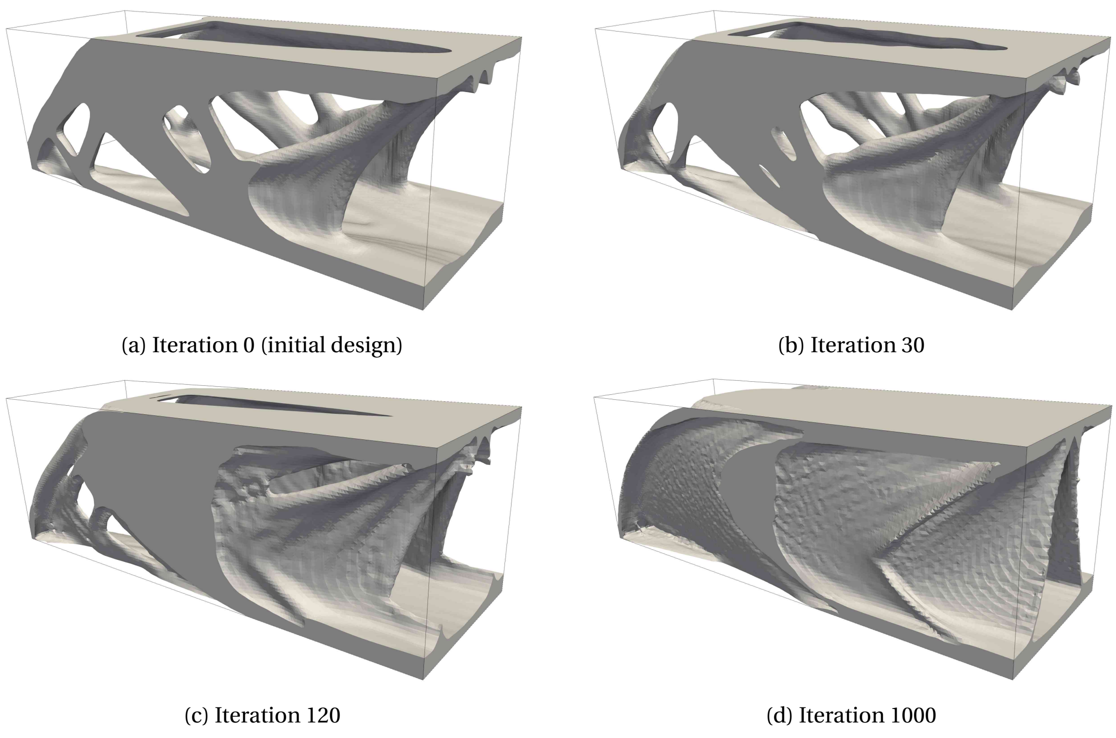
MBB beam:



(a) Iteration 0 (initial design) (b) Iteration 25

(c) Iteration 60 (d) Iteration 1000

Figure F.9: Structure for the *CutFEM* test of the MBB beam problem during optimization. For input, see section 4.2.1



(a) Iteration 0 (initial design) (b) Iteration 30

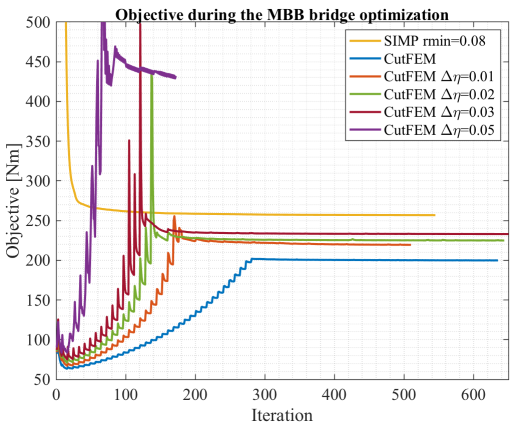
(c) Iteration 120 (d) Iteration 1000

Figure F.10: Structure for the *CutFEM cont* test of the MBB beam problem during optimization. For input, see section 4.2.1

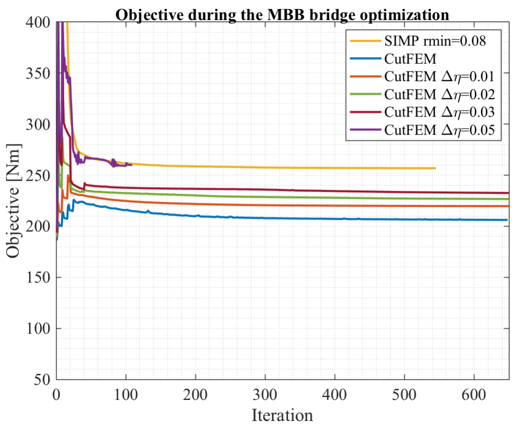
G

Robust optimization results

G.1. MBB test objective function



(a) *CutFEM* test

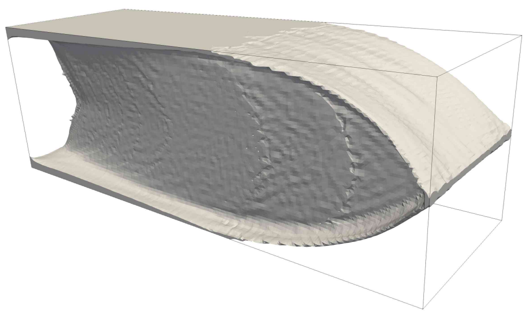


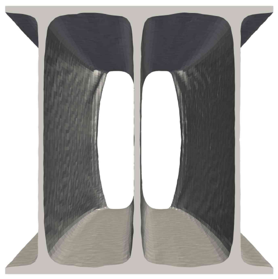
(b) *CutFEM cont* test

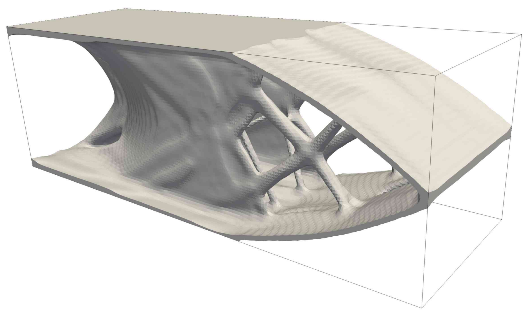
Figure G.1: Objective functions for different threshold offset values for the MBB beam tests. For input, see section 4.3

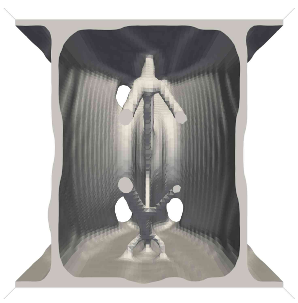
G.2. Robust design layouts

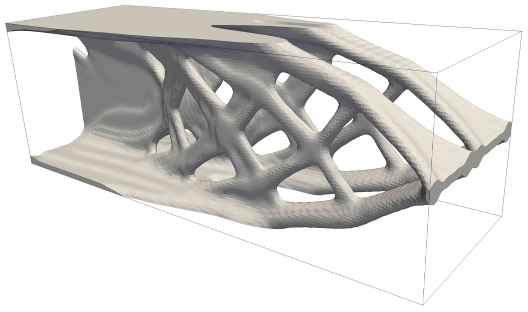
The resulting designs for the *CutFEM* and the *CutFEM cont* tests of the cantilever beam problem can be seen in Figure G.2 and Figure G.3. The resulting designs for the *CutFEM* and the *CutFEM cont* tests of the MBB problem can be seen in Figure G.4 and Figure G.5.

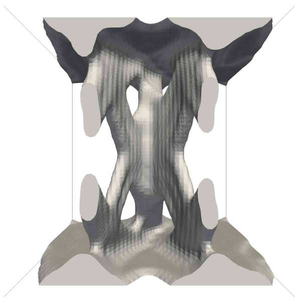

(a) Optimized structure, $\Delta\eta = 0.00$

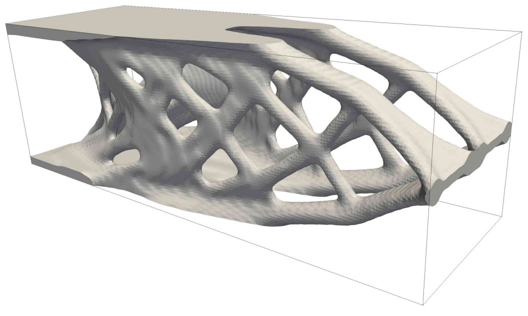

(b) Clip, $\Delta\eta = 0.00$

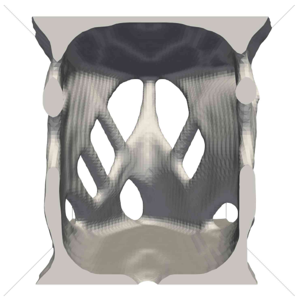

(c) Optimized structure, $\Delta\eta = 0.01$

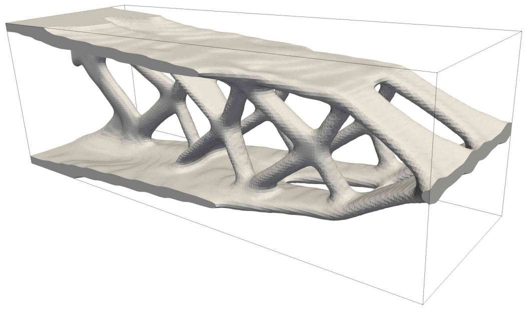

(d) Clip, $\Delta\eta = 0.01$


(e) Optimized structure, $\Delta\eta = 0.02$


(f) Clip, $\Delta\eta = 0.02$


(g) Optimized structure, $\Delta\eta = 0.03$


(h) Clip, $\Delta\eta = 0.03$


(i) Optimized structure, $\Delta\eta = 0.05$, iteration 279

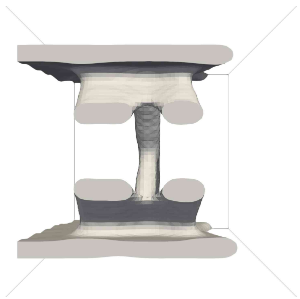
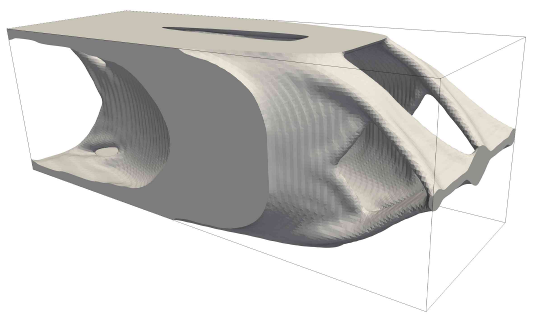
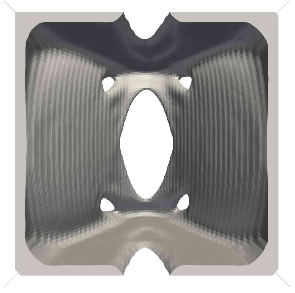
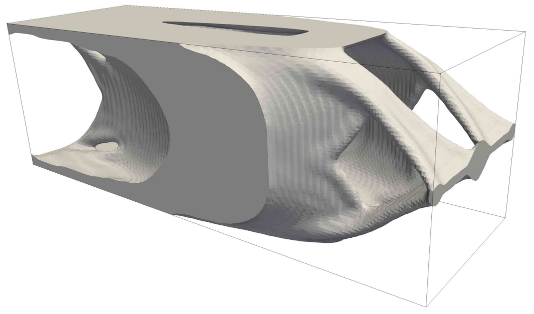
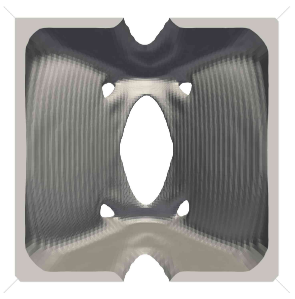

(j) Clip, $\Delta\eta = 0.05$, iteration 279

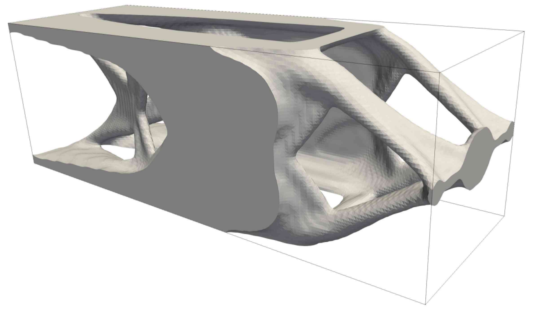
Figure G.2: Optimized robust structure for the *CutFEM* test of the cantilever beam problem for different threshold offset values, at iteration 1000. For input, see section 4.3

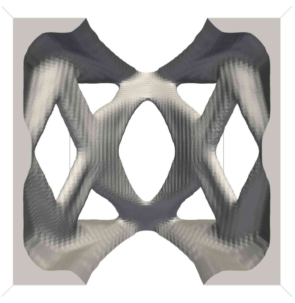

(a) Optimized structure, $\Delta\eta = 0.00$

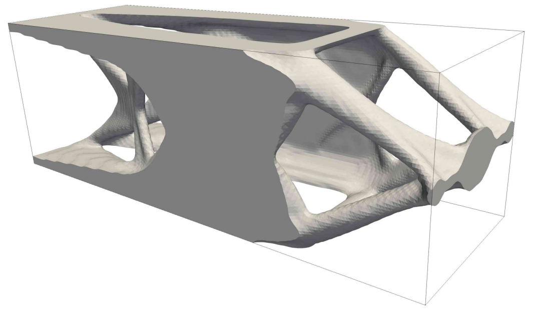

(b) Clip, $\Delta\eta = 0.00$

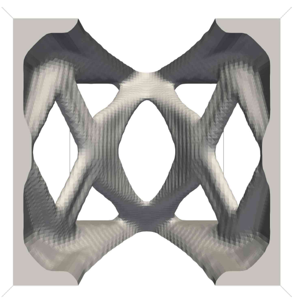

(c) Optimized structure, $\Delta\eta = 0.01$

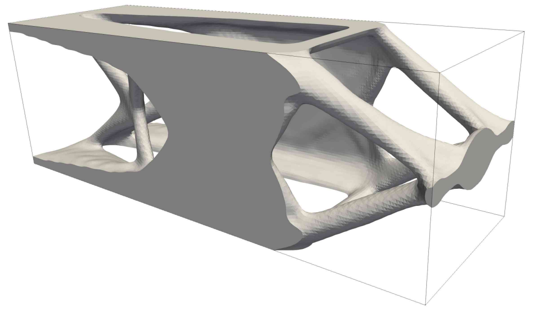

(d) Clip, $\Delta\eta = 0.01$


(e) Optimized structure, $\Delta\eta = 0.02$


(f) Clip, $\Delta\eta = 0.02$


(g) Optimized structure, $\Delta\eta = 0.03$


(h) Clip, $\Delta\eta = 0.03$


(i) Optimized structure, $\Delta\eta = 0.05$, iteration 64

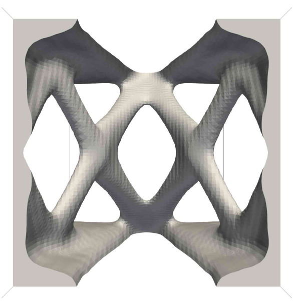
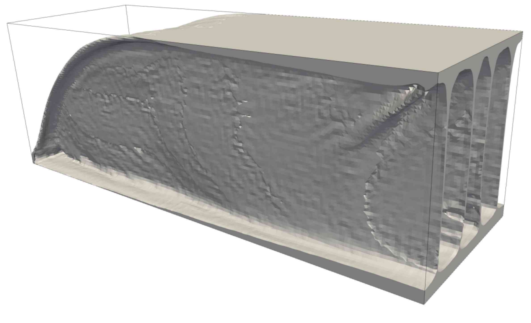
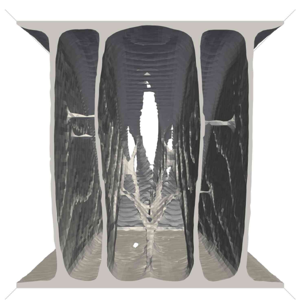
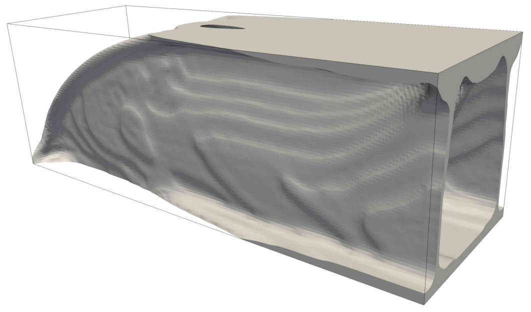
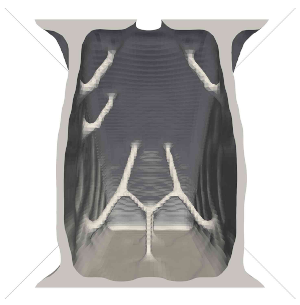

(j) Clip, $\Delta\eta = 0.05$, iteration 64

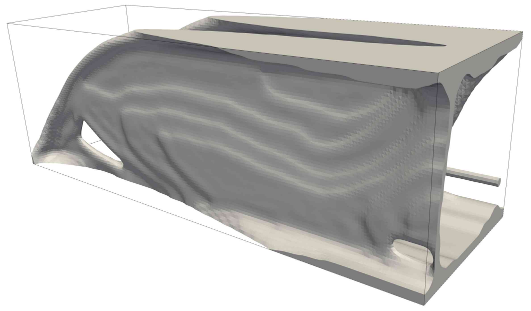
Figure G.3: Optimized robust structure for the *CutFEM cont* test of the cantilever beam problem for different threshold offset values, at iteration 1000. For input, see section 4.3

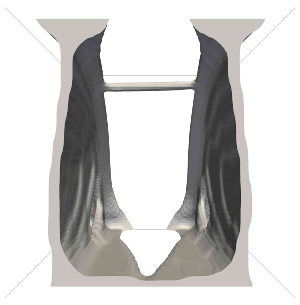

(a) Optimized structure, $\Delta\eta = 0.00$

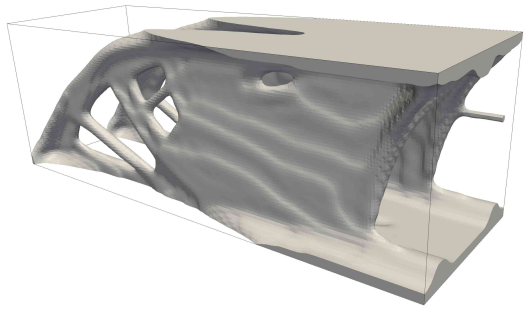

(b) Clip, $\Delta\eta = 0.00$

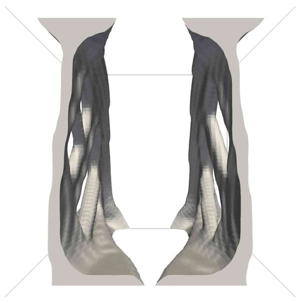

(c) Optimized structure, $\Delta\eta = 0.01$

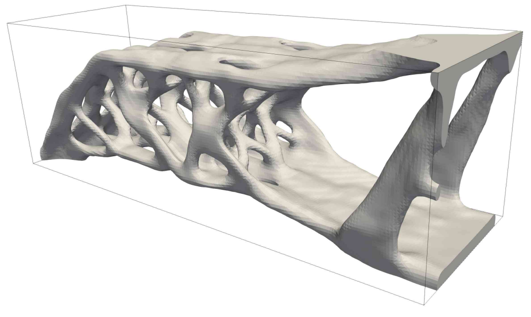

(d) Clip, $\Delta\eta = 0.01$


(e) Optimized structure, $\Delta\eta = 0.02$


(f) Clip, $\Delta\eta = 0.02$


(g) Optimized structure, $\Delta\eta = 0.03$


(h) Clip, $\Delta\eta = 0.03$


(i) Optimized structure, $\Delta\eta = 0.05$, iteration 172

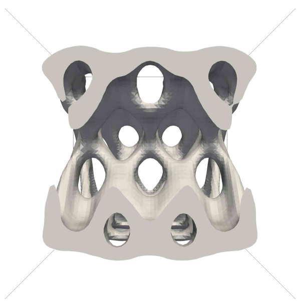
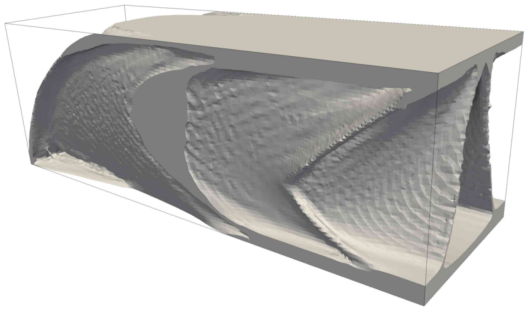
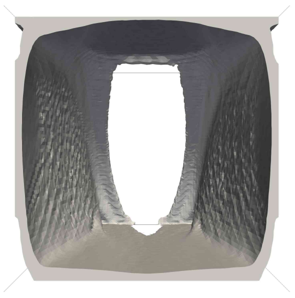
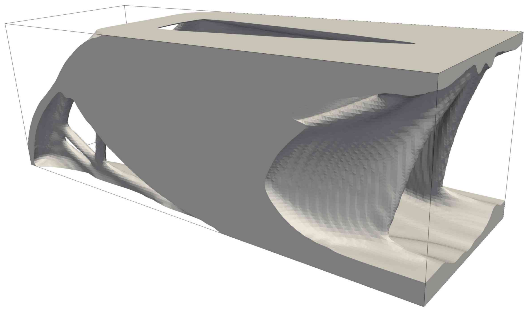
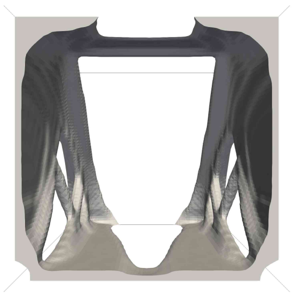

(j) Clip, $\Delta\eta = 0.05$, iteration 172

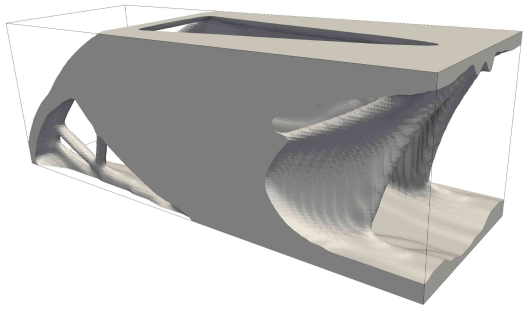
Figure G.4: Optimized robust structure for the *CutFEM* test of the MBB beam problem for different threshold offset values, at iteration 1000. For input, see section 4.3

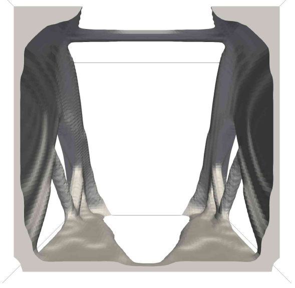

(a) Optimized structure, $\Delta\eta = 0.00$

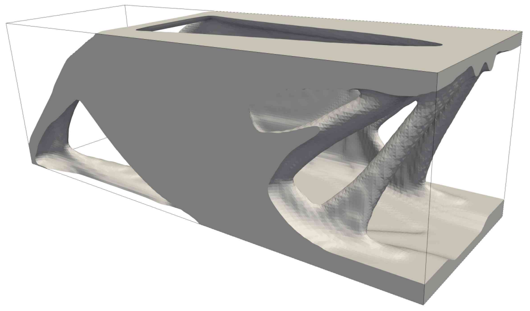

(b) Clip, $\Delta\eta = 0.00$

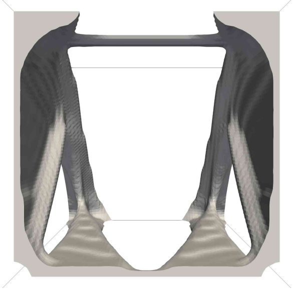

(c) Optimized structure, $\Delta\eta = 0.01$

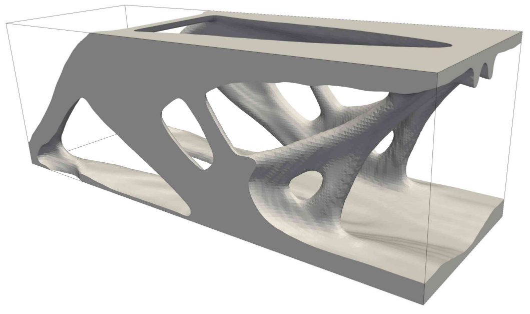

(d) Clip, $\Delta\eta = 0.01$


(e) Optimized structure, $\Delta\eta = 0.02$


(f) Clip, $\Delta\eta = 0.02$


(g) Optimized structure, $\Delta\eta = 0.03$


(h) Clip, $\Delta\eta = 0.03$


(i) Optimized structure, $\Delta\eta = 0.05$, iteration 109

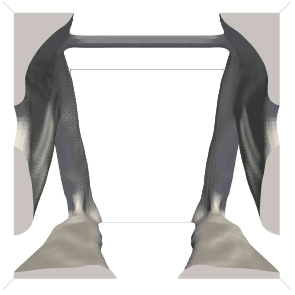

(j) Clip, $\Delta\eta = 0.05$, iteration 109

Figure G.5: Optimized robust structure for the *CutFEM cont* test of the MBB beam problem for different threshold offset values, at iteration 1000. For input, see section 4.3

Bibliography

[1] Niels Aage and Boyan S. Lazarov. Parallel framework for topology optimization using the method of moving asymptotes. *Structural and Multidisciplinary Optimization*, 47(4):493–505, 2013. ISSN 1615147X. doi: 10.1007/s00158-012-0869-2.

[2] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, 51(3):565–572, 2015. ISSN 16151488. doi: 10.1007/s00158-014-1157-0.

[3] Joe Alexandersen, Niels Aage, Casper Schousboe Andreasen, and Ole Sigmund. Topology optimisation for natural convection problems. *International Journal for Numerical Methods in Fluids*, 76(10):699–721, 12 2014. ISSN 02712091. doi: 10.1002/fld.3954. URL `http://doi.wiley.com/10.1002/fld.3954`.

[4] Grégoire Allaire, François Jouve, and Anca Maria Toader. A level-set method for shape optimization. *Comptes Rendus Mathematique*, 334(12):1125–1130, 2002. ISSN 1631073X. doi: 10.1016/S1631-073X(02)02412-3.

[5] Grégoire Allaire, François Jouve, and Anca Maria Toader. *Structural optimization using sensitivity analysis and a level-set method*, volume 194. 2004. ISBN 3316933301. doi: 10.1016/j.jcp.2003.09.032.

[6] G S Almasi and A Gottlieb. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1989. ISBN 0-8053-0177-1.

[7] Oded Amir, Niels Aage, and Boyan S. Lazarov. On multigrid-CG for efficient topology optimization. *Structural and Multidisciplinary Optimization*, 49(5):815–829, 2014. ISSN 16151488. doi: 10.1007/s00158-013-1015-5.

[8] Casper Schousboe Andreasen and Ole Sigmund. Topology optimization of fluid-structure-interaction problems in poroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 258:55–62, 2013. ISSN 00457825. doi: 10.1016/j.cma.2013.02.007. URL `http://dx.doi.org/10.1016/j.cma.2013.02.007`.

[9] Casper Schousboe Andreasen, Allan Roulund Gersborg, and Ole Sigmund. Topology optimization of microfluidic mixers. *International Journal for Numerical Methods in Fluids*, 61(5):498–513, 10 2009. ISSN 02712091. doi: 10.1002/fld.1964. URL `http://doi.wiley.com/10.1002/fld.1964`.

[10] E. Arquis and J. P. Caltagirone. Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide - milieu poreux: application à la convection naturelle. *C. R. Acad. Sci. Paris 299*, 1984.

[11] S. Balay, S. Abhyankar, and M. Adams. PETSc Users Manual, 2019.

[12] A A Becker. *Background to Finite Element Analysis of Geometric Non-linearity Benchmarks*. Ref: National Agency for Finite Element Methods and Standards. NAFEMS, 2000. URL `https://books.google.dk/books?id=5kndnQEACAAJ`.

[13] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, 45(5):601–620, 6 1999. ISSN 0029-5981. doi: 10.1002/(SICI)1097-0207(19990620)45:5<601::AID-NME598>3.0.CO;2-S. URL `http://doi.wiley.com/10.1002/%28SICI%291097-0207%2819990620%2945%3A5%3C601%3A%3AAID-NME598%3E3.0.CO%3B2-S`.

[14] M P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*, volume 2nd Editio. 2003. ISBN 3540429921. doi: 10.1063/1.3278595.

[15] Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224, 1988. ISSN 00457825. doi: 10.1016/0045-7825(88)90086-2.

[16] Martin Philip Bendsøe, Alejandro Díaz, and Noboru Kikuchi. Topology and Generalized Layout Optimization of Elastic Structures. In Martin Philip Bendsøe and Carlos A Mota Soares, editors, *Topology Design of Structures*, pages 159–205. Springer Netherlands, Dordrecht, 1993. ISBN 978-94-011-1804-0. doi: 10.1007/978-94-011-1804-0{_}13. URL `https://doi.org/10.1007/978-94-011-1804-0_13`.

[17] Friedrich Bessel. Untersuchungen über die Länge des einfachen Sekundenpendels. 1828.

[18] Paul T. Boggs and Jon W. Tolle. Sequential Quadratic Programming. *Acta Numerica*, 4(January):1, 1995. ISSN 0962-4929. doi: 10.1017/s0962492900002518.

[19] Thomas Borrvall and Joakim Petersson. Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids*, 41(1):77–107, 2003. ISSN 02712091. doi: 10.1002/fld.426.

[20] Blaise Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158, 2001. ISSN 00295981. doi: 10.1002/nme.116.

[21] Blaise Bourdin and Antonin Chambolle. Design-dependent loads in topology optimization. *ESAIM: Control, Optimisation and Calculus of Variations*, 9(January):19–48, 1 2003. ISSN 1292-8119. doi: 10.1051/cocv:2002070. URL `http://archive.numdam.org/article/COCV_2002__8__741_0.pdfhttp://www.esaim-cocv.org/10.1051/cocv:2002070`.

[22] Stephen Boyd and Lieven Vandenberghe. *Convex optimization theory*, volume 25. 2004. ISBN 9780521833783. doi: 10.1080/10556781003625177. URL `http://www.tandfonline.com/doi/abs/10.1080/10556781003625177`.

[23] Patrick Bridges, Nathan Doss, William Gropp, Edward Karrels, Ewing Lusk, and Anthony Skjellum. User's Guide to MPICH, a Portable Implementation of MPI. *Argonne National Laboratory*, 9700:60439–64801, 1995. ISSN 2198-6061. doi: 10.1007/978-90-481-2945-4.

[24] Tyler E. Bruns and Daniel A. Tortorelli. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190(26-27):3443–3459, 2001. ISSN 00457825. doi: 10.1016/S0045-7825(00)00278-4.

[25] Michael Bungartz, Hans-Joachim; Schäfer. *Fluid-structure Interaction: Modelling, Simulation, Optimization*. 2006.

[26] Erik Burman and Peter Hansbo. Fictitious domain finite element methods using cut elements II. *Applied Numerical Mathematics*, 62(4):328–341, 2012. ISSN 01689274. doi: 10.1016/j.apnum.2011.01.008. URL `http://dx.doi.org/10.1016/j.apnum.2011.01.008`.

[27] Erik Burman, Susanne Claus, Peter Hansbo, Mats G. Larson, and André Massing. CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 11 2015. ISSN 00295981. doi: 10.1002/nme.4823. URL `http://doi.wiley.com/10.1002/nme.4823`.

[28] Erik Burman, Daniel Elfverson, Peter Hansbo, Mats G. Larson, and Karl Larsson. Shape optimization using the cut finite element method. *Computer Methods in Applied Mechanics and Engineering*, 328:242–261, 2018. ISSN 00457825. doi: 10.1016/j.cma.2017.09.005.

[29] Evgeni V Chernyaev. Marching Cubes 33. *Elements*, pages 1–8, 1996.

[30] Peter W. Christensen and Anders Klarbring. *An Introduction to Structural Optimization*, volume 153 of *Solid Mechanics and Its Applications*. Springer Netherlands, Dordrecht, 2008. ISBN 978-1-4020-8665-6. doi: 10.1007/978-1-4020-8666-3. URL `http://link.springer.com/10.1007/978-1-4020-8666-3`.

[31] COMSOL. Large Deformation Analysis of a Beam, 2019.

[32] Robert D. Cook and H. Saunders. Concepts and Applications of Finite Element Analysis (2nd Edition). 12 1981. ISSN 00949930. doi: 10.1115/1.3264300. URL `http://pressurevesseltech.asmedigitalcollection.asme.org/article.aspx?articleid=1455188`.

[33] B Delaunay. Sur La Sphere Vide. *Bulletin de l'Académie des Sciences de l'URSS*, (6):793–800, 1934.

[34] A. Díaz and O. Sigmund. Checkerboard patterns in layout optimization. *Structural Optimization*, 10(1): 40–45, 1995. ISSN 09344373. doi: 10.1007/BF01743693.

[35] Earl H Dowell and Kenneth C Hall. Modeling of Fluid-Structure Interaction. *Annual Review of Fluid Mechanics*, 33(1):445–490, 1 2001. ISSN 0066-4189. doi: 10.1146/annurev.fluid.33.1.445. URL `http://www.annualreviews.org/doi/10.1146/annurev.earth.33.092203.122610http://www.annualreviews.org/doi/10.1146/annurev.fluid.33.1.445`.

[36] Maria B. Dühring, Jakob S. Jensen, and Ole Sigmund. Acoustic design by topology optimization. *Journal of Sound and Vibration*, 317(3-5):557–575, 2008. ISSN 0022460X. doi: 10.1016/j.jsv.2008.03.042.

[37] C. Farhat, M. Lesoinne, and P. LeTallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 157(1-2):95–114, 1998. ISSN 00457825. doi: 10.1016/S0045-7825(97)00216-8.

[38] R FLETCHER. A General Quadratic Programming Algorithm. *IMA Journal of Applied Mathematics*, 7 (1):76–91, 2 1971. ISSN 0272-4960. doi: 10.1093/imamat/7.1.76. URL `https://doi.org/10.1093/imamat/7.1.76`.

[39] Claude Fleury and Vincent Braibant. Structural optimization: A new dual method using mixed variables. *International Journal for Numerical Methods in Engineering*, 23(3):409–428, 3 1986. ISSN 0029-5981. doi: 10.1002/nme.1620230307. URL `https://doi.org/10.1002/nme.1620230307`.

[40] Thomas-Peter Fries and Ted Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, (February):n/a–n/a, 8 2010. ISSN 00295981. doi: 10.1002/nme.2914. URL `http://doi.wiley.com/10.1002/nme.2914`.

[41] Jean-Frédéric Gerbeau and Marina Vidrascu. A Quasi-Newton Algorithm Based on a Reduced Model for Fluid-Structure Interaction Problems in Blood Flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(4):631–647, 2003. ISSN 0764-583X. doi: 10.1051/m2an:2003049. URL `http://www.esaim-m2an.org/10.1051/m2an:2003049`.

[42] Jean-frédéric Gerbeau, Vidrascu Marina, and Pascal Frey. Fluid-Structure Interaction in Blood Flows on Geometries coming from Medical Imaging. *Simulation*, 2003.

[43] J. E. Gordon. Structures or Why things don't fall down, 1978.

[44] W Gropp and B. Smith. Users manual for the Chameleon parallel programming tools. *Math. and Comp. Science Div., Argonne Nat.*, 1993. URL `http://heim.ifi.uio.no/~kt-lab/Dok/chameleon/chameleon.ps.gz`.

[45] W Gropp, E Lusk, and A Skjellum. *Using MPI: Portable Parallel Programming with the Message-passing Interface*. Number v. 1 in Scientific and engineering computation. MIT Press, 1999. ISBN 9780262571326. URL `https://books.google.dk/books?id=xpBZ0RyRb-oC`.

[46] William Gropp. *Using MPI: Portable parallel programming with the message-passing interface*. 2014. ISBN 0262571323. doi: 10.1016/S0898-1221(00)90207-4.

[47] William Gropp and Nathan Doss. A high-performance, portable implementation of the MPI message passing interface standard. 1996.

[48] J. K. Guest, J. H. Prévost, and T. Belytschko. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International Journal for Numerical Methods in Engineering*, 61(2):238–254, 2004. ISSN 00295981. doi: 10.1002/nme.1064.

[49] Choi K.K. Komkov V. Haug, E.J. Design sensitivity analysis of structural systems. *Academic Press, New York*, (May), 1992.

[50] Nicholas Jenkins and Kurt Maute. Level set topology optimization of stationary fluid-structure interaction problems. *Structural and Multidisciplinary Optimization*, 52(1):179–195, 2015. ISSN 16151488. doi: 10.1007/s00158-015-1229-9.

[51] Jakob S. Jensen and O. Sigmund. Topology optimization for nano-photonics. *Laser and Photonics Reviews*, 5(2):308–321, 2011. ISSN 18638880. doi: 10.1002/lpor.201000014.

[52] C S Jog, R B Haber, and M P Bendsøe. A Displacement-Based Topology Design Method with Self-Adaptive Layered Materials. In Martin Philip Bendsøe and Carlos A Mota Soares, editors, *Topology Design of Structures*, pages 219–238. Springer Netherlands, Dordrecht, 1993. ISBN 978-94-011-1804-0. doi: 10.1007/978-94-011-1804-0{_}15. URL `https://doi.org/10.1007/978-94-011-1804-0_15`.

[53] Chandrashekhar S. Jog and Robert B. Haber. Computer methods in applied mechanics and engineering Stability of finite element models for distributed-parameter optimization and topology design. *Comput. Methods Appl. Mech. Engrg*, 130(95):203–226, 1996. ISSN 00457825. doi: 10.1016/0045-7825(95)00928-0.

[54] Benjamin Kadoch, Dmitry Kolomenskiy, Philippe Angot, and Kai Schneider. A volume penalization method for incompressible flows and scalar advection-diffusion with moving obstacles. *Journal of Computational Physics*, 231(12):4365–4383, 2012. ISSN 10902716. doi: 10.1016/j.jcp.2012.01.036. URL `http://dx.doi.org/10.1016/j.jcp.2012.01.036`.

[55] Nicholas K.R. Kevlahan and Jean Michel Ghidaglia. Computation of turbulent flow past an array of cylinders using a spectral method with Brinkman penalization. *European Journal of Mechanics, B/Fluids*, 20 (3):333–350, 2001. ISSN 09977546. doi: 10.1016/S0997-7546(00)01121-3.

[56] Amir Kolaei, Subhash Rakheja, and Marc J. Richard. An efficient methodology for simulating roll dynamics of a tank vehicle coupled with transient fluid slosh. *JVC/Journal of Vibration and Control*, 23(19): 3216–3232, 2017. ISSN 17412986. doi: 10.1177/1077546315627565.

[57] Markus Kollmann. Sensitivity Analysis: The Direct and Adjoint Method. *Environmental Pollution*, page 75, 2010. ISSN 02697491. doi: 10.1016/j.envpol.2016.03.009. URL `https://mathematik.jku.at/Teaching/Diplom/Finished/kollmann-dipl.pdf%0Ahttps://linkinghub.elsevier.com/retrieve/pii/S0269749116301907`.

[58] Qianlong Liu and Oleg V. Vasilyev. A Brinkman penalization method for compressible flows in complex geometries. *Journal of Computational Physics*, 227(2):946–966, 2007. ISSN 10902716. doi: 10.1016/j.jcp.2007.07.037.

[59] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, volume 5, pages 163–169, New York, New York, USA, 1987. ACM Press. ISBN 0897912276. doi: 10.1145/37401.37422. URL `http://portal.acm.org/citation.cfm?doid=37401.37422`.

[60] David Makhija and Kurt Maute. Numerical instabilities in level set topology optimization with the extended finite element method. *Structural and Multidisciplinary Optimization*, 49(2):185–197, 2014. ISSN 1615147X. doi: 10.1007/s00158-013-0982-x.

[61] A. Massing, B. Schott, and W. A. Wall. A stabilized Nitsche cut finite element method for the Oseen problem. *Computer Methods in Applied Mechanics and Engineering*, 328:262–300, 2018. ISSN 00457825. doi: 10.1016/j.cma.2017.09.003. URL `http://dx.doi.org/10.1016/j.cma.2017.09.003`.

[62] James Clerk Maxwell. The Scientific Papers of James Clerk Maxwell. *The Scientific Papers of James Clerk Maxwell*, 1869. doi: 10.1017/cbo9780511710377.

[63] J.M. Melenk and I. Babuška. The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):289–314, 12 1996. ISSN 00457825. doi: 10.1016/S0045-7825(96)01087-0. URL `http://linkinghub.elsevier.com/retrieve/pii/S0045782596010870`.

[64] A. G. M. Michell and M. C. E. Melbourne. LVIII . The Limits of Economy of lllaterial in Frame-structures . *Philosophical Magazine Series 6*, 8(47):589–597, 1904. doi: 10.1080/14786440409463229. URL `http://www.tandfonline.com/doi/abs/10.1080/14786440409463229`.

[65] Gary L Miller, Dafna Talmor, Shang-Hua Teng, Noel Walkington, and Han Wang. Control volume meshes using sphere packing. M(December):128–131, 1999. doi: 10.1007/bfb0018533.

[66] H. P. Mlejnek. Some aspects of the genesis of structures. *Structural Optimization*, 5(1-2):64–69, 1992. ISSN 09344373. doi: 10.1007/BF01744697.

[67] J. Nitsche. Uber ein Variationsprinzip zur Losung von Dirichlet-Problemen bei Verwendung von Teilraumen, die keinen Randbedingungen unterworfen sind. *Abh. Math. Sem. Univ. Hamburg*, 1971.

[68] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization Second Edition*, volume 83. 2006. ISBN SBN-10: 0-387-30303-0.

[69] William K. Pratt. *Digital image processing(Book)*, volume 5. John Wiley & Sons, Inc., 1977. ISBN 0-471-37407-5.

[70] H C Rodrigues and P A Fernandes. Topology Optimization of Linear Elastic Structures Subjected to Thermal Loads. In Martin Philip Bendsøe and Carlos A Mota Soares, editors, *Topology Design of Structures*, pages 437–450. Springer Netherlands, Dordrecht, 1993. ISBN 978-94-011-1804-0. doi: 10.1007/978-94-011-1804-0{_}31. URL `https://doi.org/10.1007/978-94-011-1804-0_31`.

[71] Wen Bin Shangguan and Zhen Hua Lu. Experimental study and simulation of a hydraulic engine mount with fully coupled fluid-structure interaction finite element analysis model. *Computers and Structures*, 82(22):1751–1771, 2004. ISSN 00457949. doi: 10.1016/j.compstruc.2004.05.017.

[72] Ashesh Sharma, Hernan Villanueva, and Kurt Maute. On shape sensitivities with heaviside-enriched XFEM. *Structural and Multidisciplinary Optimization*, 55(2):385–408, 2017. ISSN 16151488. doi: 10.1007/s00158-016-1640-x.

[73] H. Si. TetGen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator. *UR L http://tetgen.berlios. de*, (13):vii + 97, 2013. URL `http://wias-berlin.de/software/tetgen/1.5/doc/manual/manual.pdf`.

[74] Hang Si. TetGen, a Quality Tetrahedral Mesh Generator. *AMC Transactions on Mathematical Software*, 41(2):11, 2015. ISSN 00983500. doi: 10.1007/3-540-29090-7{_}9.

[75] O. Sigmund and J. Petersson. Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural Optimization*, 16(1):68–75, 1998. ISSN 09344373. doi: 10.1007/BF01214002.

[76] Ole Sigmund. Design of Thermal Structures using Topology Optimization. (August):232, 1994.

[77] Ole Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424, 2007. ISSN 1615147X. doi: 10.1007/s00158-006-0087-x.

[78] Ole Sigmund and Kurt Maute. Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, 2013. ISSN 1615147X. doi: 10.1007/s00158-013-0978-6.

[79] WS Slaughter and J Petrolito. Linearized Theory of Elasticity. *Applied Mechanics Reviews*, 55(5):B90, 2002. ISSN 00036900. doi: 10.1115/1.1497478. URL `http://appliedmechanicsreviews.asmedigitalcollection.asme.org/article.aspx?articleid=1397323`.

[80] J. Sokolowski and A. Zochowski. The topological derivative method in shape optimization. *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, 1(4):1251–1272, 1999. ISSN 0191-2216. doi: 10.1109/CDC.1999.832864.

[81] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 2 1987. ISSN 0029-5981. doi: 10.1002/nme.1620240207. URL `https://doi.org/10.1002/nme.1620240207`.

[82] Tayfun E. Tezduyar, Sunil Sathe, Matthew Schwaab, and Brian S. Conklin. Arterial fluid mechanics modeling with the stabilized space–time fluid–structure interaction technique. *International Journal for Numerical Methods in Fluids*, 57(5):601–629, 6 2008. ISSN 02712091. doi: 10.1002/fld.1633. URL `http://doi.wiley.com/10.1002/fld.1633`.

[83] S. Timoshenko and J.N. Goodier. Theory of Elasticity. *Journal of the Royal Aeronautical Society*, 4 1951.

[84] N. P. Van Dijk, K. Maute, M. Langelaar, and F. Van Keulen. Level-set methods for structural topology optimization: A review. *Structural and Multidisciplinary Optimization*, 48(3):437–472, 2013. ISSN 1615147X. doi: 10.1007/s00158-013-0912-y.

[85] Carlos H. Villanueva and Kurt Maute. CutFEM topology optimization of 3D laminar incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 320:444–473, 2017. ISSN 00457825. doi: 10.1016/j.cma.2017.03.007. URL `http://dx.doi.org/10.1016/j.cma.2017.03.007`.

[86] Fengwen Wang, Boyan Stefanov Lazarov, and Ole Sigmund. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6): 767–784, 2011. ISSN 1615147X. doi: 10.1007/s00158-010-0602-y.

[87] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. HR 1995 Condition facilitiating interorganizational collaboration.pdf. 192:227–246, 2003.

[88] G.H. Weber, Gerik Scheuermann, Hans Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. volume 2, pages 171–178, 2003. ISBN 0780374983. doi: 10.1109/visual.2002.1183772.

[89] R. Wenger. Chapter 2 Marching Cubes. *Isosurfaces - geometry, topology, and algorithms*, pages 17–44, 2013.

[90] Robert B. Wilson. A simplicial method for convex programming, 1963.

[91] Shoufei Wu and Zonghuai Wang. A Numerical Simulation of Fluid-Structure Interaction for Refrigerator Compressors Suction and Exhaust System Performance Analysis. *International Compressor Engineering Conference*, pages 1–7, 2014.

[92] Shengli Xu, Yuanwu Cai, and Gengdong Cheng. Volume preserving nonlinear density filter based on heaviside functions. *Structural and Multidisciplinary Optimization*, 41(4):495–505, 2010. ISSN 1615147X. doi: 10.1007/s00158-009-0452-7.

[93] Gil H. Yoon and Ole Sigmund. A monolithic approach for topology optimization of electrostatically actuated devices. *Computer Methods in Applied Mechanics and Engineering*, 197(45-48):4062–4075, 2008. ISSN 00457825. doi: 10.1016/j.cma.2008.04.004.

[94] M. Zhou and G.I.N. Rozvany. The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1-3):309–336, 1991. ISSN 00457825. doi: 10.1016/0045-7825(91)90046-9. URL `http://linkinghub.elsevier.com/retrieve/pii/0045782591900469`.