Department of Precision and Microsystems Engineering

Prevention of enclosed voids in topology optimization

Joran van der Zwet

Report no : 2021.038
Coach : Arnoud Delissen
Professor : Matthijs Langelaar
Specialisation : Engineering Mechanics
Type of report : Master Thesis
Date : 28-06-2021

TU Delft Delft University of Technology

Challenge the future

Abstract

Topology optimization has seen increased interest with the development of additive manufacturing (AM) as a manufacturing method, because of its ability to utilize the geometric complexity that AM offers. However AM still imposes some restrictions on the design, most notably on its minimum feature size, overhang, and enclosed voids. Enclosed voids are problematic because for most AM methods it is impossible to remove supports from them, additionally for powder-based AM these enclosed voids trap unmelted powder adding parasitic mass. In this thesis, two new methods have been investigated to alleviate this issue, the eigenfrequency method and the flood fill method. The eigenfrequency method utilizes the eigenfrequency of an inverted density field, which changes enclosed voids to floating masses. These floating masses have rigid body modes, which can then be prevented by applying a minimum eigenfrequency constraint. The eigenfrequency method encountered a large problem in the form of intermediate densities, which were used to satisfy the eigenfrequency constraint instead of the elimination of enclosed voids. The flood fill method utilizes a modified flood fill algorithm, which is a filter with a resulting density field where every enclosed void element is changed to solid. The flood fill method did successfully eliminate enclosed voids in both 2D and 3D problems, at the cost of very little additional computational effort. It also allows for direct control over the location, amount, and size of the void elimination features by varying boundary elements, running additional flood fills, and morphologic operators, respectively.

Acknowledgments

Ever since I started my studies at the TU Delft I always felt that, while I really enjoyed the course work, I hadn't really found my main area of interest within mechanical engineering. For this reason I also chose the broadest of the master specializations, high tech engineering. This turned out to be a great choice as within the introductory week I found myself enamored with these incredibly strong but also natural looking shapes, which turned out to be created by topology optimization. The introductory week also held a 'beat the optimizer' challenge and while my team failed miserably, it still cemented this interest for the coming years. When the time came to choose an assignment for the master thesis, I immediately looked for assignments to do with topology optimization and a year later I am still happy with that decision.

Unfortunately this thesis had to be done remotely, due to an ongoing pandemic. Luckily this topic does lend itself well to be done remotely, although it feels weird that the first time I will see my supervisors in person will be at my defence and graduation. On the topic of supervisors I would like to thank Arnoud Delissen for his coaching in the weekly meetings, when the internet would allow it at least. The meetings were always helpful in knowing what next steps to take and meeting weekly also helped to keep me focused throughout the week so that I would have something to show or ask. Furthermore I would like to thank Matthijs Langelaar for his insight in the more sporadic meetings we had, as I always left those meetings with a large list of possible additions, ideas or changes to the methods. I would finally like to thank both supervisors for the large help in writing this report in the later stages of the thesis, as their feedback was both very comprehensive and given to me within a few days of handing in a draft.

I would finally like to thank my girlfriend, family and friends for being a welcome distraction in this past year.

Contents

1 Introduction

1.1 Background

Before presenting the problem statement, this section briefly discusses the two main background topics for this thesis, namely topology optimization and additive manufacturing (AM). Topology optimization has seen increased interest with the development of AM as a manufacturing method, because of its ability to utilize the geometric complexity that AM offers [1].

1.1.1 Topology Optimization

Topology optimization is a computational process aimed at finding the optimal material distribution for a given structural optimization problem, the considered optimization problems can be denoted using the following general form:

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in\Omega} & f(\mathbf{x}) & \\ \text{s.t.} & g_i(\mathbf{x}) \leq 0 & i = 1,\ldots,m \\ & h_j(\mathbf{x}) = 0 & j = 1,\ldots,p \\ & x_{lower} \leq \mathbf{x} \leq x_{upper} & \end{array} . \tag{1.1}$$

Here the objective function $f(x)$ denotes the main property to be optimized, e.g. minimizing compliance, minimizing or maximizing the first eigenfrequency or maximizing surface area. This optimization is then subject to different constraints to make sure the final result of the optimized structure meets certain conditions, e.g. a maximum mass or a maximum stress. These constraints are subdivided into two groups: g_i denotes inequality constraints and h_j denotes equality constraints. Finally the optimization is performed using the design variables $\mathbf{x}$, which can represent the material used, the cross-section of a beam or in the case of topology optimization: the material distribution. Most common optimization methods use sensitivity information to determine how changing the design variable affects the objective or constraints, which is then used to perform the next optimization iteration. Topology optimization generally uses the finite element method to calculate the current performance of the design and necessary sensitivities.

Topology optimization can be implemented with different methods, the most common of which are listed below [2]:

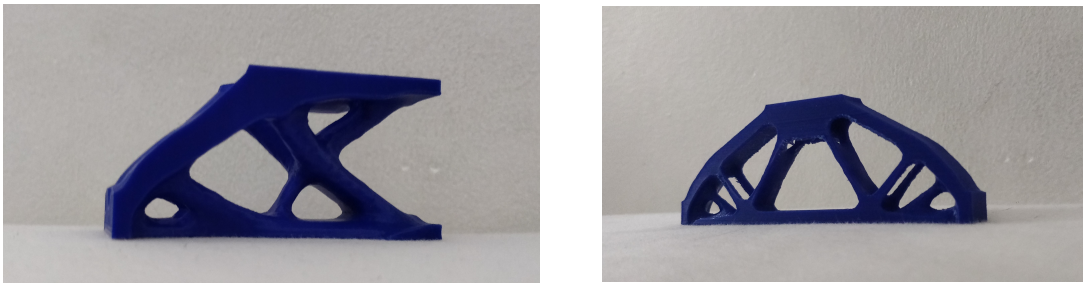
- The solid isotropic material with penalization (SIMP) method is the most commonly used method. It adds a density variable to each element, optimization is then done by changing these density variables to achieve an optimized design. The density variables are continuous over a prespecified range, which makes it possible to determine gradients that indicate the influence changing the density has on the objective or constraints. It suppresses intermediate densities by adding penalization to them and with sufficient penalization the optimizer then gets forced towards a solution with strictly void and solid elements, resulting in an interpretable design.
- The level set method uses a function depending on multiple variables to define the boundaries of the part. It does this by considering any region where the function is larger than zero as solid and smaller than zero as void.
- The evolutionary structural optimization (ESO) method is based on considering any material that is below a certain sensitivity threshold as redundant and removing it.
- The bi-directional evolutionary structural optimization (BESO) method is an extension of the ESO method in that in addition to removing material it can also add material where necessary.

While topology optimization on its own has complete freedom in possible resulting geometries, the final design still needs to be manufacturable. Therefore additional constraints or filters are necessary to comply with the restrictions of the chosen manufacturing method.

1.1.2 Additive Manufacturing

The main subject of this report is on topology optimization in combination with AM methods, as topology optimization has seen increased interest with the development of AM as a manufacturing method, because of its ability to utilize the geometric complexity that AM offers [1]. Multiple AM techniques have been developed, of which the main focus in this thesis is on powder bed fusion. Powder bed fusion follows the basic process [3] of first depositing a new layer of powder, then guiding a laser beam to scan the desired shape of the layer. This process is repeated until the complete part is finished.

Due to the layer-by-layer approach of AM it has a large advantage over conventional manufacturing methods in the ability to manufacture complex structures, for example some of these complex structures are shown in Figure 1.1. Additionally, compared to subtractive manufacturing methods like milling and turning an advantage can be the reduced material waste, as subtractive methods take material away from a larger block to manufacture the part. Forming manufacturing methods like casting and injection molding also have little material waste, but here AM has a different advantage in terms of a lower lead-time. For every new design a forming method needs a new mould, while AM could immediately start producing the new design.



(a) **(b)**

Figure 1.1: Two examples of complex optimized structures manufactured by 3D printing.

Despite these advantages AM is still not widely applied as a manufacturing method, although its usage is growing [4]. It is currently mostly used as a way to quickly manufacture prototypes due to its size restrictions, relatively slow production speed and high cost [4]. However this does not mean that AM is only being used for this purpose, e.g. Liu et al. [5] found that in the aerospace industry AM is already being used to manufacture parts with complex geometry, expensive materials, small production runs or parts that need quick turnaround time. Additionally, AM can also be used to aid conventional manufacturing methods, e.g. Hawaldar and Zhang [6] found that using AM to fabricate a sand casting mould yielded advantages in terms of weight and surface quality over conventional methods.

Even though AM has the ability to make very complex shapes that conventional methods are unable to make, it still places some restrictions on the design. The main restrictions are minimum feature size, maximum overhang angle and the presence of enclosed voids [7].

The restriction on minimum feature size comes from the fact that material is either deposited by an extruder, or in the case of powder based methods the fact that powder is melted by a laser. The extruder and laser have a finite diameter, therefore features made by them can not be smaller than this diameter.

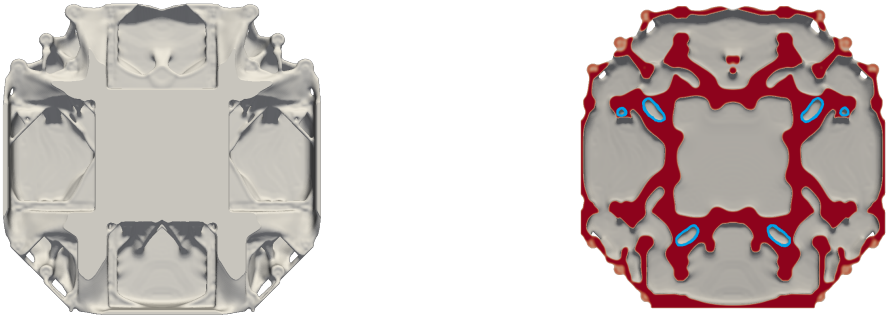
The restriction on overhang is caused by the fact that AM works on a layer by layer basis. Any new layer needs to be sufficiently supported by the previous layers, otherwise the new structure could collapse or sag. Generally an angle of 45 degrees is considered as the minimum overhang angle [8], but this angle can both be larger or smaller depending on different manufacturing conditions.

The last restriction is on enclosed voids, because for most AM methods the main issue with these is that it is impossible to remove supports from them, additionally for powder-based AM there is another problem in the form of trapped unmelted powder adding unnecessary mass. For example, Reddy et al. [9] found that in a part with a volume of 175 cm^3, more than 82 cm^3 of loose powder was trapped in the part and between supports. This is currently remedied either by manually removing the voids in post-processing once the optimization is finished or by drilling holes in the manufactured part, which adds unnecessary compliance and either more design or more manufacturing time.

1.2 Problem statement

1.2.1 Motivation and Aim

As seen in the previous section, AM has the ability to manufacture complex geometries designed by topology optimization, however the discussed restrictions still apply. Enclosed voids are sometimes present in the optimal shape for a given problem and can thus be generated by topology optimization, an example of such a design is shown in Figure 1.2. Therefore the aim of this research is to develop a method which can successfully eliminate enclosed voids during the topology optimization process. This method should either fill up any enclosed void or create a pathway to connect it to the boundary, while adding little compliance and computational effort. Additionally, the method should allow for sufficient control over the size of these pathways.



(a) **(b)**

Figure 1.2: Topology optimized design with enclosed voids, images provided by Arnoud Delissen. (a) Bottom view of design; (b) Cut view of design with enclosed voids outlined in blue.

1.2.2 Scope

Topology optimization can be implemented with a wide range of choices for different optimization approaches, i.e. which optimization algorithm to use or which type of physical problem to solve. In this research the focus lies solely with density-based topology optimization, which is then used to solve compliance minimization problems. The optimization algorithm used for this is the Method of Moving Asymptotes (MMA) [10]. This combination of the SIMP method with MMA is the most common topology optimization approach, which makes it a good choice to investigate the new enclosed void prevention methods.

1.2.3 Approach

Two new methods are considered for this thesis: The first method is based on the fact that rigid body modes, which occur when a body is not connected to a base, correspond to an eigenfrequency of zero. This phenomenon could then be utilized when applied to an inverted density field, where this low eigenfrequency indicates the presence of an enclosed void. The second method utilizes a flood fill algorithm to fill up any enclosed voids present in the design every iteration. The only way these voids can then exist in the design is if the optimizer makes a path of void elements towards it.

1.3 Outline

As mentioned in Section 1.2.1 and 1.2.3 the main focus of this research is the investigation of two new methods to eliminate enclosed voids during or after the topology optimization process. This thesis is divided into chapters: Chapter 2 discusses the current state of the art when it comes to the elimination of enclosed voids for topology optimization. Chapter 3 discusses the working principle, parameters to choose and results of the eigenfrequency method and Chapter 4 discusses these topics for the flood-fill method. Chapter 5 evaluates specific aspects of the two methods in detail. Chapter 6 contains the conclusions and recommendations for further research.

2 State of the art

2.1 Casting and Machining Methods

2.1.1 Casting constraints

It is impossible to produce a part designed with enclosed voids using a mold, therefore multiple studies have been done to prevent enclosed voids from forming in cast parts. Wang and Kang [11] presented a level set based method for preventing enclosed voids by adding a constraint to the optimization called the moldability condition. The moldability condition states that the normal of every point on the surface of the structure cannot deviate from the predefined parting direction of the molds by more than 90 degrees. Figure 2.1 shows the moldability condition applied to a cast part, where any surface that does not comply with the condition is coloured red. It is impossible for an enclosed void to comply with the moldability condition and therefore implementing it will succesfully prevent enclosed voids from forming. Xia et al. [12] use a similar approach to Wang and Kang, based on the level set method again but with a small change in formulation. They instead constrain every point on the boundary to only be able to move in the parting direction, this makes undercuts and enclosed voids impossible because these features require at least one part of the boundary to move in a different direction.

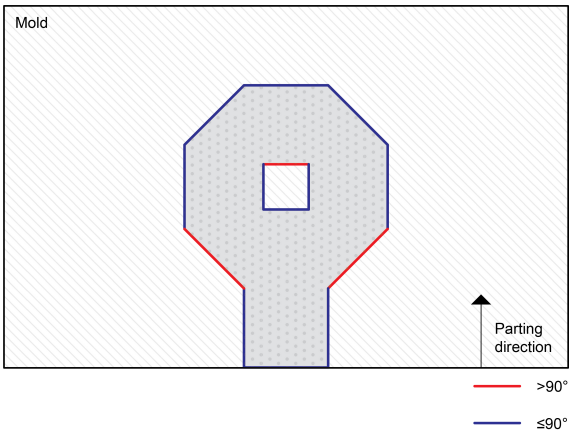


Figure 2.1: The moldability condition applied to an example part, image based on work by Wang and Kang [11].

Lu and Chen [13] proposed a SIMP-based method to ensure the optimization does not produce enclosed voids. It does this by forcing the optimizer to prefer material close to the mold interface by gradually reducing the maximum density of elements based on their distance from this interface. In combination with the SIMP method this essentially penalizes any material that is further away from the mold interface, which forces the optimizer to fill up an enclosed void rather than add material to the outside.

In addition to preventing enclosed voids, these casting methods also prevent undercuts from forming. This is a necessity for casting as having undercuts makes removal of the molds impossible, or would necessitate the use of complex inserts. However when AM is concerned this feature becomes a major drawback, because it is too restrictive.

2.1.2 Machining constraints

Next to casting other conventional manufacturing methods are the machining methods, milling and turning. It is also impossible to produce enclosed voids with machining methods, as any void needs to be reachable by the machining tool for the part to be manufacturable. For this reason multiple methods to ensure topology optimized part is manufacturable with machining have been proposed.

Langelaar [14] proposed a density-based filter to convert any input design into a manufacturable geometry. This filter has as main working mechanism that densities are cumulatively summed up in the insertion direction of the machining tool. Meaning that a void element which is blocked by any number of solid elements will automatically have a density of one or more in the filter. Additional steps of the filter can also be utilized to account for tool length and tool shape. This process is repeated for all possible tool insertion directions, where the geometry is rotated to accommodate the new insertion direction using a affine transformation to obtain the mapped density field. The filter finally combines all directions by first reverse mapping the density fields to the original mesh, then only taking the intersection of remaining solid elements to find the geometry of the final part.

Vatanabe et al. [15] proposed a projection based method that maps the design variables domain to a pseudo density domain by using a max function in the insertion direction. This mapping achieves the same result as the filter proposed by Langelaar, where any void element in the design variables domain that is blocked by any number of solid elements will have a density of one.

Liu and Ma [16] use a feature-based approach with the level set method. They achieve a machinable design by making the optimization about fitting features known to be machinable to the design. The method uses a virtual velocity field derived from the sensitivity analysis, after which the least squares method is used to minimize the difference between the feature velocity field and the virtual velocity field.

While the manufacturable geometries with machining are more complex than casting, there are still significant restrictions on complexity that AM does not have. This is still a large drawback when considering the complex geometries that are possible with topology optimization and AM.

2.2 Virtual temperature Method

An AM oriented method for detecting enclosed voids designed for topology optimization is the virtual temperature method proposed by Liu et al. [17]. In this method any voids are replaced with a virtual heat source of high conductivity and the solids replaced with virtual thermally insulating material. In addition to this the external boundaries of the design space are defined as heat dissipation boundaries with a temperature of 0. The heat generated in any enclosed void then cannot be transferred to the dissipation boundary effectively, which results in a high local temperature. However, if a void is connected to the external boundary the heat will easily be transferred to the dissipation boundary and therefore the local temperature remains low. Enclosed voids can then be detected by looking at which elements of the structure have the highest temperature. A visual representation of this method is shown in Figure 2.2. The

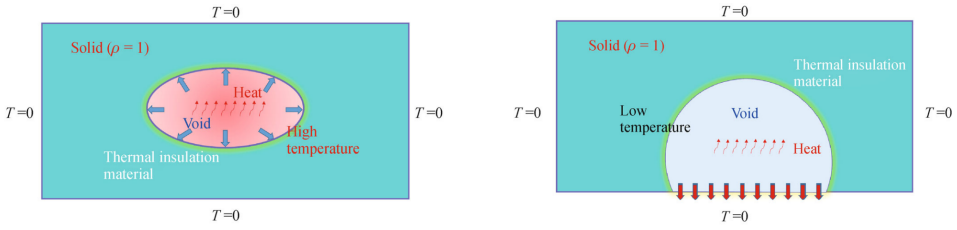


Figure 2.2: Visual representation of the virtual temperature method. (a) Structure with an enclosed void; (b) structure without enclosed voids. Images taken from [17].

virtual temperature method has also been implemented as a void prevention technique by utilizing its characteristic that any enclosed void will have a high maximum temperature. This is done by adding a constraint for the maximum temperature during the optimization process. The constraint temperature is found by first finding the maximum steady state temperature if all elements were void elements, then this maximum temperature is multiplied with a variable chosen by the designer to get the constraint temperature [18].

The advantages of this method are that calculating the temperature field does not require a large computational effort and that the possible geometric complexity of designs is unaltered. However the method also has its drawbacks, the first drawback is that the maximum temperature constraint has to be chosen by the designer and its influence is difficult to determine by hand. Different chosen values for the maximum temperature constraint result in large differences in the final design, this phenomenon is visible in Figure 2.3 for a beam torsion problem. The second drawback of this method is its tendency to make a void elimination channel as close to the void elements as possible, this is also the reason for the unnecessary amount of void elimination channels seen in Figure 2.3b. Yunfeng et al. [19] proposed a slight alteration to the method where the heat source is temperature-dependent, which is then used to make the local temperature of enclosed voids as uniform as possible. This alteration mitigates the issue of the first drawback of the method by making the local temperature more predictable, although whether this temperature could then be used to directly control void elimination features is still unknown.

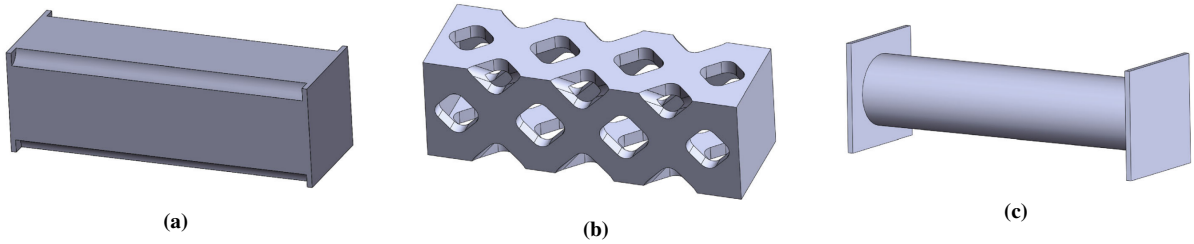


Figure 2.3: Different solutions to a torsion beam problem depending on different temperature constraints [17]. (a) No constraint, enclosed void in middle; (b) constraint is 10 times maximum steady state temperature and (c) constraint is 6 times maximum steady state temperature, solid in middle.

2.3 Adjacent Element Method

Another method for detecting enclosed voids is the neighboring element method presented by Xiong et al [20]. This method works by defining all neighbouring elements with the same density as either a void or solid set, then any enclosed void set can be detected by determining whether or not the set is connected to the boundary. For determining which neighboring elements should be added to a set the method utilizes a modified floodfill algorithm, where any element without a set is compared with its neighboring elements one by one until one with a matching density is found. The decision tree for a single element is shown in Figure 2.4.

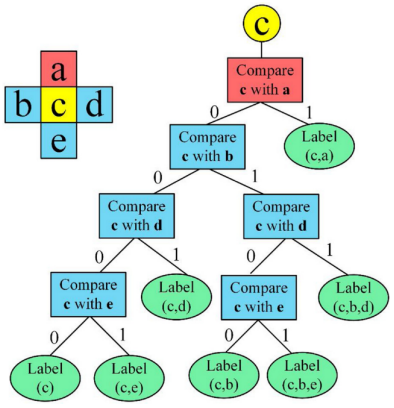
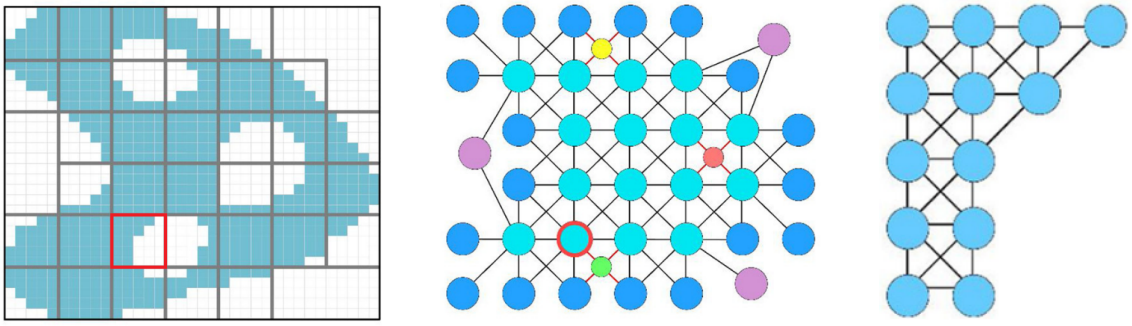


Figure 2.4: Decision tree for set determination in the neighbouring element method[20].

Once the enclosed voids are identified a path can be generated for powder removal. To generate the shortest powder removal path Xiong et al. [20] first convert the element based model into graphs in order to utilize existing shortest path algorithms. However performing this conversion for large amounts of elements requires a large computational effort, therefore they also propose to use a hierarchical graph scheme to do the computation. The hierarchical graph scheme works by having sets of elements at the upper levels and individual elements only at the lowest level, which is shown for two levels in Figure 2.5. Afterwards Dijkstra's shortest path algorithm is used for every graph level to find

the shortest path from an enclosed void to the outside. The resulting 2D structure is shown in Figure 2.6, additionally this method applied to a 3D platform problem is shown in Figure 2.7.



(a) (b) (c)

Figure 2.5: Example of the hierarchical graph scheme. (a) Division of element sets, (b) top level graph of element sets with the green, yellow, red and purple dots as void regions and (c) lowest level graph of element set outlined in red.

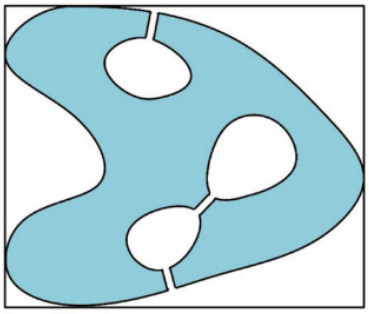
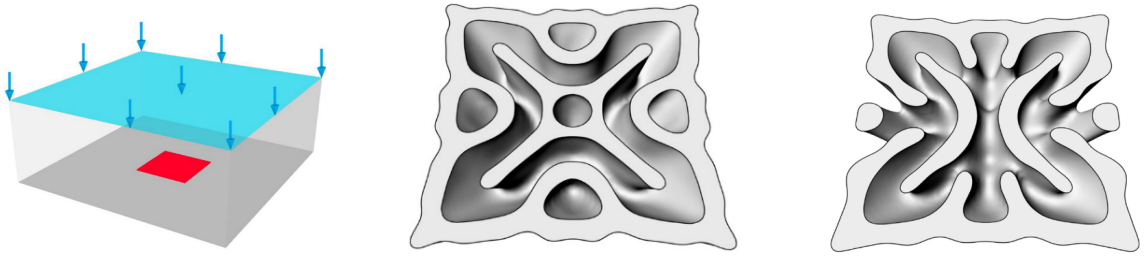


Figure 2.6: Solution of 2D problem [20].



(a) (b) (c)

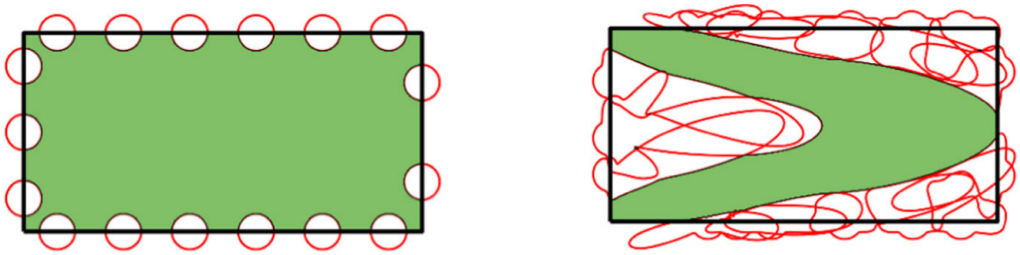
Figure 2.7: Path generation in a 3D problem [20] (a) Description of platform problem; (b) cross-section of free-form solution and (c) cross-section of solution with generated paths.

This method costs little computational effort and uses the free-form topology optimized structure as starting point, but it has two large drawbacks. The first drawback is that it assumes the shortest path will automatically have the least negative impact on the compliance of the structure, while this will not always be the case. The second drawback is that the void detection is done by a non-differentiable operation, which makes integration into topology optimization difficult. A final minor drawback is that the different hierarchical levels have to be chosen and its effect on computation time is difficult to determine by hand.

2.4 Side constraint Method

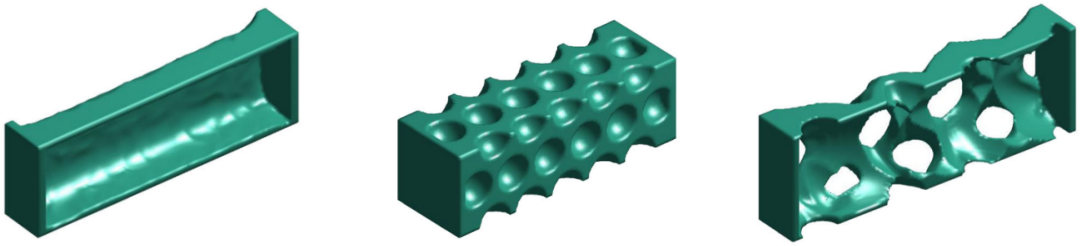
The side constraint method proposed by Zhou et al. [21] uses the level set material model to define void features, which are predetermined before the optimization. The optimization is carried out by changing the size and shape

of the existing void features, which are restricted to the shape of a super-ellipse. A super-ellipse can be seen as a generalized representation of a circle, ellipse, square and rectangle, therefore the optimization can be summarized as finding the optimal width, height, location, shape and angle of every super ellipse. Because additional voids cannot be nucleated during the optimization process, enclosed voids can be prevented by restricting the centerpoints of all predetermined voids to be outside the design domain. Figure 2.8 shows an example of a 2D problem optimized using the side constraint methods, where all predefined voids are denoted with red outlines. Figure 2.9 shows a 3D torsion beam problem optimized with this method. These methods do have a major issue, restricting voids to the outside of the design space can prove too restrictive in combination with the super ellipse shape. Furthermore all voids being predefined makes it so that some initial calculation or choice is required on a sufficient number of voids.



(a) (b)

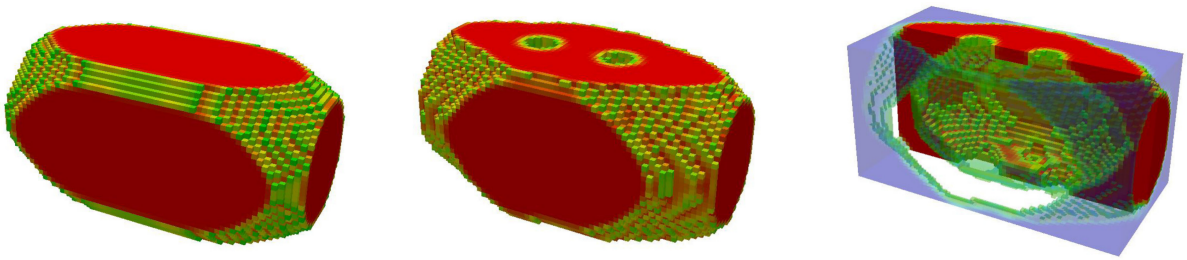
Figure 2.8: Compliance problem optimized using the side constraint method [21]. (a) Initial void configuration; (b) Final result.



(a) (b) (c)

Figure 2.9: Torsion problem optimized using the side constraint method [21]. (a) Cross-section of free-form solution; (b) Initial void configuration and (c) cross-section of final result.

Gaynor and Johnson [22] used a similar approach in the void projection method, which is based on the SIMP material model and the Heaviside projection method. The Heaviside projection method is used to solve the issue of minimum feature size, by taking a weighted average density for a predefined region and using this weighted density to determine the density of all elements in that region. The void projection method changes the design problem to determining where the void should exist, which is restricted to nucleating from a designated surface by the program. Additionally the method is integrated with the overhang projection method [23] to also comply with the overhang restrictions of AM. The solution to the torsion problem found with this method is visible in Figure 2.10.



(a) (b) (c)

Figure 2.10: The void projection method applied to a beam torsion problem [22]. (a) free form solution; (b) solid solution with void projection method and (c) cut view of solution with void projection method.

This technique solves most of the previous drawbacks for the side constraint method, as only a surface needs to be predefined and the shape of the void features is not restricted to a super ellipse. However some issues are still present: the void projection method used produces rounded corners, which in certain cases means that the design space is not utilized fully. Furthermore the algorithm is unable to produce more complex void pathways, which are useful in heat exchanger design for example. Figure 2.10 also shows this issue where the algorithm produced four holes, while one or two holes would be sufficient to get rid of trapped powder.

2.5 Comparison of current methods

Table 2.1 shows a comparison of all methods discussed in this chapter, rated on five basic criteria:

- **User control** is a measure of how much control a potential user has over the properties of the void elimination features.
- **Computational effort** shows how computationally expensive the method is.
- **Ease of use** measures how much additional user effort is needed to implement the method.
- **Mechanical performance** rates the method on how many unnecessary void elimination pathways are created.
- **Geometric complexity** is a measure of the additional geometric restrictions that the method imposes.

The given ratings are based on advantages and drawbacks discussed in the papers themselves and this chapter.

Method	User Control	Computational Effort	Ease of Use	Mechanical Performance	Geometric Complexity
Casting Constraint [11][12][13]	−	++	+	−−	−−
Machining Constraint [14][15][16]	+−	++	+	−	−−
Shortest path [20]	+	++	+−	−	++
Virtual Temperature Method [17][18]	−−	+	−−	+−	++
Side Constraint Method [21]	−	+	−	−	−
Void Projection Method [22]	++	+	+−	+	−

Table 2.1: Comparison of new methods with current void elimination methods

The table clearly reiterates that while some methods are performing quite well, there is no method that performs well on every criteria. Additionally only four methods have been found that are specifically designed for AM, of which the most promising method is the Void projection method by Gaynor et al. [22], but it still has its drawbacks. A new method is succesful when it performs well on each of the five criteria.

3 | Eigenfrequency Method

3.1 Working principle

The first method to be investigated is the eigenfrequency method. It is based on the fact that the eigenfrequency of a body is zero when it is not connected to a boundary, because then it can translate freely. This principle can then be used on the inverted density field during the topology optimization process to recognize any enclosed voids, where a low eigenfrequency indicates the presence of an enclosed void. Enclosed voids can then be prevented from forming during the optimization by adding a minimum constraint to the eigenfrequency. This eigenfrequency is calculated with a generalized eigenvalue problem involving the stiffness and mass matrix, which requires a large amount of computational effort for larger meshes. Therefore degrees of freedom corresponding to a specific direction can be omitted from the stiffness and mass matrices, essentially locking a direction in place to reduce the size of the problem. The working principle of this method can be summarized by the following basic process:

1. Invert current density field
2. Use inverted density field to determine stiffness and mass matrix
3. Optionally lock certain directions to save on computation time
4. Determine eigenfrequencies of inverted density field by solving an generalized eigenvalue problem involving stiffness and mass matrix.
5. Add constraint on determined eigenfrequencies to prevent enclosed voids from forming.

3.2 Problem formulation

3.2.1 Optimization problem

Using the working principle of the eigenfrequency method it is possible to update the optimization problem:

$$\begin{aligned} \min_{\rho\in\Omega} \quad & \mathbf{f}^T\mathbf{u} \\ \text{s.t.} \quad & \mathbf{K}(\rho_e)\,\mathbf{u} = \mathbf{f} \\ & \frac{\sum_{e\in\Omega}\rho_e v_e}{V_{cons}} - 1 \leq 0 \\ & \frac{\lambda_{cons}}{\lambda_{inv}} - 1 \leq 0 \\ & 0 < \rho_{min} \leq \rho_e \leq 1 \end{aligned} \tag{3.1}$$

Here $\mathbf{f}^T\mathbf{u}$ is the compliance (c) minimized during the optimization, with element densities ρ as design variables with their maximum and minimum described in the final constraint. Compliance is calculated by multiplying the transposed force vector $\mathbf{f}$ with the displacement vector $\mathbf{u}$. The first constraint the optimization is subject to is the equilibrium equation for a static object discretized using finite elements, which also uses the previously described force and displacement vector as well as the global stiffness matrix $\mathbf{K}$. This constraint is inherently satisfied every iteration as the objective is determined using the same relation. The second constraint is the volume constraint, where $\rho_e v_e$ denotes the volume per element and V_{cons} the imposed volume constraint. The third constraint is this method's eigenfrequency constraint, which is calculated in the steps described in Section 3.1 and uses the imposed eigenfrequency constraint

λ_{cons} and the eigenfrequency of the inverted density field λ_{inv}. Since λ_{inv} is part of the new method to be investigated its definition will be explained further using the following equations:

$$\rho_{inv} = 1 + \rho_{min} - \rho_e \tag{3.2}$$

$$(\mathbf{K}(\rho_{inv}) - \lambda_{j,inv}\mathbf{M}(\rho_{inv}))\,\Phi_j = 0 \tag{3.3}$$

Here the inverted densities ρ_{inv} are used to determine the inverted stiffness $\mathbf{K}_{inv}$ and mass $\mathbf{M}_{inv}$ matrices, which are subsequently used to calculate their generalized eigenvalues λ_{inv} and eigenmodes Φ. These eigenvalues are then the square of the actual eigenfrequency ω_{inv}. The eigenmodes are also mass-normalized using the following equation:

$$\Phi_i^T \mathbf{M}(\rho_{inv})\Phi_j = \delta_{ij}, \tag{3.4}$$

in which δ_{ij} describes the Kronecker delta.

In reality the density used for the calculation of the inverted stiffness matrix is slightly more complex, as it also uses some additional penalization on intermediate densities as proposed by Zhu et al. [24]. In order to make the use of intermediate densities to satisfy the eigenfrequency constraint more costly, the densities used in the stiffness matrix calculation are defined as:

$$\rho_K = (1-\gamma)\rho_{inv} + \gamma\rho_{inv}^3 \tag{3.5}$$

Here γ denotes the penalization factor used, where for $\gamma = 1$ a full SIMP penalization is applied to the densities and for $\gamma = 0$ the unchanged densities are used. At $\gamma = 1$ Zhu et al. [24] found that a problem arises where the stiffness for low density areas is so low that internal modes will start dominating the lowest eigenfrequencies. This means that the constraint will be active on these low density internal modes instead of the rigid body modes of enclosed void areas. Because of this a γ of 0.9 is used in all results.

These constraints are not the only addition to the optimization, a density filter with a radius of two elements as proposed by Bruns and Tortorelli [25] is also used to prevent checkerboarding and impose a minimum lengthscale. Furthermore as already discussed in the scope the optimization algorithm used is MMA [10] and is then used for density based topology optimization with SIMP.

3.2.2 Sensitivity analysis

Eigenvalue optimization is not a new topic and the derivation of its sensitivities can be found in literature [26][27], where the sensitivity of an eigenfrequency λ_j with respect to an element density ρ_K is equal to:

$$\frac{\partial \lambda_j}{\partial \rho_{K,e}} = \Phi_j^T \frac{\partial \mathbf{K}}{\partial \rho_{K,e}} \Phi_j - \lambda_j \Phi_j^T \frac{\partial \mathbf{M}}{\partial \rho_{K,e}} \Phi_j. \tag{3.6}$$

In this method however the eigenfrequency calculation is done on the inverted and penalized density field, therefore for it to apply to this method additional steps are necessary:

$$\frac{\partial \lambda_j}{\partial \rho_e} = \frac{\partial \lambda_j}{\partial \rho_{K,e}} \frac{\partial \rho_{K,e}}{\partial \rho_{inv,e}} \frac{\partial \rho_{inv,e}}{\partial \rho_e} = \left(\lambda_j \Phi_j^T \frac{\partial \mathbf{M}}{\partial \rho_e} \Phi_j - \Phi_j^T \frac{\partial \mathbf{K}}{\partial \rho_e} \Phi_j \right) (3\gamma\rho_e^2 - \gamma + 1). \tag{3.7}$$

3.3 Constraint limit

Determining the constraint limit is an important step in the implementation of the eigenfrequency method. A constraint that is too loose would result in designs where some enclosed voids are still present, but making the constraint too strict could result in a sub-optimal design. However this is not a trivial task, because eigenfrequencies are determined with an eigenvalue problem that is difficult to solve by hand. Therefore numerical testing is required to determine whether or not it is possible to directly influence the void elimination features by changing the eigenfrequency constraint.

3.3.1 Simple beam model

A preliminary way to determine the correct eigenfrequency constraint in 2D is to equate the problem to a simple mass-spring system, where the mass is determined by the size of the void and the spring stiffness is determined by the cross-section and length of the void elimination feature. For this study both were assumed to be rectangular in

shape. The stiffness of the spring can be subdivided into two categories: the translational stiffness of the spring and the rotational stiffness of the spring. Both these properties can be determined from basic rules-of-thumb on beam mechanics, while keeping in mind the basic relation of $F = Ku$ for translational and $M = K\theta$:

$$K_{trans} = \frac{EA}{l} = \frac{Eht}{l}, \tag{3.8}$$

$$K_{rot} = \frac{EI}{l} = \frac{Eth^3}{12l} \tag{3.9}$$

These equations show how the stiffness of the beam is influenced by the Young's modulus of its material E, width h, length l and thickness t. The translational mass of the system is rather straightforward, as it is simply the mass of the block on the end of the beam. However the rotational mass is a bit more complex, as it is more accurately described by the moment of inertia of the mass around the base of the beam. The two masses can then be determined with the following equations:

$$M_{trans} = \rho V = \rho LHt, \tag{3.10}$$

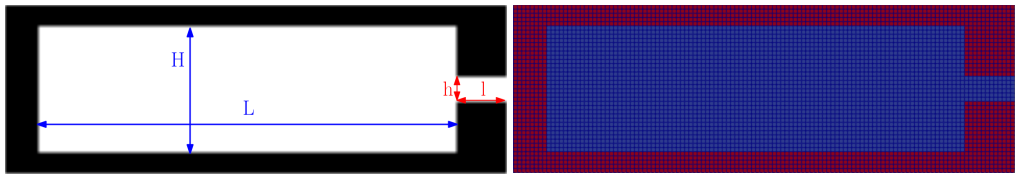
$$M_{rot} = \rho V r^2 = \rho LHt\,(l + 0.5L)^2, \tag{3.11}$$

where ρ is the density of the material, L denotes the length of the mass, H its width, t its thickness and r the distance between the center of the mass and the base of the beam. Here uppercase letters are used to differentiate the mass variables from the beam variables, of which a more clear definition is shown in Figure 3.1a. Furthermore since this is a 2D problem the thickness t does not vary between the mass and beam. These equations can then be combined to get a general formula for any rectangular beam-mass combination:

$$\omega_{trans} = \sqrt{\frac{K_{trans}}{M_{trans}}} = \sqrt{\frac{Eh}{l\rho LH}}, \tag{3.12}$$

$$\omega_{rot} = \sqrt{\frac{K_{rot}}{M_{rot}}} = \sqrt{\frac{Eh^3}{12l\rho LH\,(l + 0.5L)^2}}. \tag{3.13}$$

These equations do show that there is a relation between the mass of the void and the dimensions of the void elimination feature, where for instance a larger mass or a mass that is further away from the boundary will require a wider channel than a smaller mass or a mass that is close to the boundary. This could be beneficial in that in situations where there is more powder to be removed or a longer channel to remove it through the width of the channel is automatically increased to accommodate that. However, it could also be a disadvantage as it makes the specific shape and size of a void elimination feature more difficult to directly control. Using the dimensions of the test case depicted in Figure 3.1b and the material properties of Ti-6Al-4V, we can now determine an eigenfrequency from these equations. All input variables and their corresponding output eigenfrequencies are shown in Table 3.1.



(a) **(b)**

Figure 3.1: Test case for beam model with (a) the explanation of variables and (b) Test case in elements where one element is 2.5mm by 2.5mm.

3.3.2 Beam model comparison with test cases

The beam model gives an initial guess for the true value of the eigenfrequency constraint, but it is now necessary to validate that calculation by performing tests on different cases with the finite element method. The example from the beam model is the first test case and the results for ω_{trans} and ω_{rot} are $1.53 \cdot 10^4$ and $7.20 \cdot 10^2$, respectively.

From these results it is clear that while the analytically determined rotational eigenfrequency closely resembles its computational counterpart, there is a discrepancy between the two translational eigenfrequencies. For this discrepancy two main reasons can be identified. Firstly, the necessity of a minimum density value in the finite element method to

Property	Symbol	Value	Unit
Young's Modulus	E	$113.8 \cdot 10^9$	$\mathrm{N\,m^{-2}}$
Density	ρ	$4.43 \cdot 10^3$	$\mathrm{kg\,m^{-3}}$
Beam Width	h	0.015	m
Beam Length	l	0.03	m
Mass Width	H	0.075	m
Mass Length	L	0.25	m
Translational Eigenfrequency	ω_{trans}	$2.62 \cdot 10^4$	$\mathrm{rad\,s^{-2}}$
Rotational Eigenfrequency	ω_{rot}	$7.32 \cdot 10^2$	$\mathrm{rad\,s^{-2}}$

Table 3.1: First test case for beam model

prevent singular matrices envelops the mass spring system in low stiffness material which increases the eigenfrequency. Secondly and more importantly in the model the beam is only considered to add stiffness and the inverted void region is only considered to add mass. In reality the case is more complex where the beam and the void region both add to the stiffness and mass matrix. The test case used has a rather slender mass and because of this the actual length of the void removal feature is underestimated, this can be tested by varying the length of the void elimination feature.

The discrepancy between analytical and computational translational eigenfrequency values on the initial test necessitates additional tests to get a better understanding on how different void removal features affect the eigenfrequency and also to determine if the analytically determined rotational eigenfrequency still resembles its computational counterpart when parameters are varied. Four main variables of interest can be identified: the mass of the void and the length, width and density of the void removal feature. These variables were implemented both analytically and using finite elements in Figure 3.2, which shows how the eigenfrequency changes with respect to the four variables.

The first observation that can be made from this figure is that rotational eigenfrequencies are generally always lower than its translational counterpart, although an increased width of the void elimination feature reduces the difference between the two eigenfrequencies. This is explained by how the eigenfrequencies scale with regards to the feature width in equation 3.8 and 3.9, where the rotational eigenfrequency scales with a power of 3, while the translational eigenfrequency scales linearly. Another observation is that the analytical translational results consistently overestimate the eigenfrequency because of the discrepancy between the models, except for when the inverted void area shows rigid body modes, because then the minimum densities add some stiffness. The analytical rotational eigenfrequencies show similar behaviour, but instead they mostly closely resemble their computational counterpart, except for when the minimum densities start having a significant contribution to the stiffness.

Finally, the figures show that multiple possible combinations of void size and feature width, length or density can be found for any given eigenfrequency, which indicates that it will be difficult to directly choose the properties of the void elimination feature and void size by changing the eigenfrequency constraint.

3.4 Aggregation of eigenfrequencies

When only the single lowest eigenfrequency is considered several issues arise for the optimization algorithm. Multiple voids could not be sufficiently dealt with as only one enclosed void is considered every iteration. This results in a phenomenon that any time the method deals with a void for an iteration, then at the next iteration a new void gets considered, for which sensitivities are still unknown and this allows the original void to open up again. It is therefore necessary to calculate multiple eigenfrequencies for every iteration, which can be implemented in different ways. The simplest way would be to introduce a separate constraint for every calculated eigenfrequency, but this would result in the constraints constantly switching around as the optimization is running and new voids nucleate or existing voids get smaller or larger. Constraints switching around is problematic because MMA uses information from the two previous iterations to determine a new linear sub-problem, which would then be incorrect for any switched constraint. This could possibly be fixed by mode tracking, but this requires additional computational effort. For this reason there is a need for a way to aggregate multiple eigenfrequencies into a single constraint. Several techniques for aggregation have been proposed in literature, the most common techniques consisting of the p-norm and the Kreisselmeier-Steinhauser

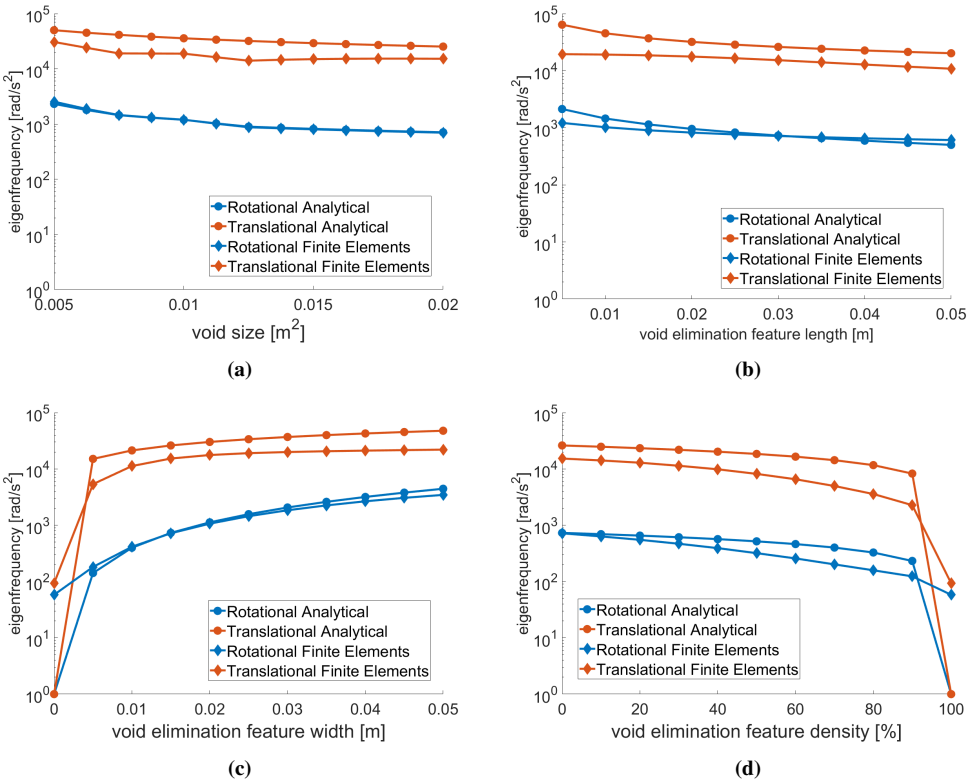


Figure 3.2: Eigenfrequency with respect to (a) volume of the void, (b) length of the void removal feature, (c) width of the void removal feature and (d) density of the void removal feature.

function [28], which can be described using the following equations:

$$\Psi_p = \left(\sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}}, \tag{3.14}$$

$$\Psi_{KS} = \frac{1}{p} \ln \left(\sum_{i=1}^{n} e^{p x_i} \right). \tag{3.15}$$

Here a higher absolute value of p means a closer approximation to the true maximum or minimum function, while a positive and negative p describe a maximum and minimum function, respectively. In this case the lowest eigenfrequency should get the most attention, while still keeping track of the higher eigenfrequencies and their sensitivities. Therefore the p-factor used in these techniques should remain negative and quite small, while it would normally be larger to more closely represent the true minimum or maximum value. The total amount of modes considered should also be taken in consideration, as it is very inefficient to determine all modes when only a few low frequency modes are actually important. Furthermore the choice of technique matters less than in for instance stress constraints, where an underestimation could mean failure of the part. In this case an underestimation or overestimation would mean a slight alteration in the size of the void or void removal feature, which is not an issue. In the end the choice was made to use p-norm aggregation with $p = -4$ to aggregate thirty modes with the lowest eigenfrequency. This slightly changes the sensitivities determined in Section 3.2.2, as the following additional term has to be added:

$$\frac{\partial \lambda_{ag}}{\partial \lambda_j} = \lambda_j^{p-1} \left(\sum_{i=1}^{n} \lambda_i^p \right)^{\frac{1}{p}-1}, \tag{3.16}$$

where λ_{ag} denotes the aggregated eigenfrequency and n denotes the total amount of eigenfrequencies calculated. λ_{ag} is the value that actually gets used in Equation 3.1 for the eigenfrequency constraint.

3.5 Results and Discussion

3.5.1 2D geometric optimization

To test the effectiveness of the newly proposed method a geometric approach is first considered. This approach takes an already optimized design for a given problem and only considers the eigenfrequency constraint on this design, while leaving compliance out of the equation. This allows us to get a better look solely at how the new method handles enclosed voids and possible issues that arise are then easier to analyze. Instead of compliance the optimization is done on the least squares error between the input design and the current design. The formulation for the constraint can be found in Section 3.2.1, but the objective is new and can be described using the following equation:

$$\min_{\rho \in \Omega} \sum_{i=1}^{\Omega} (\rho_i - \rho_{i,in})^2. \tag{3.17}$$

The method was first tested on a simple 2D cantilever beam problem, which has a supported edge on the left and a downward force applied on the right with a maximum volume fraction of 0.5. The full problem definition and its solution using only the volume constraint is found in Figure 3.3.

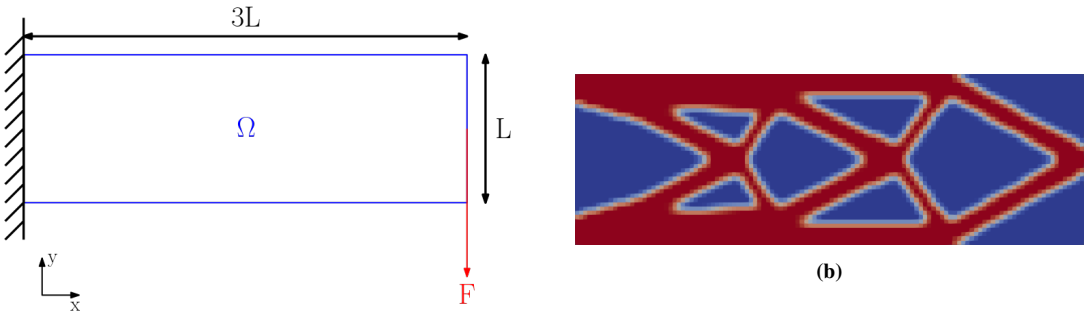


Figure 3.3: Cantilever problem divided into 120x40 elements; (a) Cantilever beam problem description, with $L = 0.1$m and (b) Optimization with only volume constraint with c_{ref} = 100.

From Figure 3.3 we can deduct that there are a total of six enclosed voids present in the unconstrained solution, which makes this problem a suitable test for the eigenfrequency method to eliminate these.

The results of the geometric approach for different eigenfrequency constraints are visible in Figure 3.4. Here the main issue with the new method becomes abundantly clear, as these designs had their constraints satisfied by utilizing intermediate densities, which are not desirable as they can not be related to a real world design. This happened even though additional penalization on the inverted stiffness matrix was used as discussed in Section 3.1.2. Additionally the designs show some checkerboard patterns, as these have been known to have an artificially high stiffness and a density filter is not applied.

Next to the penalization already present in the eigenfrequency calculation itself, another remedy would be to add an additional penalization on intermediate densities to the objective, in this geometric case by raising the objective to the power of a penalization factor ζ, with ζ larger than 0 and smaller than 0.5. The effect of this additional penalization is that smaller differences are more expensive to use than complete differences, rather than the other way around as in Equation 3.15. Figure 3.5a shows a penalization of 0.1 does force the method to create channels of void material, but at the cost of adding considerable noise visible in Figure 3.5b. This noise is caused by the fact that this penalization causes gradients of the objective to both go to infinite and change sign at a difference of zero, which is not something the optimizer can deal with.

Effect of Locking DOFs

The computational effort necessary to calculate the eigenfrequencies gets exponentially larger when increasing the amount of elements, which is why restricting a direction of these elements can become a necessity to curb the computational effort required. Performing this on a geometric optimization first allows us to see how locking different degrees

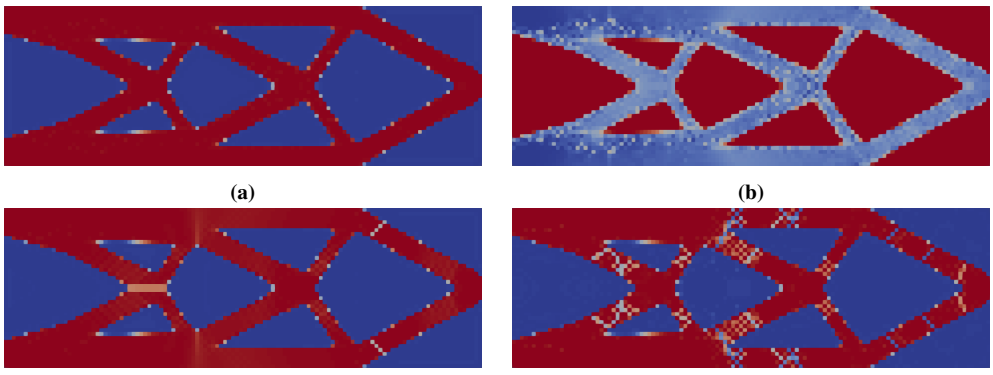


Figure 3.4: Geometric optimization with increasing eigenfrequency constraint. (a) $\omega^2_{cons} = 10^7$; (b) $\omega^2_{cons} = 10^7$ with inverted and logarithmically scaled density field; (c) $\omega^2_{cons} = 10^8$; (d) $\omega^2_{cons} = 10^9$.

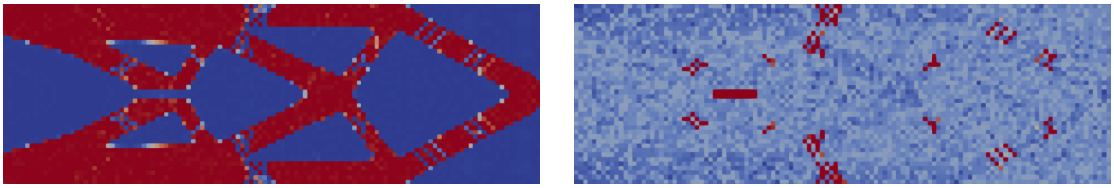


Figure 3.5: Geometric optimization with an eigenfrequency constraint of $\omega^2_{cons} = 10^9$ and extra penalization on intermediate densities. (a) Normal result; (b) the noise present, visualized by subtracting the result from the original input design.

of freedom for the eigenfrequency calculation will impact the way enclosed voids are dispersed. In this case all previously discussed methods to penalize intermediate densities were applied and the eigenfrequency constraint was set to $\omega^2_{cons} = 10^9$. Figure 3.6 shows the geometric optimization with the x-direction and y-direction locked, respectively.

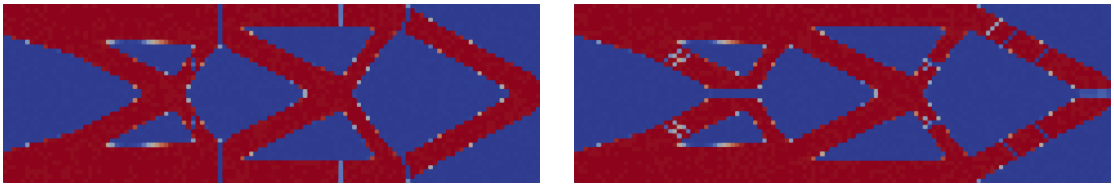


Figure 3.6: Geometric optimization with different degrees of freedom locked. (a) x-direction; (b) y-direction.

For these results the computational effort was considerably less, as the time required for the optimization reduced by a factor of at least two. They also show how locking a specific degree of freedom causes the void pathways to be formed in the remaining unlocked degree of freedom, so when the horizontal x-direction is locked the void pathways will all be orientated vertically and vice-versa. This is because when a degree of freedom is locked movement is only possible in the direction of the remaining degree of freedom, for which adding a void elimination feature loaded in tension is the option that adds the most stiffness. Another interesting thing to note is how the void pathways are smaller and less numerous than when all degrees of freedom are considered, because locking degrees of freedom also removes rotational eigenfrequencies. Rotational eigenfrequencies are almost universally the lowest eigenfrequency as shown in Section 3.3.2 and thus locking degrees freedom causes a slight relaxation of the eigenfrequency constraint.

3.5.2 2D compliance problem

Intermediate densities

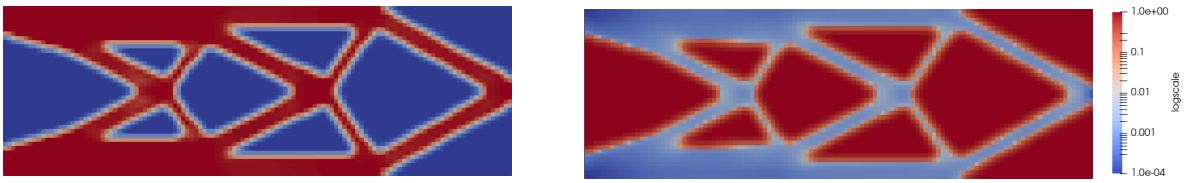
The geometric tests clearly show the main Achilles' heel of this method, where intermediate densities are often enough to satisfy the eigenfrequency constraint. The next tests will now also consider compliance, which could exacerbate this

problem due to the large increase in compliance enclosed void elimination causes.

Three options to alleviate the problem of intermediate densities will be considered. The first option is to simply increase the eigenfrequency constraint, which could force the channels of intermediate densities to be so wide that the optimizer eventually prefers to get rid of the void. The second option is to increase the SIMP penalization on the compliance to make any intermediate density highly expensive in terms of compliance, making it a less desirable option than simply closing enclosed voids. Finally, the third option is to utilize the robust formulation proposed by Wang et. al [29] which uses a smooth Heaviside projection to change densities and force them to a binary state.

Regular SIMP penalization

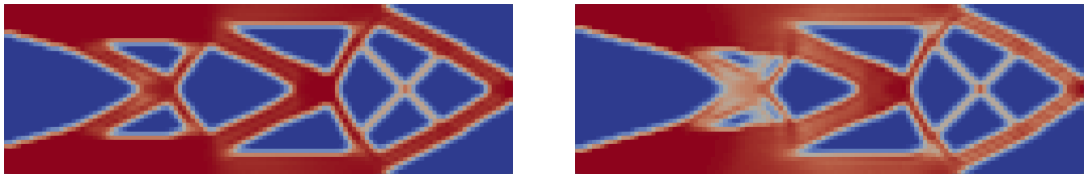
Before going through the trouble to alleviate the issue of intermediate densities the method was first tested with regular SIMP settings, to see if the suspected problems actually do occur. For this the method was implemented with the eigenfrequency values found in Section 3.3, which resulted in the topology found in Figure 3.7a. At first sight this topology seems to be similar to the topology found in the unconstrained optimization, which should not be possible with the eigenfrequency constraint. Inspecting the topology further by inverting the density field and applying a logarithmic scaling to it in Figure 3.7b is what shows the expected outcome. The optimizer did manage to satisfy the eigenfrequency constraint by having each enclosed void connected to the boundary with a band of lower density elements, even with SIMP penalization this is favourable for compliance when compared to completely getting rid of enclosed voids.



(a) **(b)**

Figure 3.7: Cantilever problem with tested eigenfrequency constraint of $\omega^2_{cons} = 10^7$. (a) Regular density field and (b) Inverted logarithmic density field.

The first option discussed above is to increase the eigenfrequency constraint and resulting designs for that are shown in Figure 3.8. These still clearly show the same issue found with a lower eigenfrequency constraint, albeit in a more obvious manner. In other words, the optimizer still manages to satisfy the eigenfrequency constraint by utilizing elements with intermediate densities to connect enclosed voids to the boundaries.



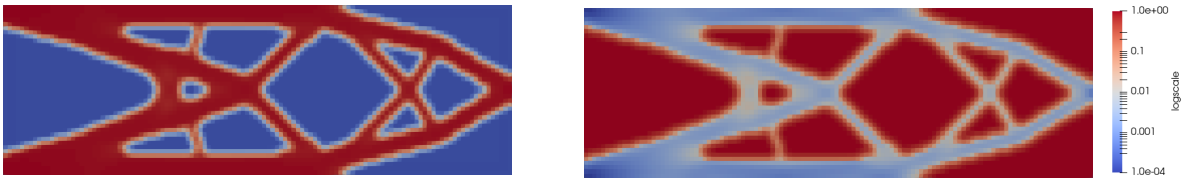
(a) **(b)**

Figure 3.8: Cantilever problem with increased eigenfrequency constraint of (a) $\omega^2_{cons} = 10^{8.5}$ and (b) $\omega^2_{cons} = 10^9$.

Extreme SIMP penalization

Even with larger eigenfrequency constraint values the intermediate density problem still persists with regular SIMP penalization, therefore the second option to further increase the penalization factor from $q = 3$ to $q = 10$ is now studied. The first test of this solution is to once again use the eigenfrequency determined in Section 3.3, of which the results are shown in Figure 3.9. Again the optimizer finds a way to satisfy the eigenfrequency constraint by using intermediate densities even with increased penalization.

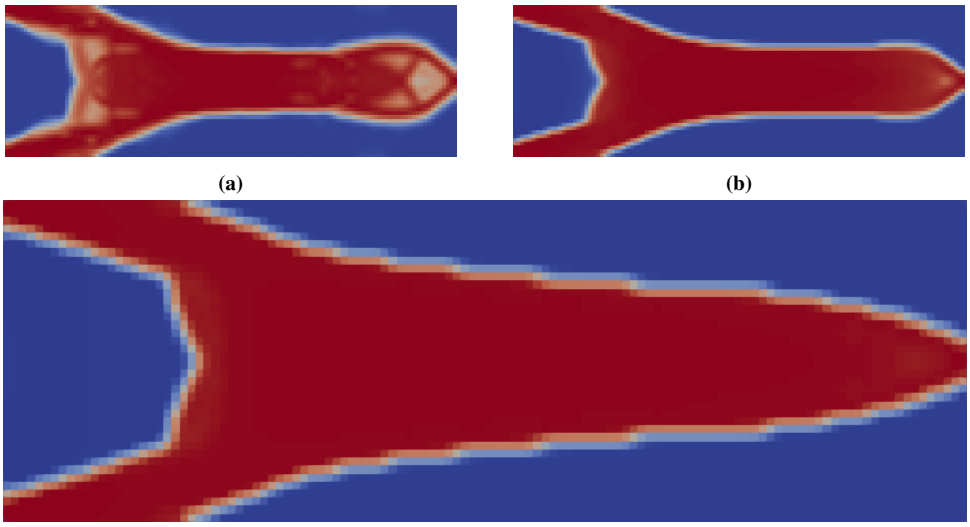
The next step is to combine this increased SIMP penalization with a higher constraint limit to see if this forces the optimizer to truly get rid of enclosed voids. This is where the method finally seems to achieve the desired result as



(a) **(b)**

Figure 3.9: Cantilever problem with tested eigenfrequency constraint ($\omega^2_{cons} = 10^7$) and $q = 10$. (a) Regular density field and (b) Inverted logarithmic density field.

shown in Figure 3.10. The optimizer does initially open up voids in the structure, but these quickly get closed to satisfy the eigenfrequency constraint.



(a) **(b)**

(c)

Figure 3.10: Cantilever problem with high eigenfrequency constraint $\omega^2_{cons} = 6 \cdot 10^9$, $q = 10$ and $\frac{c}{c_{ref}} = 268.7\%$; (a) iteration 15, (b) iteration 30 and (c) iteration 100.

Unfortunately when inspecting this result more thoroughly by looking at its shape, compliance and the inverted and logarithmic density field it becomes clear that this result still leaves a lot to be desired. The compliance has seen an increase that is much larger than could be expected when eliminating enclosed voids [21] and this can be attributed to two main causes. The first cause is simply that the shape is suboptimal even when considering the removal of enclosed voids, as the optimal shape is considered to be a large sideways arch shown in Figure 2.8 rather than a short arch with one single beam towards the load. The second cause for this large increase becomes clear when once again inspecting the inverted density field in Figure 3.11. The densities of the design are less uniform than the initial result would suggest, with many elements having a density of slightly less than one. This occurrence combined with the heavy penalization on intermediate densities is the cause of the large increase in compliance. Now what is left unanswered is how the optimizer again opts to use these intermediate densities even though there are no enclosed voids for which they could have been used. Figure 3.12 shows how it turns out that the design without an eigenfrequency constraint is exactly the same as with the tested eigenfrequency constraint in Figure 3.9, meaning that this behaviour is caused by the high penalization itself rather than any change in the eigenfrequency constraint. One possible cause of this behaviour is that the sensitivities near high densities are so high that the step size used by the optimizer becomes very small, therefore causing the result to get stuck in this suboptimal shape.

Robust formulation

As described at the start of this section, the robust formulation proposed by Wang et al. [29] could also be used to force the design towards a binary solution by applying a smooth Heaviside projection to the regular and inverted densities,

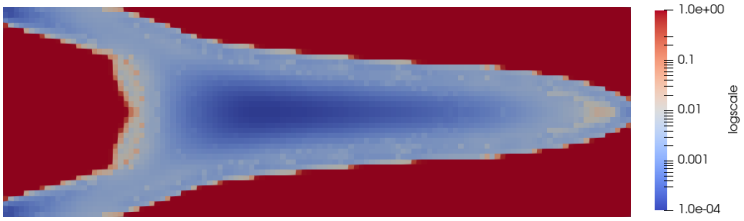
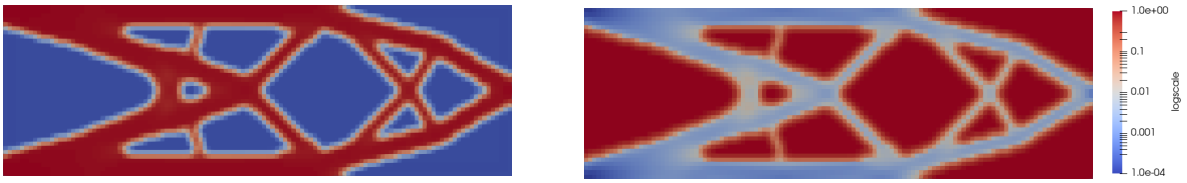


Figure 3.11: Inverted and logarithmic density field for high penalization and high eigenfrequency constraint.



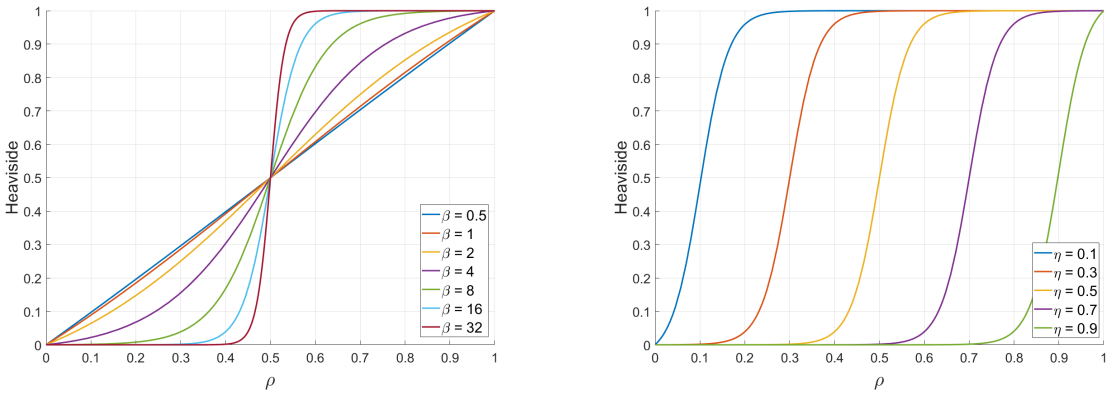
(a) **(b)**

Figure 3.12: Unconstrained cantilever problem at high penalization (a) Normal result and (b) inverted logarithmic density field.

which can be described as:

$$H(\rho) = \rho_{min} + \frac{\tanh(\beta\eta) + \tanh(\beta(\rho - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))}. \tag{3.18}$$

Here β controls the steepness of the function and η defines the point at which the function steps. Figure 3.13 shows how the function behaves for these two parameters. The method works by always considering a worst case design to optimize, which is achieved by eroding the stiffness matrix and dilating the mass matrix. Erosion and dilation is done by either lowering η to dilate the design or increasing η to erode the design. In practice this method utilizes a continuation in β, which means that as the iterations increase the projection gets sharpened up to a predetermined limit. This is done to avoid convergence to a local minimum early in the optimization and can be implemented in various ways, but in the studies in this thesis β starts at 0.5 and is increased by 5% every iteration. Due to constantly changing the input value needed to output a specific intermediate density this method could make it too difficult for the optimizer to use these to satisfy the eigenfrequency constraint. For all robust results the radius of the density filter [25] was also increased to three elements.

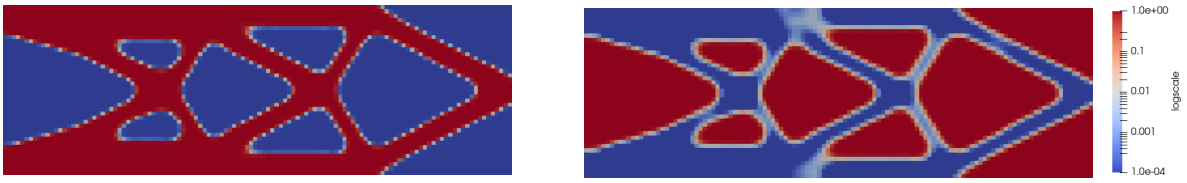


(a) **(b)**

Figure 3.13: Smooth Heaviside function when varying (a) β and (b) η.

First the robust formulation was applied using a β limit of 20 and the tested eigenfrequency constraint. The results are visible in Figure 3.14 and they once again show how the optimizer opts to utilize intermediate densities to satisfy the constraint.

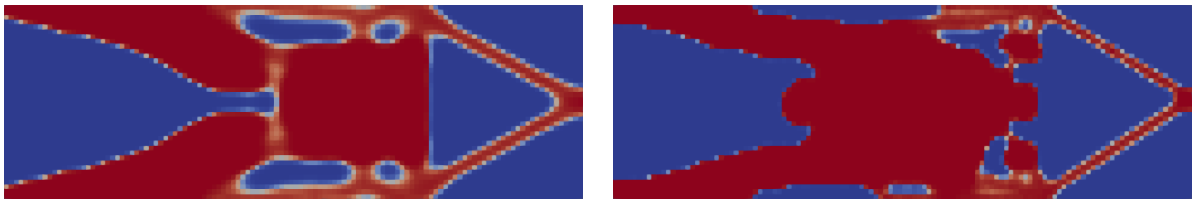
The next step is to once again increase the eigenfrequency constraint to see how it will affect the design, which is done in Figure 3.15. This result has less voids than the unconstrained design and the shape seems to tend towards a sideways



(a) **(b)**

Figure 3.14: Robust formulation with a β limit of 20 and the tested eigenfrequency constraint of $\omega^2_{cons} = 10^7$, with (a) Regular density field and (b) inverted and logarithmic density field.

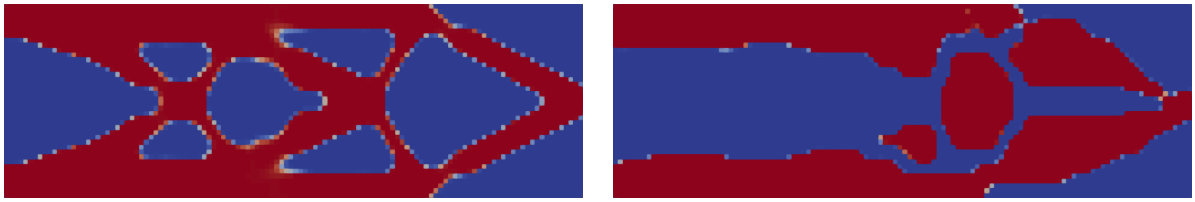
arch, which has been found to be an optimal solution [21]. Unfortunately though some voids are simply deemed too important to eliminate and the method still utilizes intermediate densities for these.



(a) **(b)**

Figure 3.15: Robust formulation with a β limit of 20 and an eigenfrequency constraint of (a) $\omega = 10^8$ and (b) $\omega = 10^9$.

Another possibility is to increase the limit on β in the formulation, which could finally make it impossible to find the correct density to utilize intermediate densities. This is however not common practice because, as shown in Figure 3.13, increasing β subsequently decreases sensitivity values at densities close to 0 or 1. The results for different eigenfrequency constraints for a β limit of 40 are shown in Figure 3.16. The tested eigenfrequency constraint still utilizes small channels of intermediate densities and has a design which is relatively similar to the unconstrained design. This is in stark contrast to the result with a high eigenfrequency constraint, which is a perfect example of why a high limit on β is generally not utilized. While the design does not have any enclosed voids, it is clear that this design is far from anything that could be construed as optimal and the compliance of 971.3 reinforces that point.



(a) **(b)**

Figure 3.16: Robust formulation with a β limit of 40 with an eigenfrequency constraint of (a) $\omega = 10^6$ and (b) $\omega = 10^8$.

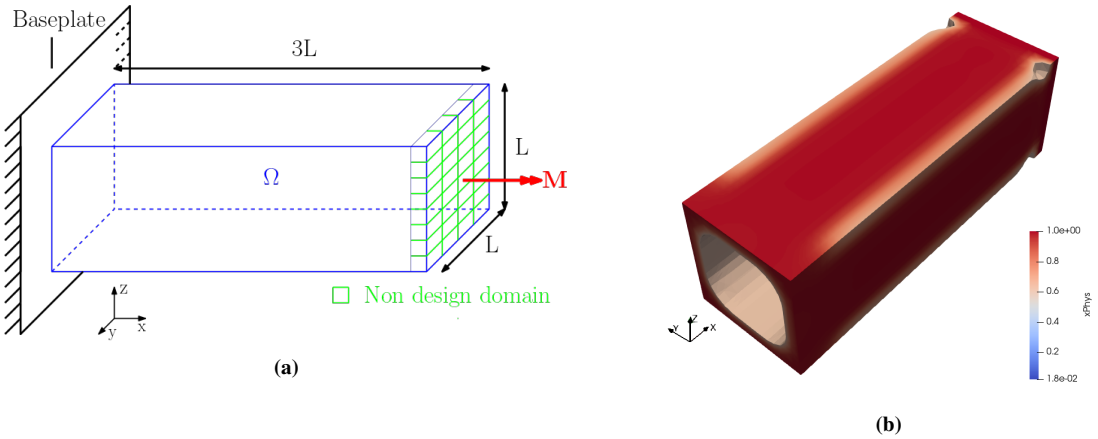
Now that the robust formulation also has not achieved the elimination of enclosed voids, it is safe to say that it is very difficult to eliminate enclosed voids in 2D with the eigenfrequency method. The method's main Achilles heel of intermediate densities has been found to be too much of a problem for it to work properly. Some other combination of specific settings could still be enough to remedy this, but finding these settings is not a trivial task.

3.5.3 3D compliance problem

Problem definition

It is also necessary to test the effectiveness of the method in a 3D environment, this is to see if the method could still be applied to real design cases. Even though the method failed to achieve the desired result in 2D, it has been documented in literature that 3D designs without enclosed voids more closely resemble free-form designs as the optimizer has more directions in which to eliminate these enclosed voids [22].

The test case chosen for this is the torsion beam problem, because it is a rather simple case where the free form solution is known to have a large enclosed void in the middle. Additionally this case is also used in previous work on enclosed voids [17][18][20][21][22], allowing for some comparison between different methods. The full test problem is and its free-form solution is visible in Figure 3.17. All other 3D results will use the same orientation and colour map for densities, unless specified otherwise.



(a) (b)

Figure 3.17: Torsion beam problem, divided into 48x16x16 elements, (a) problem description with $L = 0.1$m and (b) Free form solution with c_{ref} = 100.0.

Effect of constraint values

The first test is to apply different eigenfrequency constraints to the torsion beam to see whether or not the resulting design has a pathway towards the enclosed void. To achieve this the boundary conditions for the eigenvalue problem are on the entire outside surface of the beam depicted in Figure 3.17a, which excludes the surface in contact with the baseplate. Figure 3.18 contains the resulting design for three different eigenfrequency constraints, starting at the tested constraint value from Section 3.3. These results show that once more the tested eigenfrequency constraint value is too low to properly create void elimination pathways. At a constraint value of $\omega^2_{cons} = 10^{8.5}$ a void pathway does finally form near the base plate and at $\omega^2_{cons} = 10^9$ additional holes are also present at several locations along the beam.

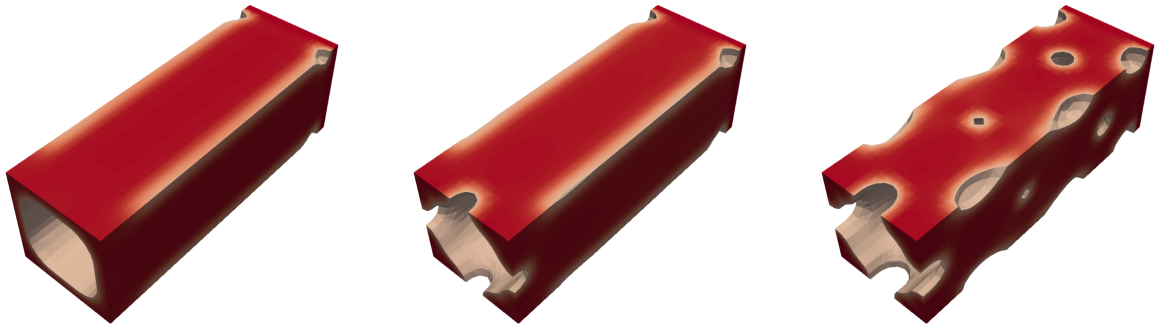
In terms of compliance the first beam shows barely any difference, once again confirming that the tested eigenfrequency constraint is simply too forgiving to have any impact on enclosed voids. Because the stricter constraints do have holes the compliance for them is also increased, which is to be expected. It does show how finding the right choice for a constraint is paramount in this method, as all these unnecessary holes can heavily increase the compliance without adding any additional value.

The average computation effort when applying this method is about 1.43 times higher than the unconstrained optimization, this makes this method a very expensive option to use when eliminating enclosed voids. As discussed before one way to reduce this additional time is to lock certain degrees of freedom in the eigenfrequency calculation.

Effect of Locking DOFs

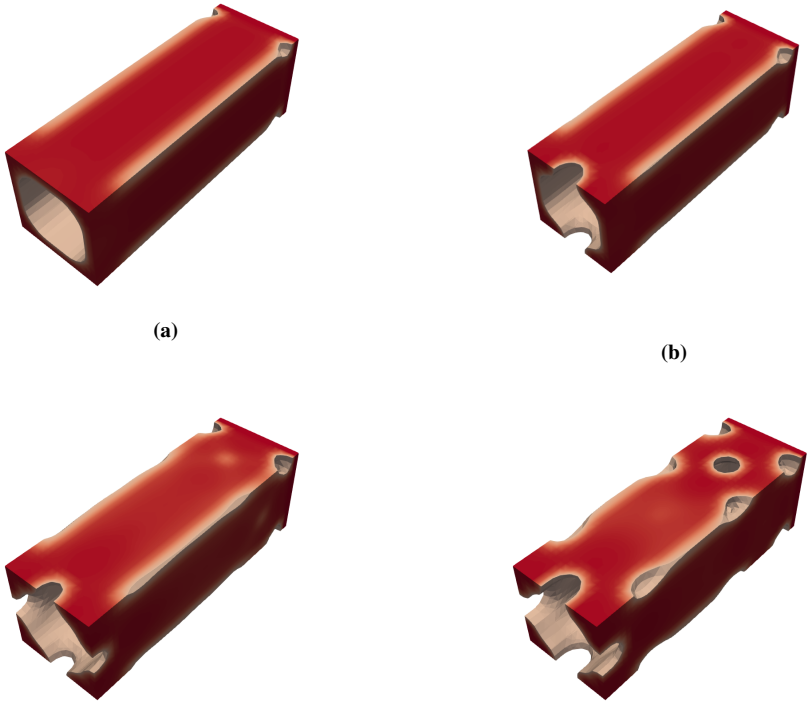
Now that a constraint value has been found that works to eliminate enclosed voids in 3D and that there is a reference design for this constraint value, it is possible to see how locking degrees of freedom affects the final design and computational effort. For this purpose two tests have been conducted for both $\omega^2_{cons} = 10^{8.5}$ and $\omega^2_{cons} = 10^9$, shown in Figure 3.20. These tests are to see what effect locking a single direction has. It is important to note that the problem is symmetric along the y- and z-axis and that any duplicate tests because of this symmetry were omitted.

The results confirm the conclusion from the initial geometric tests done in Section 3.5.1. Firstly it is observed that locking a degree of freedom slightly relaxes the eigenfrequency constraint due to the removal of some rotational eigenfrequencies, which were found to be the lowest in Section 3.3. This can be seen in both the reduced amount of void pathways created as well as the reduced compliance. Additionally void removal features that would add stiffness for the eigenfrequency calculation in the direction that was locked are no longer present, most notably seen in Figure



(a) **(b)** **(c)**

Figure 3.18: Solutions to the torsion beam problem for different eigenfrequency constraints. (a) $\omega^2_{cons} = 10^6$ with $\frac{c}{c_{ref}}$ = 100.0%; (b) $\omega^2_{cons} = 10^{8.5}$ with $\frac{c}{c_{ref}}$ = 103.2%; (c) $\omega^2_{cons} = 10^9$ with $\frac{c}{c_{ref}}$ = 113.7%.



(a) **(b)**

(c) **(d)**

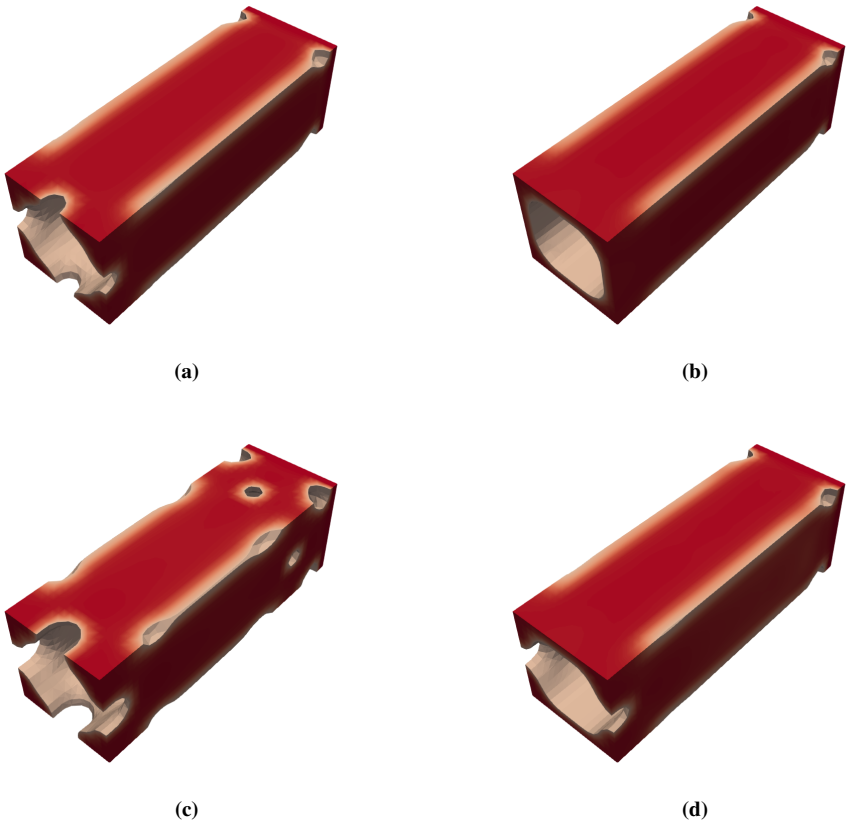
Figure 3.19: Solutions to the torsion beam problem for different eigenfrequency constraints and locked degrees of freedom. (a) $\omega^2_{cons} = 10^{8.5}$ and x locked with $\frac{c}{c_{ref}}$ = 100.8%; (b) $\omega^2_{cons} = 10^{8.5}$ and y locked with $\frac{c}{c_{ref}}$ = 102.3%; (c) $\omega^2_{cons} = 10^9$ and x locked with $\frac{c}{c_{ref}}$ = 108.2%; (d) $\omega^2_{cons} = 10^9$ and y locked with $\frac{c}{c_{ref}}$ = 111.7%.

3.19b, where holes that would add stiffness in the y-direction are removed. For the x-direction this effect is more nuanced, as the two surfaces where a channel would add the most stiffness in this direction are not designated as boundary conditions for the eigenfrequency calculation anyway.

The computation effort was successfully reduced to 1.3 times the unconstrained optimization, this is still a considerable increase and warrants a new test that locks additional degrees of freedom. The advantages of locking certain degrees

of freedom are clear, as it reduces some of the additional computation time required for this method. Additionally the slight relaxation of the constraint improves the compliance slightly, although as shown in Figure 3.19a this could have the adverse effect of no longer eliminating enclosed voids.

Finally another test is to simultaneously lock two directions, of which results are shown in Figure 3.20, where again symmetry causes some tests to be duplicates and those were omitted. Here the trend that locking a single degree of freedom sets gets continued, where the void elimination pathways are smaller and subsequently the compliance is also slightly lower. The effect on computation effort is also very noticable, as that is now reduced to just 1.1 times the unconstrained computation effort.



(a) **(b)**

(c) **(d)**

Figure 3.20: Solutions to the torsion beam problem for different eigenfrequency constraints and multiple locked degrees of freedom. (a) $\omega^2_{cons} = 10^{8.5}$ and x free with $\frac{c}{c_{ref}} = 101.8\%$; (b) $\omega^2_{cons} = 10^{8.5}$ and y free with $\frac{c}{c_{ref}} = 100.0\%$; (c) $\omega^2_{cons} = 10^{9}$ and x free with $\frac{c}{c_{ref}} = 109.3\%$; (d) $\omega^2_{cons} = 10^{9}$ and y free with $\frac{c}{c_{ref}} = 103.9\%$.

Combining the results of all tests done on locking degrees of freedom shows that in the case of the torsion beam locking the x-direction has the largest effect on how the void pathways are generated. This is mostly caused by the previously discussed reason that there are no boundary surfaces in the eigenfrequency calculation where void elimination features that add the most stiffness in the x-direction can be added, which means that when the x-direction is unlocked multiple void elimination channels are necessary. In some cases where the x direction was locked the eigenfrequency constraint was even satisfied with the enclosed void still present. In a relatively simple problem like the torsion beam an educated guess could be made on what direction could be locked without heavily impacting void pathways. However for more complex problems this decision may not be as straightforward, which could pose a problem because it is strongly advisable to utilize the locking of degrees of freedom to reduce computational effort.

Feature location control

A final test for the method is to see if the location of void elimination features could be controlled in some way. In the case of the eigenfrequency constraint this is done by simply setting the boundary conditions used in the eigensolve

to the desired location of the void elimination feature. The first tested location puts the desired location of the void elimination features in the middle of the beam for the x- and y-axis and at the two edges for the z-axis. The second tested location is the same for the y- and z-axis, but changes it for the x-axis so that one feature is located at the base of the beam and the other feature is located at the end of the beam. As seen in the previous section the computational effort can be cut down without heavily interfering with the void pathways by locking both the y- and z-direction, therefore these two tests only have the x-direction free.

The result and problem description of the first test location for an eigenfrequency constraint of $\omega^2_{cons} = 10^{8.5}$ is shown in figure 3.21, which shows that the location of the void elimination feature can be directly controlled by changing the boundary conditions. However with regards to the size of this feature the second test provides more insight. As expected the compliance increased slightly, which does mean that it is recommendable to leave the feature location to the optimizer when possible.

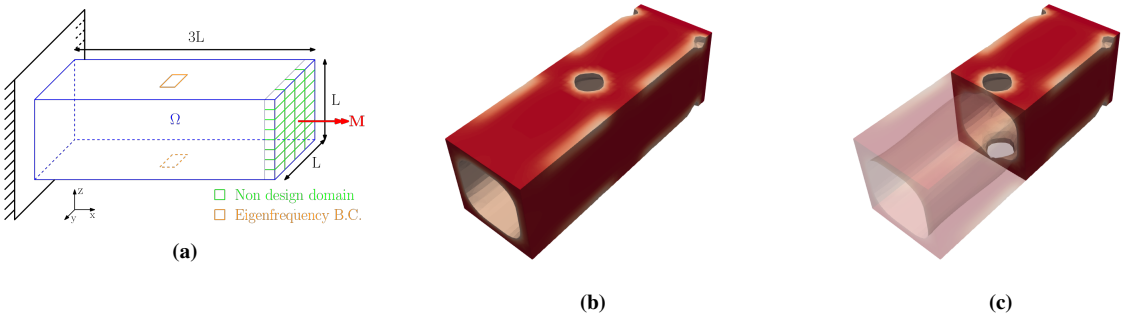


Figure 3.21: Torsion beam subdivided into 48x16x16 elements with hole boundaries at same location along x-axis and $\frac{c}{c_{ref}} = 105.0\%$. (a) Problem description with $L = 0.1$m; (b) Regular view and (c) view cut at hole location.

The second test, depicted in Figure 3.22 shows similar results to the first test, as the optimizer deals well with the different boundary conditions and creates two holes to eliminate the enclosed void. However the hole is larger near the base where it has less impact on compliance, for this reason the compliance is slightly lower. This could either be seen as an advantage, where locations with little impact on the compliance automatically will carry more of the enclosed void elimination duties. On the other hand this could also be seen as a disadvantage, because hole sizes are difficult to control directly.

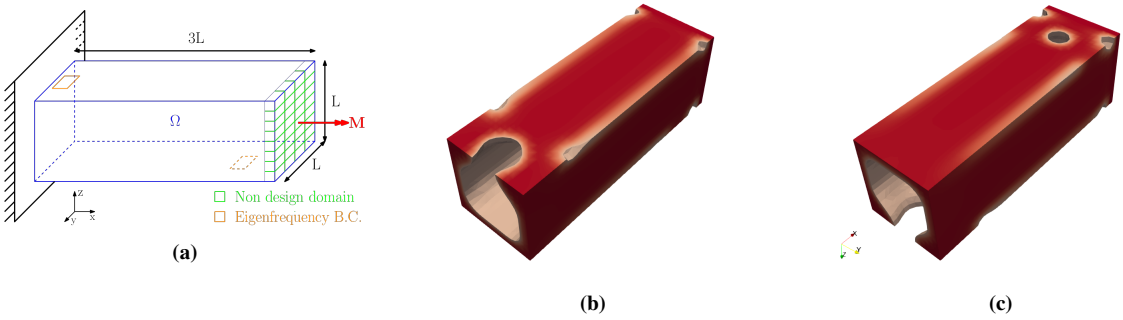


Figure 3.22: Torsion beam subdivided into 48x16x16 elements with hole boundaries at different locations along x-axis and $\frac{c}{c_{ref}} = 103.8\%$. (a) Problem description with $L = 0.1$m; (b) Regular view and (c) view of beam flipped 180° around x-axis.

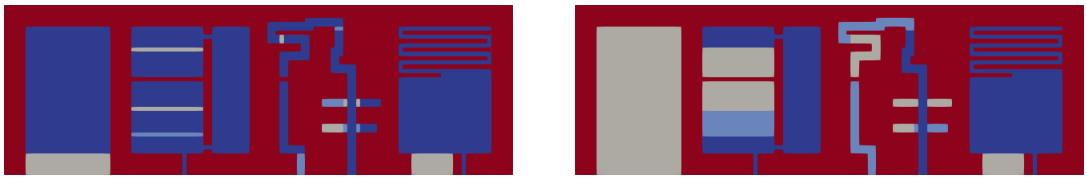
4 | Flood-fill Method

4.1 Working principle

The second proposed method is a more geometrical approach to eliminate enclosed voids, it utilizes a modified flood-fill algorithm to fill up any void in the design. A flood fill algorithm is commonly used for image editing purposes to change certain attributes for connected areas of pixels, most notable example of which is the 'bucket' tool. The unaltered algorithm works by starting at a pixel with a specific target attribute that should be changed and then checking neighboring pixels for this same attribute and subsequently changing it to the desired state. These new pixels are then added to a queue of pixels to process and are then processed in the same way as the starting pixel, this is then repeated until all pixels in the queue have been processed.

Changing the formulation by replacing pixels with elements would already allow us to identify enclosed voids in any design as demonstrated by Xiong et al. [20], but for full incorporation into density based topology optimization three key alterations are necessary. The first alteration is that the algorithm should take all possible elements into consideration, instead of only running the algorithm until no elements of the target attribute remain. This leads us to the next alteration which is that instead of changing these new elements to a prescribed value, they are changed by a performing a smooth maximum with the element currently being processed. The final alteration changes the order in which elements are processed, where the element with the lowest density is always processed first.

The effect of this algorithm on the density field is that any enclosed void element gets its density increased to a value which corresponds with the lowest density path towards it from the prescribed boundary. Figure 4.1 is a visual representation of how the flood fill algorithm affects an input density field with starting elements at the boundary, although in this case a true maximum function is utilized. The test clearly demonstrates how the algorithm fills up any void disconnected from the boundary and how it is able to deal with any complexity in void elimination feature.



(a) **(b)**

Figure 4.1: Visual representation of the flood fill working principle (a) input design and (b) output design.

The previously discussed working principle of this method can then also be converted into a pseudo code in Algorithm 1, which changes input densities ρ into flood filled densities ξ.

Algorithm 1: Flood fill pseudo code

```
Initialize arrays Queue, Result;
Add prescribed boundary elements to Queue and Result;
while Queue ∃ do
    Current element ξ_k = min(Queue);
    Find elements ρ_i adjacent to ξ_k;
    Filter out elements from ρ_i already in Result;
    Change density of elements in ρ_i: ξ_i = smoothmax(ρ_i, ξ_k);
    Add new adjacent elements ξ_i to Queue and Result;
    Remove current element ξ_k from Queue;
end
Return Result
```

4.2 Smooth Maximum Function

The floodfill method has a large dependence on the smooth maximum function chosen to approximate the new element densities, as it is called many times per iteration and any error will have a knock-on effect for the next element. In contrast to the eigenfrequency method the flood fill therefore needs more careful consideration of an aggregation or smooth maximum function. The smooth maximum function is chosen based on two criteria: maximum error and ease of implementation. From literature multiple different smooth max functions were taken and evaluated on these criteria, including the p-norm, KS-norm, two unit softmax functions and multiple more complex variants [30][31][32][33][34].

The more complex variants use splines [31][32], large equations [33] or additional optimization algorithms [34] to create a smooth maximum function, which do not satisfy the requirement for a function that is easy to implement. Therefore these complex variants will no longer be considered when considering the maximum error criterion. The remaining smooth maximum functions can be described using Equation 4.1 through 4.4.

$$\text{p-norm} \qquad \Psi_p(x_1,x_2,...,x_n) = \left(\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}} \tag{4.1}$$

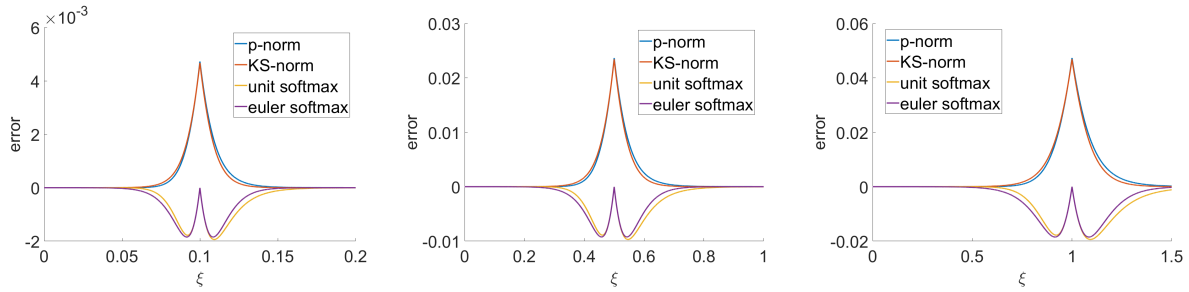
$$\text{KS-function} \qquad \Psi_{KS}(x_1,x_2,...,x_n) = \frac{1}{m}\ln\left(\sum_{i=1}^{n} e^{mx_i}\right) \tag{4.2}$$

$$\text{Unit softmax} \qquad \Psi_{soft}(x_1,x_2,...,x_n) = \sum_{i=1}^{n} x_i \frac{x_i^p}{\sum_{j=1}^{n} x_j^p} \tag{4.3}$$

$$\text{Euler softmax} \qquad \Psi_{softe}(x_1,x_2,...,x_n) = \sum_{i=1}^{n} x_i \frac{e^{wx_i}}{\sum_{j=1}^{n} e^{wx_j}} \tag{4.4}$$

Comparing these functions on their maximum error is best done graphically, as it gives a more complete and clear overview of how these functions behave for different inputs. However the parameters m and w have a different role in the KS-function and Euler softmax than p in p-norm and unit softmax. For the two latter functions a specific value of p changes the relative error of the smooth maximum function, but to achieve this same effect for the former functions m and w would have to scale dynamically with the input variables. Therefore the comparison of the smooth maximum error is done with $p = 15$ for p-norm and unit softmax, but with m and w scaled for the KS-function and Euler softmax. This scaling is done so that the maximum value of the second derivative of the function is the same as their regular counterparts, essentially equalizing the non-linearity of the functions. Furthermore all smooth maximum functions were tested with only two inputs, reflecting how the functions are utilized in the algorithm.

The full comparison of the smooth maximum function is done in Figure 4.2. The first observation that can be made is that the softmax functions have zero error when the two element densities are the same and underestimate the true



(a) $\rho = 0.1$, $m = 148.2$, $w = 150.7$ **(b)** $\rho = 0.5$, $m = 29.6$, $w = 30.15$ **(c)** $\rho = 1$, $m = 14.8$, $w = 15.05$

Figure 4.2: Errors of smooth maximum function where specific values of ρ are compared with ξ.

maximum for slightly different densities. The p-norm and KS-norm are quite the opposite of this, as their error is largest when both densities are the same and they overestimate the true maximum. Furthermore from this comparison it becomes clear that there is very little difference between the smooth maximum functions with the scaled m and w. But scaling them required additional calculations just to determine their values, making it unnecessarily complex. Therefore the main two smooth maximum functions to be investigated for their viability in this method are p-norm and the unit softmax. Both these functions can be converted to how they will be used in practice, where they only use two inputs. Then the new density of the element ξ_i can be calculated using equations 4.5 when p-norm is used and 4.6 when unit softmax is used.

$$\xi_i = \left(\rho_i^p + \xi_k^p\right)^{\frac{1}{p}} \tag{4.5}$$

$$\xi_i = \rho_i \frac{\rho_i^p}{\rho_i^p + \xi_k^p} + \xi_k \frac{\xi_k^p}{\rho_i^p + \xi_k^p} \tag{4.6}$$

Here ρ_i denotes the new element to be altered, ξ_k denotes the element currently being processed.

4.3 Implementation Methods

The flood fill algorithm is currently written as a filter, this opens up multiple ways to actually implement the method. In this case three different options will be considered: only using it as a filter, only using the filtered densities for the volume constraint and adding a geometric constraint to limit the difference between the filtered and regular densities.

Only using the algorithm as a filter is the simplest option, but in its simplicity could be too easy for the optimizer to circumvent. For instance by again utilizing intermediate densities like with the eigenfrequency method.

Only using the filtered densities in the volume constraint mimics the main issue with enclosed voids in powder based methods. It basically means that any enclosed void filled up by the flood fill only adds unnecessary mass to the result, while offering nothing in terms of extra stiffness. This will at least force the optimizer to fill up any enclosed void with material, although once again intermediate densities could still be preferable for it. Furthermore this implementation only works when adding more mass always results in a better objective, as is the case with compliance. In optimization problems where this is not necessarily the case, for example an eigenfrequency maximization, this implementation could be abused by the optimizer to take more material away than allowed.

Finally adding a geometric constraint between the filtered and regular densities is the strictest and most controllable option of the three. It works by imposing a maximum total amount of elements that the regular and filtered density field are allowed to differ. Just as in the geometric optimization in Section 3.5.1 one way to implement this could be the least squares error, however this means that intermediate differences between elements are unjustly favoured because of the quadratic term. One option could be to once again add additional penalization in the form of a square root, but this would cause a discontinuity around zero. Therefore a smooth approximation of the absolute value is needed, where the most important property of the approximation is that there should be no error when the difference is zero. For that reason a slightly altered softmax formulation is implemented, shown in the following equation:

$$s_A = x\frac{e^{px}}{e^{px} + e^{-px}} - x\frac{e^{-px}}{e^{px} + e^{-px}} \tag{4.7}$$

Plots for different values of p, as well as sensitivity analysis on this smooth absolute value can be found in Appendix B.1. For all results in this thesis a value of $p = 15$ is used. The main issue in this implementation arises from the smooth maximum function used for the flood fill, which adds an unpredictable error to the filtered density field.

4.4 Problem formulation

4.4.1 Optimization Problem

With all the different implementation methods known it is now possible to formulate the different optimization problems with them in Table 4.1.

Filter	Volume constraint	Geometric constraint
$\min_{\rho\in\Omega}\ \mathbf{f}^T\mathbf{u}$	$\min_{\rho\in\Omega}\ \mathbf{f}^T\mathbf{u}$	$\min_{\rho\in\Omega}\ \mathbf{f}^T\mathbf{u}$
s.t. $\mathbf{K}(\xi_e)\mathbf{u} = \mathbf{f}$	s.t. $\mathbf{K}(\rho_e)\mathbf{u} = \mathbf{f}$	s.t. $\mathbf{K}(\rho_e)\mathbf{u} = \mathbf{f}$
$\frac{1}{V_{cons}}\sum_{e\in\Omega}(\xi_e v_e) - 1 \leq 0$	$\frac{1}{V_{cons}}\sum_{e\in\Omega}(\xi_e v_e) - 1 \leq 0$	$\frac{1}{V_{cons}}\sum_{e\in\Omega}(\rho_e v_e) - 1 \leq 0$
$0 < \rho_{min} \leq \rho_e \leq 1$	$0 < \rho_{min} \leq \rho_e \leq 1$	$0 < \rho_{min} \leq \rho_e \leq 1$
		$\sum_{e\in\Omega}(s_A(\rho_e - \xi_e)) - \chi \leq 0$

Table 4.1: Optimization problem for all three implementation methods

The optimization problem for the flood fill method is largely the same as for the eigenfrequency method in Section 3.2.1, with a few key differences. The first difference is that, depending on which implementation is used, the input used for certain constraints is the flood filled density field $\boldsymbol{\xi}$ rather than the regular density field $\boldsymbol{\rho}$. Furthermore the eigenfrequency constraint is no longer utilized in this method, which in the case of the geometric constraint implementation is replaced by a different constraint on the difference between the regular and flood filled density fields. In this constraint the total amount of elements the two density fields are allowed to differ is denoted with χ.

The other additions to the optimization are unaltered, which means that a density filter [25] with a radius of two elements is still used, as well as MMA [10] and a density based SIMP approach.

4.4.2 Sensitivity Analysis

As seen above either the objective or one of the constraints of the optimization depends on the flood filled density field $\boldsymbol{\xi}$, which in turn depends on the actual design variables $\boldsymbol{\rho}$. The sensitivities of the objective or constraint to the flood filled density field are known, but to the actual design variables are still unknown. These can be determined using the chain rule, which is depicted for any response f, such as compliance or volume, in the following equation:

$$\frac{\partial f}{\partial \boldsymbol{\rho}} = \frac{\partial f}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\rho}}. \tag{4.8}$$

In this equation the unknown dependence of the response on the design variables is depicted by $\frac{\partial f}{\partial \boldsymbol{\rho}}$. As noted before the dependence of the response on the flood filled density field $\frac{\partial f}{\partial \boldsymbol{\xi}}$ is already known. The key unknown here is the jacobian matrix describing dependence of the flood filled density field on the design variables $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\rho}}$. This matrix for n elements is set up as follows:

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\rho}} = \begin{bmatrix} \frac{\partial \xi_1}{\partial \rho_1} & \frac{\partial \xi_1}{\partial \rho_2} & \cdots & \frac{\partial \xi_1}{\partial \rho_n} \\ \frac{\partial \xi_2}{\partial \rho_1} & \frac{\partial \xi_2}{\partial \rho_2} & \cdots & \frac{\partial \xi_2}{\partial \rho_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \xi_n}{\partial \rho_1} & \frac{\partial \xi_n}{\partial \rho_2} & \cdots & \frac{\partial \xi_n}{\partial \rho_n} \end{bmatrix}. \tag{4.9}$$

Any row of this matrix contains the dependence of one flood filled element on the input density field. This can be calculated directly and determined quite trivially with the fact that the density of a new element added to the queue

only depends on the element currently being processed and its own density. From there a general formula for the sensitivities of any flood filled element ξ_i can be determined as shown the following equation:

$$\frac{\partial \xi_i}{\partial \boldsymbol{\rho}} = \frac{\partial \xi_i}{\partial \rho_i} + \frac{\partial \xi_i}{\partial \xi_k} \frac{\partial \xi_k}{\partial \boldsymbol{\rho}}. \tag{4.10}$$

Here the term $\frac{\partial \xi_k}{\partial \rho}$ is the key term, as it shows that sensitivities of the current element are carried over to the new element. This means that the order of the flood fill is indirectly saved in the sensitivities. When the order of the flood fill changes, which is prone to happen in early iterations, the sensitivities are no longer correct for the subsequent iteration. Another property is that if the sensitivities for any new element contain the sensitivities for all elements flood filled before it, the matrix is densely packed. For larger meshes this could possibly make the sensitivity calculation very expensive.

The terms $\frac{\partial \xi_i}{\partial \rho_i}$ and $\frac{\partial \xi_i}{\partial \xi_k}$ both depend on the smooth maximum function chosen and can be calculated using regular differentiation methods:

$$\frac{\partial \xi_i}{\partial \rho_i} = \rho_i^{p-1} \left(\rho_i^p + \xi_k^p\right)^{\frac{1}{p}-1}, \tag{4.11}$$

$$\frac{\partial \xi_i}{\partial \xi_k} = \xi_k^{p-1} \left(\xi_k^p + \rho_i^p\right)^{\frac{1}{p}-1}. \tag{4.12}$$

$$\frac{\partial \xi_i}{\partial \rho_i} = \frac{\rho_i^p \left(\rho_i^p + p\xi_k^p + \xi_k^p\right) - p\xi_k^{p+1}\rho_i^{p-1}}{\left(\rho_i^p + \xi_k^p\right)^2}, \tag{4.13}$$

$$\frac{\partial \xi_i}{\partial \xi_k} = \frac{\xi_k^p \left(\xi_k^p + p\rho_i^p + \rho_i^p\right) - p\rho_i^{p+1}\xi_k^{p-1}}{\left(\rho_i^p + \xi_k^p\right)^2}. \tag{4.14}$$

Here Equations 4.11 and 4.12 contain the derivatives of the p-norm from Equation 4.5 with respect to ρ_i and ξ_k, while Equations 4.13 and 4.14 contain the derivatives of the unit softmax function from Equation 4.6 with respect to ρ_i and ξ_k.

4.5 Feature control

The way the method is currently set up, an access channel of only one element wide is necessary for a void to not get filled up. This is a major issue when extremely fine meshes are considered, because then one element may not be wide enough to properly remove trapped powder. Therefore some way to control the size of the access channel needs to be added to the method, for which there are three candidates.

The first candidate is applying the floodfill to a coarser mesh, which would have the added benefit of also reducing computational effort. However multiple drawbacks could also be identified. The coarser mesh would reduce the geometric complexity and could be difficult to implement depending on the mesh. Additionally the density of a coarse element would need to be determined, either by a maximizing, averaging or minimizing function, all of which will have their drawbacks.

Expanding the window of the flood fill to consider all elements within a radius is the second candidate. However it has two large drawbacks, the first of which is that the error of the smooth maximum increases when more elements are considered at the same time. The second drawback is that this expansion would also unnecessarily fill up low density elements near edges.

Finally the morphologic operators, as proposed by Sigmund [35], would utilize a dilation of the density field before the flood fill and a erosion after. The dilation operation changes an element that has any solid element within a predefined radius around it to also be solid, whereas erosion does the opposite and changes an element with any void elements in that radius to also be void. The effect of these two operations on a given input design can be seen in Figure 4.3. Applying a dilation before an erosion is essentially adding a minimum feature size on void areas, which also means that any void elimination channel must at least be larger than the prescribed range of the morphologic operators. This method does use a smooth approximation of the maximum and minimum function, which will cause some slight errors in the output densities.

From the three candidates the morphologic operators have the least impactful drawback and will therefore be used as a way to control feature size.

(a) (b) (c)

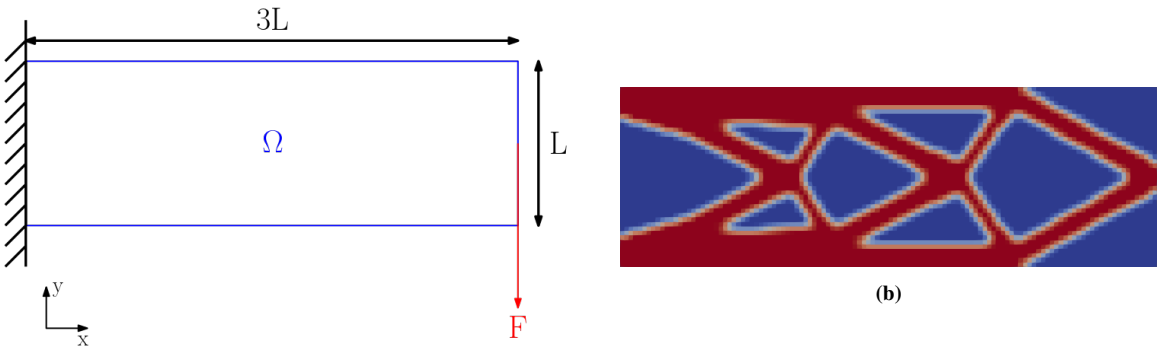
Figure 4.3: The effect of morphologic operators for a given input [35]. (a) Input design; (b) Dilation and (c) Erosion.

4.6 Results

4.6.1 2D Geometric optimization

The first step of testing the floodfill method is, similarly to the eigenfrequency method, to first only consider the geometric optimization of a problem. This again allows us to solely look at how the method handles enclosed voids and analyze possible issues easier. It is important to note that only the filter implementation can be tested with this because the geometric optimization is not compatible with the other two implementations. The geometric optimization will once again be done on the cantilever problem from Chapter 3, for which the problem description and unconstrained result are shown in Figure 4.4 as a reminder. The objective is also once gain described by the least squares error:

$$\min_{\rho \in \Omega} \sum_{i=1}^{\Omega} (\rho_i - \rho_{i,in})^2 . \tag{4.15}$$



(a) (b)

Figure 4.4: 2D cantilever of 120x40 elements with (a) problem description with L = 0.1 and (b) unconstrained optimization with c_{ref} = 100.0.

The first test is to perform the geometric optimization with the difference between the input design and the flood filled densities as objective with the two selected smooth maximum functions.

Unit Softmax

The first smooth maximum function to test is the unit softmax function in Equation 4.6. While it has a smaller error than the p-norm, its underestimation of the true maximum could pose problems. The results of geometric optimization with the unit softmax are shown in Figure 4.5 and it is immediately clear that the suspected problem did indeed occur. The optimizer kept some material in the supposed void elimination channels and then used the underestimating property of the softmax function to simultaneously keep the void areas empty.

P-norm function

With the optimizer taking advantage of the error in the softmax function it is necessary to see if the same occurs for the p-norm, of which the results are shown in Figure 4.6a. It shows how the method successfully creates channels of one element wide every for every enclosed void. It behaves similarly to the shortest path method in this case, where the location and orientation of the channel is simply chosen so that the distance to the boundary is the shortest. The optimizer does use some intermediate densities to avoid the error in the smooth maximum function, because as

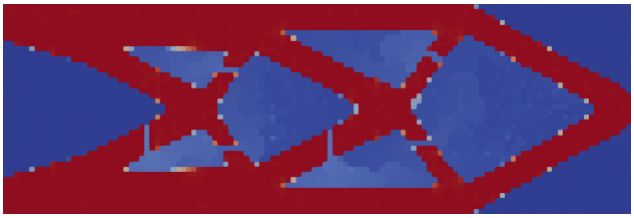
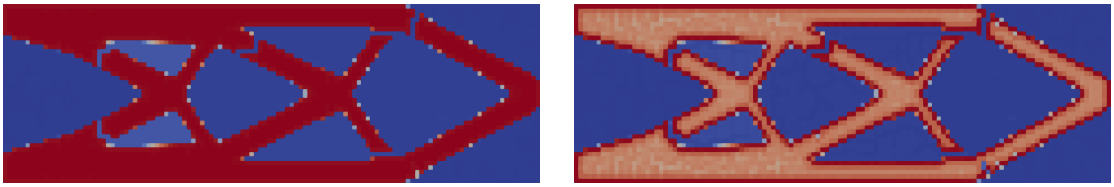


Figure 4.5: Geometric optimization using the unit softmax function with $p = 12$.

shown in Figure 4.2 the error when comparing a density of one and a lower intermediate density is much smaller. This phenomenon is visible using the unfiltered result in Figure 4.6b, where all internal elements have intermediate densities, which then get flood filled to achieve the design in Figure 4.6a.



(a) **(b)**

Figure 4.6: Geometric optimization using p-norm and least squares error with $p = 15$. (a) Filtered densities and (b) unfiltered input densities.

Another observation is that the two smallest enclosed voids have slightly higher density, this is caused by the least squares error used, as the squaring operation unjustly slightly favors element differences below one. This can be easily prevented by using the smooth approximation for the absolute value from Section 4.5, which results in the following objective:

$$\min_{\rho \in \Omega} \sum_{i=1}^{\Omega} s_A \left(\rho_i - \rho_{i,in} \right). \tag{4.16}$$

Results with this formulation are shown in Figure 4.7 and all further geometric optimizations will be done with it as well.

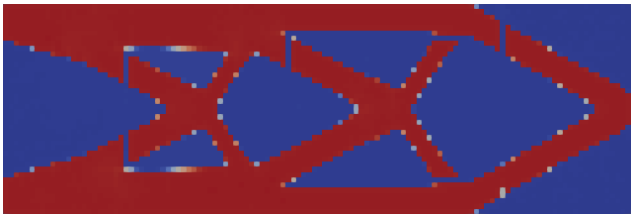


Figure 4.7: Geometric optimization using p-norm and smooth approximation of the absolute value.

The method works well when just the flood fill filter is applied, so now the next step is to add the morphologic operators and see if the method still works. The results with four different morphologic operator radii are shown in Figure 4.8. The first observation is that the use of morphologic operators successfully controls the void elimination feature size. However the smooth approximation used in the morphologic operator does cause some slight errors on elements close to void areas, where increasing the morphologic operator radius also increases the amount of elements included in the error. Increasing the radius also blunts features, most notable with a radius of $r = 3$. Finally an issue this method may encounter is also quite clear for radii $r = 1$ and $r = 1.5$. For these two cases the method opted to create multiple void elimination features through a wide group of elements, as opposed to connecting all void areas together through smaller groups of elements. Larger morphologic operator radii seem to solve this issue, possibly because sensitivities of a specific channel then depend on a larger group of elements around it.

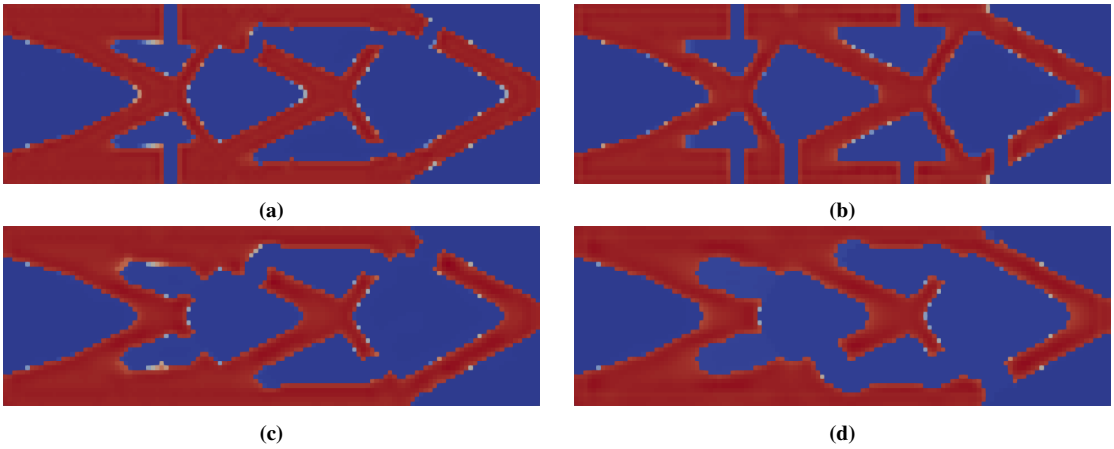


Figure 4.8: Geometric optimization with increasing morphologic operator radius. (a) $r = 1$; (b) $r = 1.5$; (c) $r = 2$ and (d) $r = 3$.

4.6.2 2D cantilever problem

Now that the filter implementation has been confirmed to work well in the geometric optimization the next step is to start studying the method when compliance is considered. For this all three options discussed in Section 4.5 will be studied, with the possibility of also applying the robust formulation when necessary.

Filter

The filter implementation has been found to successfully create void pathways with geometric optimization, but just as the eigenfrequency method this is no guarantee that it works when compliance is considered. For that same reason the unit softmax function will also be considered once again, even though it did not have the desired result in geometric optimization. The first results with the filter implementation when compliance is considered are shown in Figure 4.9 for the unit softmax functions for different values of p. The results show that the error of the unit softmax function has once again been abused by the optimizer to have extra material in between enclosed voids. The unit softmax function has therefore been written off as a possible smooth maximum function and all future optimizations will only use the p-norm.

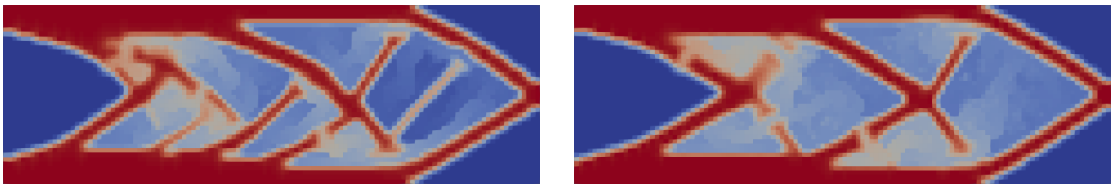
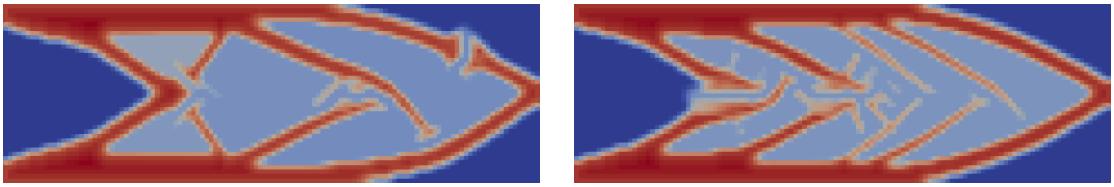


Figure 4.9: Cantilever compliance optimization results for unit softmax with (a) p = 12 and (b) p = 20.

Results when using the p-norm smooth maximum are visible in Figure 4.10, they show how all voids are interconnected and have been filled up with intermediate densities, with one single exit hole in the top right. Elements around the exit hole also have increased width, creating a larger surface area with the intermediate densities of the exit hole and thereby mitigating its effect on compliance.

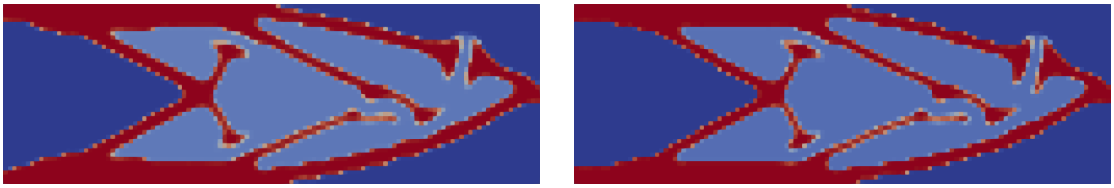
Since the filter implementation encountered the well known problem of intermediate densities, the next step is to try and remedy that with the robust formulation. Results for the robust formulation are shown in Figure 4.11 and they show how, similar to the regular optimization, all void areas are entirely filled up with low density elements. Furthermore the p used in the flood fill is 15 for this and all further results.

Slightly increasing the penalization on intermediate densities to $q = 4$ or $q = 5$ could possibly be enough to push the optimizer to stop using them, extreme penalization like $q = 10$ will not be considered as Chapter 3 has shown it to not be a viable option. Results for slightly increased penalization combined with the robust formulation are visible in Figure 4.12, which both manage to deal with enclosed voids and create the sideways arch design considered to be



(a) (b)

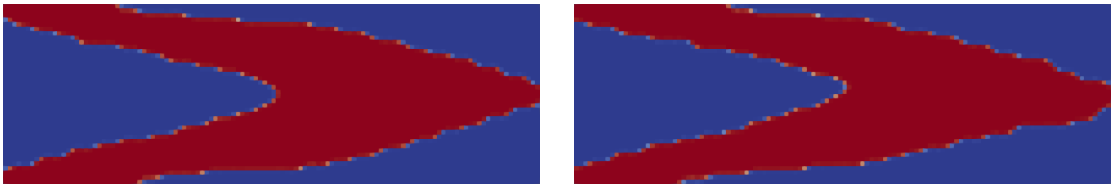
Figure 4.10: Cantilever compliance optimization results for p-norm with (a) p = 12 and (b) p = 20.



(a) (b)

Figure 4.11: Cantilever compliance robust optimization results with different β limits (a) $\beta_{max} = 15$ and (b) $\beta_{max} = 20$.

the optimal solution. This shows that the filter implementation can work when combined with the robust formulation and slightly increased penalization, although converging to this solution took two to four times as many iterations as normal robust optimization. This issue still necessitates additional tests with the other implementations to see if those can reduce the amount of necessary iterations.



(a) (b)

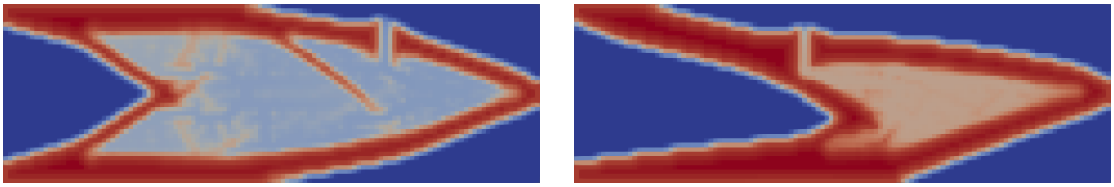
Figure 4.12: Cantilever compliance robust optimization results with β limit 15 and an increased SIMP penalization of (a) $q = 4$ and (b) $q = 5$.

Volume constraint

Results from the filter implementation show how the optimizer opts to fill up the entire void area with intermediate densities when regular penalization is used, which still add some stiffness to the structure. This was remedied using the robust formulation in combination with slightly increased penalization on intermediate densities, at the cost of a large increase in the amount of iterations. The volume implementation aims to avoid the first problem, which would avoid increasing penalization and subsequently the large amount of additional iterations required.

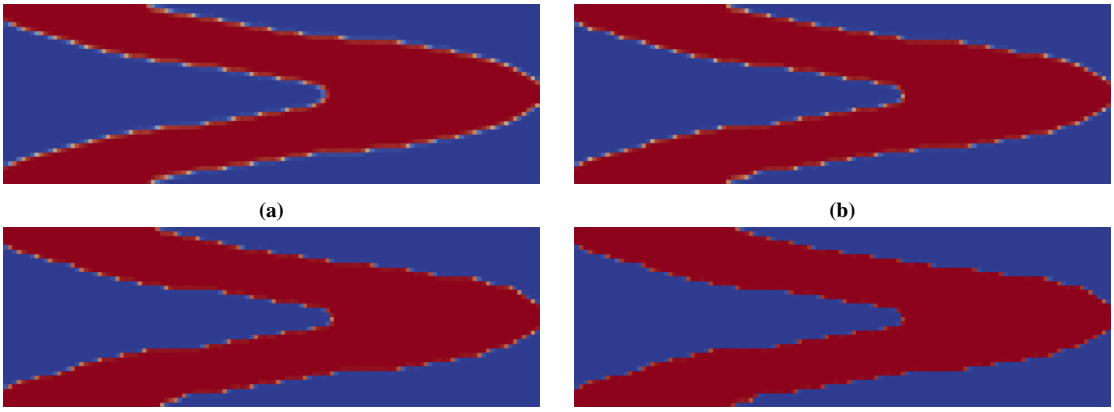
Even though the robust formulation has proven to be a necessity for the filter implementation, the volume constraint implementation is first tested using regular optimization. Results for regular and slightly increased penalization with this implementation are shown in Figure 4.13, where intermediate densities once again fill up all void areas.

The robust formulation should be able to remedy that issue, as seen before with the filter implementation. Here the biggest challenge is to see whether or not additional penalization on intermediate densities is necessary and whether or not that fixes the large iteration counts issue. Results when using the robust formulation are shown in Figure 4.14 with different β limits. It shows that for any normal β limit the volume constraint implementation succeeds in eliminating enclosed voids while generating an optimal shape. As the limit increases the compliance slightly decreases, this can be attributed to the amount of intermediate densities being lower for higher limits. The main problem of the filter implementation has also been remedied, as the amount of iterations required is no longer any higher than unconstrained optimization.



(a) (b)

Figure 4.13: Cantilever compliance optimization results using the volume constraint implementation with varied SIMP penalization. (a) $q = 3$ and (b) $q = 5$.



(a) (b)

(c) (d)

Figure 4.14: Cantilever compliance robust optimization results using the volume constraint implementation with different β limits (a) $\beta_{max} = 10$ with $\frac{c}{c_{ref}}$ = 133.2%; (b) $\beta_{max} = 12$ with $\frac{c}{c_{ref}}$ = 132.6%; (c) $\beta_{max} = 15$ with $\frac{c}{c_{ref}}$ = 132.1% and (d) $\beta_{max} = 20$ with $\frac{c}{c_{ref}}$ = 131.5%.

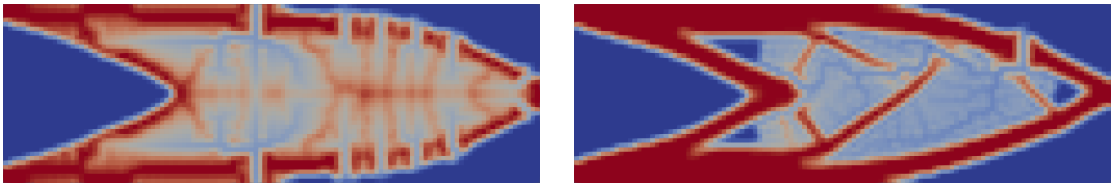
Geometric Constraint

The volume constraint has successfully eliminated enclosed voids while maintaining an optimal design, however because of the reasons stated in Section 4.5 the geometric constraint should also be tested. The first tests will once again first be on regular optimization, even though both the filter and volume constraint implementation the robust formulation was necessary. The main challenge of this method with regular optimization stems from the combination of the smooth maximum error and geometric constraint, because the error makes the difference between the flood filled and regular density field hard to predict. The total error depends on multiple different factors which differ per problem, most notable of which are mesh size and member width. A constraint value that is too strict would therefore cause the optimization to only be done to minimize the error between the two density fields, while a loose constraint value would allow enclosed voids to be generated.

Results with regular optimization are shown in Figure 4.15 for different constraint values. They show that for a low constraint value the optimization is mostly done to minimize the error in the smooth maximum function, while for a higher constraint value the extra leeway also partly goes to this minimization but also to already allow some enclosed void to exist. Most of the void area is again filled with intermediate densities however, which means that extra penalization might be enough to force the optimizer to a interpretable solution.

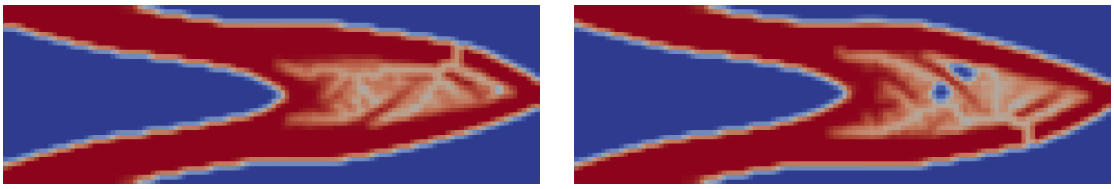
Figure 4.16 shows the geometric constraint with extra penalization on intermediate densities for two similar constraint values. This result does get close to the optimal arch shape, but unfortunately also has a patch of intermediate densities in the middle. The design with the higher constraint value also has a small patch of completely void elements, while the lower constraint design still only utilizes intermediate densities. This means that rather than using the slightly relaxed constraint to fill up the remaining intermediate densities, the optimizer opted to open up an enclosed void instead. This ultimately shows that there is no constraint value which could force the optimizer to completely get rid of enclosed voids, therefore disqualifying this implementation when regular optimization is considered.

The robust formulation has proven itself to be a necessity for the filter and volume constraint implementation and the



(a) (b)

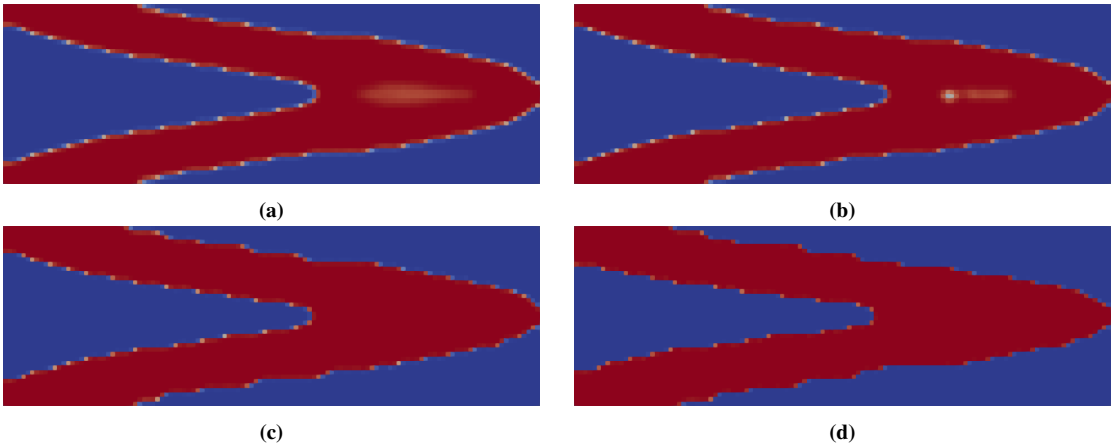
Figure 4.15: Cantilever compliance optimization results using the geometric constraint implementation with varied constraint value. (a) χ = 5 elements and (b) χ = 100 elements.



(a) (b)

Figure 4.16: Cantilever compliance optimization results using the geometric constraint implementation with SIMP penalization increased to $q = 5$ and a varied constraint value. (a) χ = 95 and (b) χ = 100.

geometric constraint is no different. It also remedies the issue of the smooth maximum error, as all values above one simply get projected back to one. The constraint value can then be as low as possible to avoid enclosed voids, which in this case could just be one element. Figure 4.17 shows the results using the geometric constraint implementation with different β limits, additionally Figure 4.18 shows what happens when the constraint is relaxed slightly. It shows that the geometric constraint implementation also successfully eliminates enclosed voids when combined with the robust formulation, although for either lower β limits or slightly relaxed constraints some small voids may slip through the cracks.

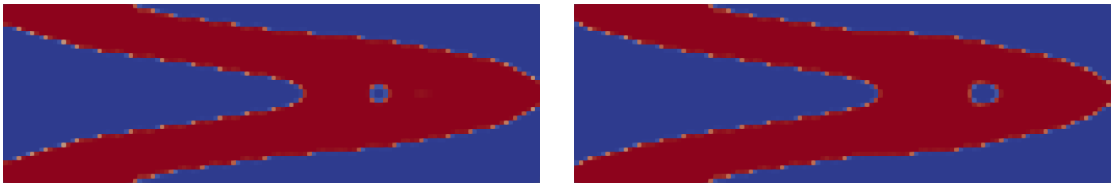


(a) (b)

(c) (d)

Figure 4.17: Cantilever compliance robust optimization results using the geometric constraint implementation with different β limits and χ = 1 element. (a) $\beta_{max} = 10$ with $\frac{c}{c_{ref}}$ = 132.9%; (b) $\beta_{max} = 12$ with $\frac{c}{c_{ref}}$ = 132.6%; (c) $\beta_{max} = 15$ with $\frac{c}{c_{ref}}$ = 135.5% and (d) $\beta_{max} = 20$ with $\frac{c}{c_{ref}}$ = 145.6%.

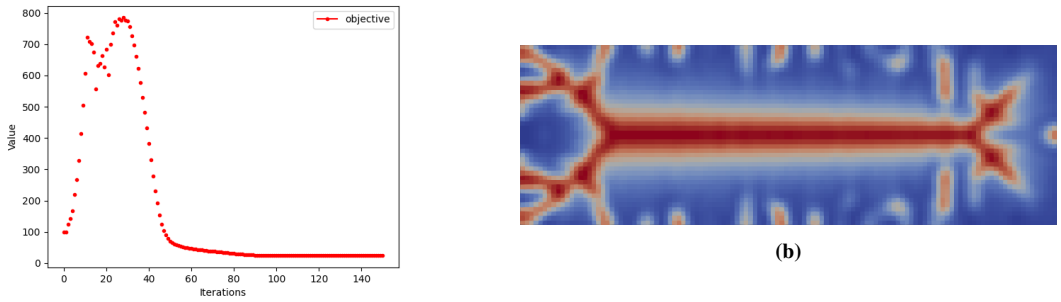
These results do have a small caveat in that they exhibits suboptimal convergence behaviour, as shown in Figure 4.19. An important note for these convergence graphs is that they show an objective scaled to the initial design of uniformly intermediate densities. The suboptimal convergence behaviour is because, as noted before, the robust formulation uses a continuation method on the Heaviside, which means that until β is increased the optimizer is again mostly trying to minimize the error in the smooth max function. Fixing this behaviour can be done by initially relaxing the constraint and slowly tightening it as β increases, however the choice of both the relaxed constraint and how fast it is tightened is not a trivial task. The choice of parameters depends on a combination of the previously described total error in the



(a) **(b)**

Figure 4.18: Cantilever compliance robust optimization results using the geometric constraint implementation with a β limit of 15 and different constraint values (a) $\chi = 10$ with $\frac{c}{c_{ref}} = 130.8\%$ and (b) $\chi = 20$ with $\frac{c}{c_{ref}} = 129.4\%$.

smooth maximum function and the increase in β limit.



(a) **(b)**

Figure 4.19: Convergence behaviour of the geometric constraint implementation. (a) Graph of scaled objective versus iterations and (b) design at peak at iteration 28.

Figure 4.20 shows a result found when the geometric constraint is scaled in too late, when β is already at its limit of 25. Figure 4.21 shows scaling in the constraint too fast is also a problem, although this mostly pertains to how much the constraint changes each iteration. While both these results do somewhat resemble the optimal arch solution, there are multiple imperfections that have been caused by these two ways of scaling in the constraints. Analytically determining a universal formula for tightening the constraint is still difficult, but two general rules to follow can be derived from these results. The first general rule is that the constraint can only become restrictive when β is high enough to curb the error, but not too high that the gradients of densities close to zero and one have already decreased too far. In this case that value lies somewhere between eight and twelve. However what a restrictive constraint means is going to be different for every case and depends on both the mesh size and the problem itself. The second rule is on how much the constraint should change every iteration, which should be as little as possible. The recommendation is then to start the constraint at 10% of the total amount of elements and end it at less than one element, linearly changing the constraint between these two values starting at the first iteration and ending at an iteration in which β is between eight and twelve.

4.6.3 Additional 2D compliance problems

Results from the cantilever tests show how all implementations of the method are able to deal with enclosed voids properly in 2D when combined with the robust formulation. This initial result is promising, although some more tests on different problems are necessary to validate that the method could be applied universally. For this purpose two additional design problems were chosen: a Mitchell cantilever and a table problem. The full problem description and their free-form results can be found in Figure 4.22 for the Mitchell cantilever and in Figure 4.23 for the table. Both these new problems were solved using the exact same settings that were successful for the cantilever problem, which could show that this method is universally applicable.

The results for the Mitchell cantilever are visible in Figure 4.24, which shows how all implementation methods have successfully dealt with enclosed voids and have generated a design which could be considered optimal. The filter result shows a small imperfection with more material on the upper part of the design, this is because during the optimization a void elimination feature was located there. This feature was closed so slowly that the Heaviside already reached its limit, causing the result to get stuck in a slightly suboptimal shape. The result for the geometric constraint also got

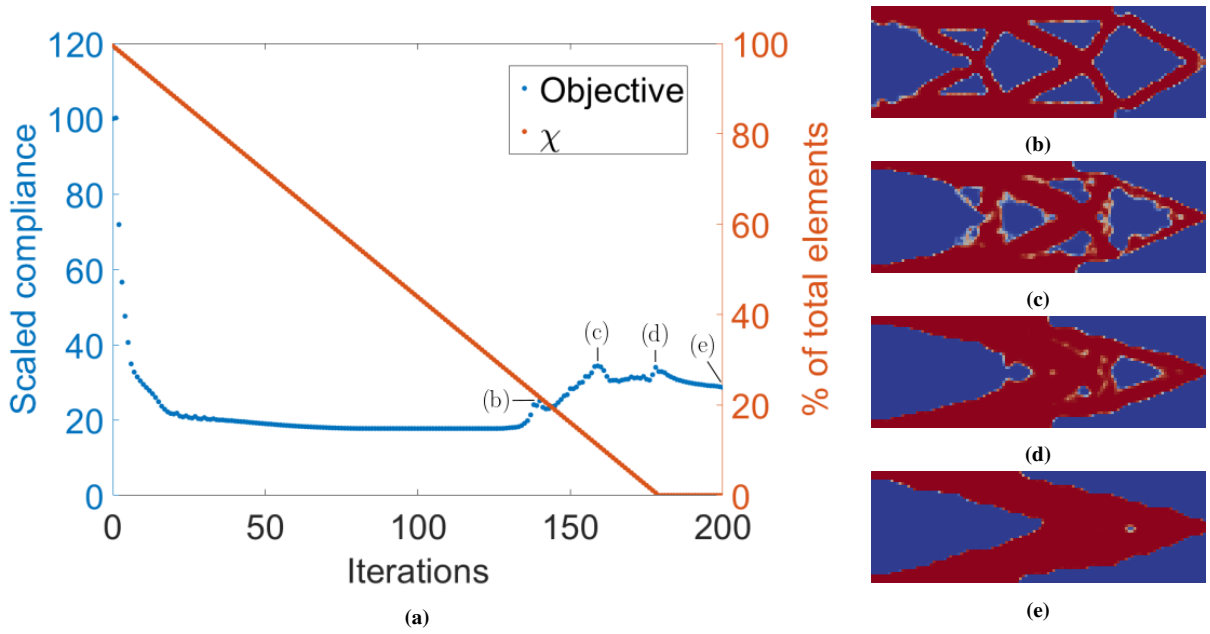


Figure 4.20: Scaling in the geometric constraint too late with graph of objective and designs at key points in graph. (a) Graphed convergence behaviour; (b) iteration 137; (c) iteration 160; (d) iteration 175; (e) iteration 200.

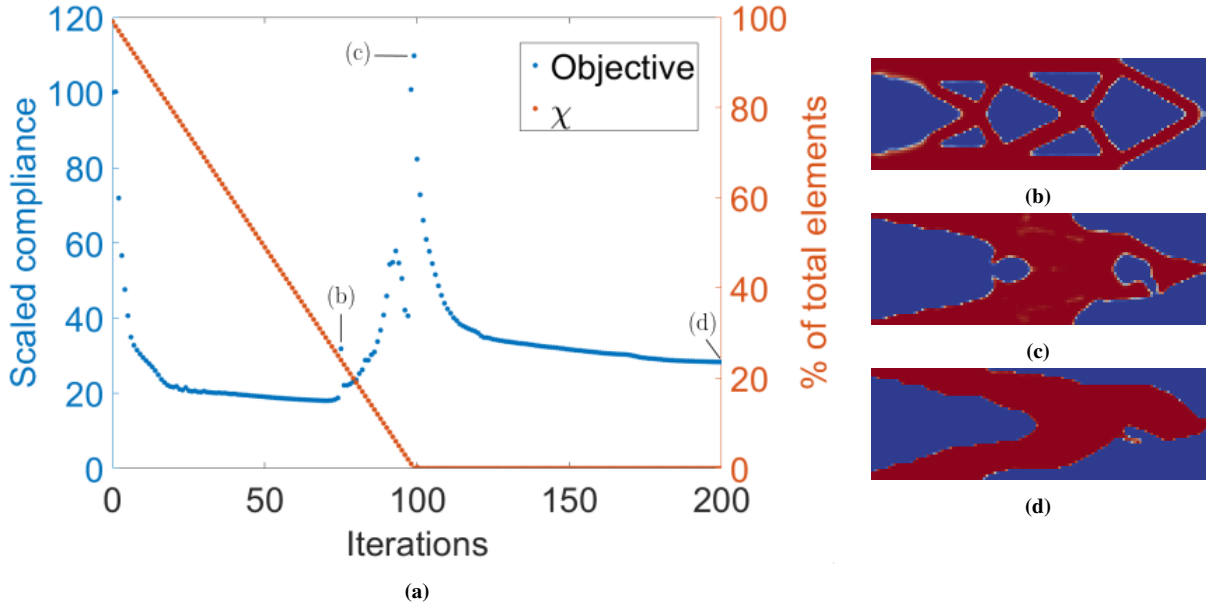
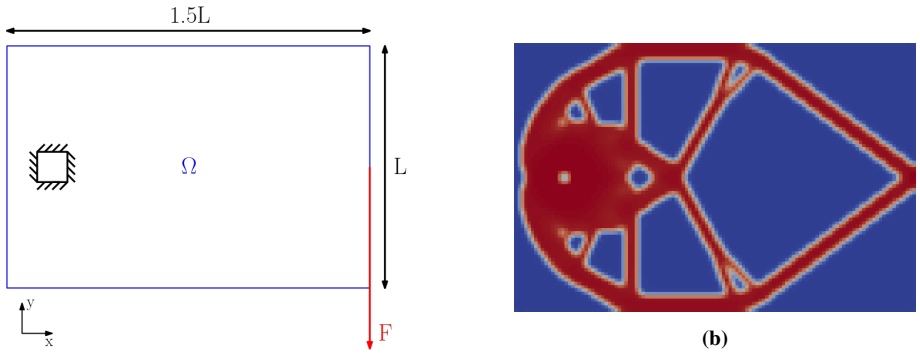


Figure 4.21: Scaling in the geometric constraint too fast with graph of objective and designs at key points in graph. (a) Graphed convergence behaviour; (b) iteration 75; (c) iteration 99; (d) iteration 200.

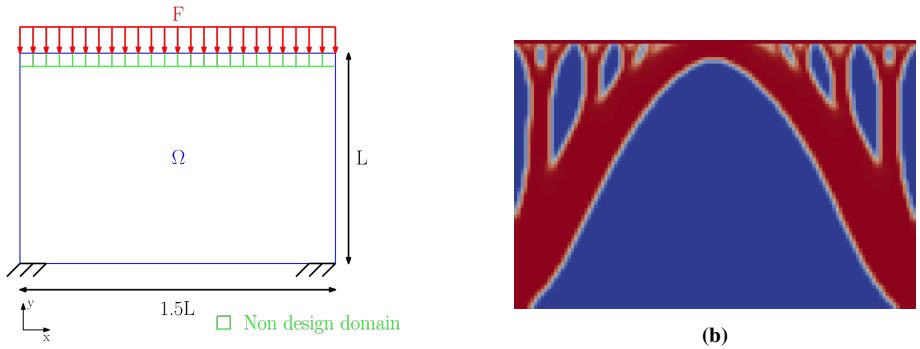
stuck in a suboptimal shape in a similar fashion, however it instead is caused by the previously discussed convergence behaviour.

Results for the table problem for all three implementation methods are shown in Figure 4.25, which shows that the filter and volume constraint implementation have again successfully dealt with enclosed voids and created an optimal design. However, the geometric constraint implementation got stuck in a suboptimal shape, again caused by the convergence behaviour. It shows that without a proper fix for the convergence behaviour the geometric constraint has a tendency to get stuck in shapes similar to its initial shape, which is close to the design shown in Figure 4.19. This behaviour also explains how the slight deviation in shape of the Mitchell cantilever result came to be. Figure 4.26 shows the table



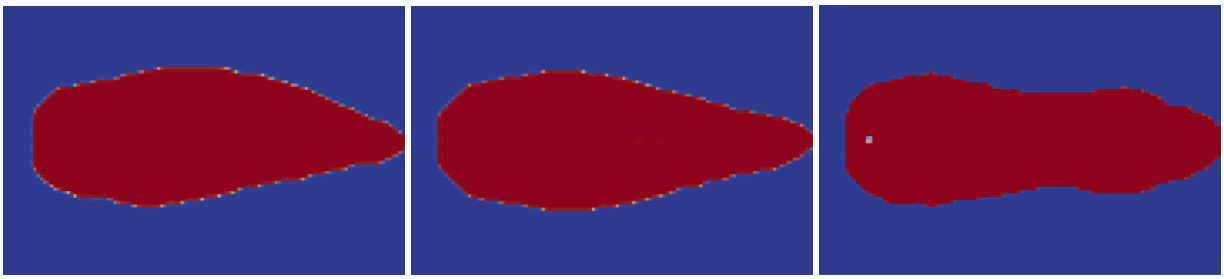
(a) (b)

Figure 4.22: The Mitchell cantilever problem with (a) Problem description and (b) free-form solution for a volume fraction of 0.35 with $c_{ref} = 100$.



(a) (b)

Figure 4.23: The table problem with (a) Problem description and (b) free-form solution for a volume fraction of 0.4 with $c_{ref} = 100$.



(a) (b) (c)

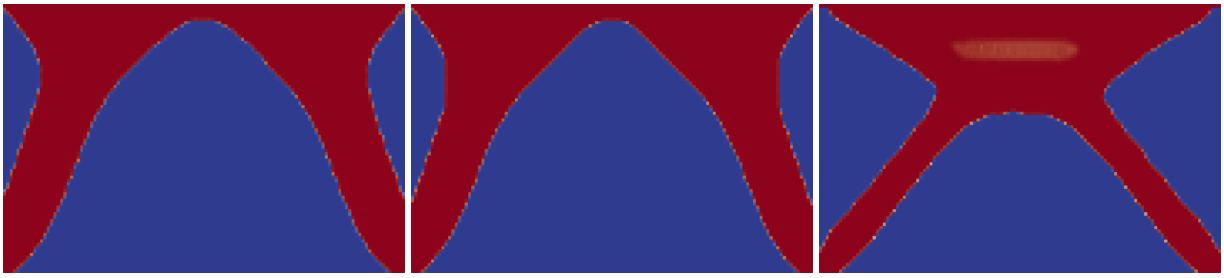
Figure 4.24: Mitchell cantilever results with β limit 15 for different implementations (a) filter implementation with increased penalization and $\frac{c}{c_{ref}} = 116.0\%$; (b) Volume constraint implementation with $\frac{c}{c_{ref}} = 113.5\%$; (c) Geometric constraint implementation with $\chi = 1$ and $\frac{c}{c_{ref}} = 114.6\%$.

results with the scaled constraint, in which this problem is fixed.

4.6.4 3D compliance problem

Initial tests

Since the method has succeeded in eliminating enclosed voids in multiple 2D problems, a 3D case should not pose too much of a problem. This is because, as noted before, there are more directions in which to create void elimination features. Nevertheless it is still good to test the method in 3D both to see if that brings any unexpected problems with



(a) (b) (c)

Figure 4.25: Table results with β limit 15 for different implementations (a) filter implementation with increased penalization and $\frac{c}{c_{ref}} = 106.6\%$; (b) Volume constraint implementation with $\frac{c}{c_{ref}} = 106.4\%$; (c) Geometric constraint implementation with $\chi = 1$ and $\frac{c}{c_{ref}} = 270.1\%$.

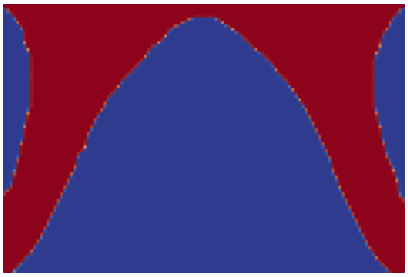
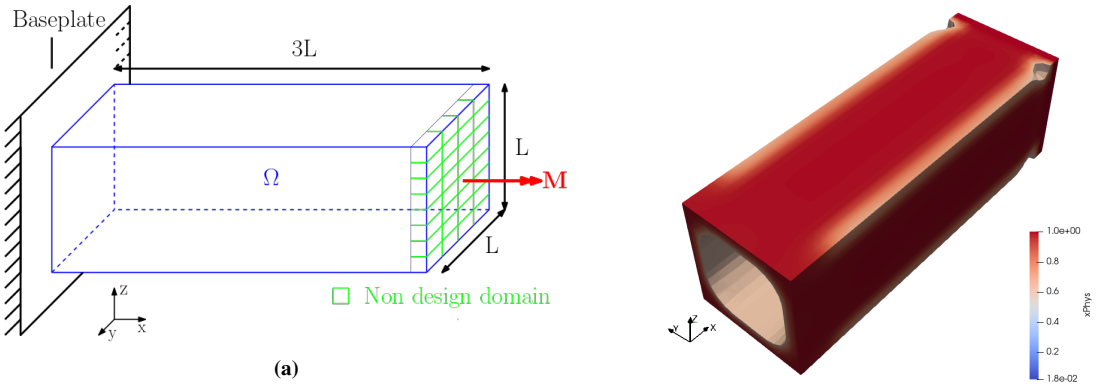


Figure 4.26: Table result with β limit 15 and a scaled geometric constraint with $\frac{c}{c_{ref}} = 106.9\%$.

it and to further test morphologic operators. The 3D test problem will be the torsion beam same as in Chapter 3 and for the same reasons listed there. Its problem description and unconstrained solution are visible in Figure 4.27 as a reminder.

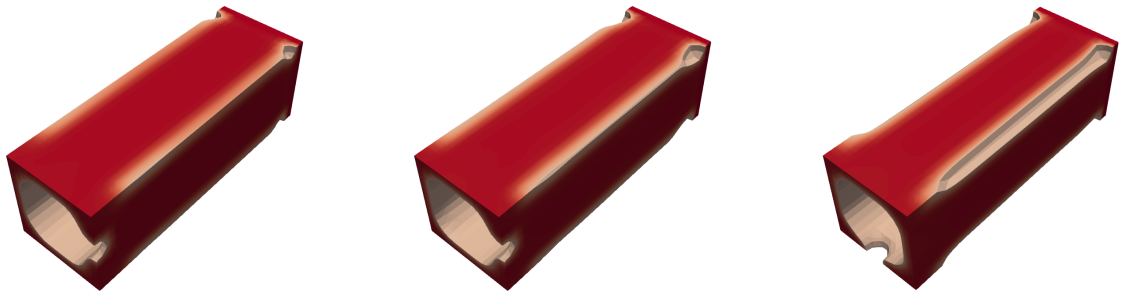


(a) (b)

Figure 4.27: The 3D torsion beam (a) problem description and (b) unconstrained result with a volume fraction of 0.5 and $c_{ref} = 100$.

The results of all three implementation methods are shown in Figure 4.28, which show how all implementations sufficiently dealt with the enclosed void in the middle by making a small hole close to the base. However, the location of this hole is different for the geometric constraint method. This does show a slight problem the method might encounter, where the location of the feature could be unpredictable. This is not an issue in this symmetric problem as the hole locations are essentially the same, but whether or not this problem also occurs in asymmetric problems should still be tested. In terms of compliance all implementations show acceptable results, with the filter implementation even

improving on compliance. This improvement is unfortunately only artificial, as it is caused by the error of the smooth maximum function increasing element densities above one.



(a) (b) (c)

Figure 4.28: Solutions to the torsion beam problem with three flood fill implementations. (a) Filter with $\frac{c}{c_{ref}} = 99.0\%$; (b) Volume constraint with $\frac{c}{c_{ref}} = 100.1\%$; (c) Geometric constraint with $\chi = 50$ and $\frac{c}{c_{ref}} = 102.1\%$.

Morphologic operators

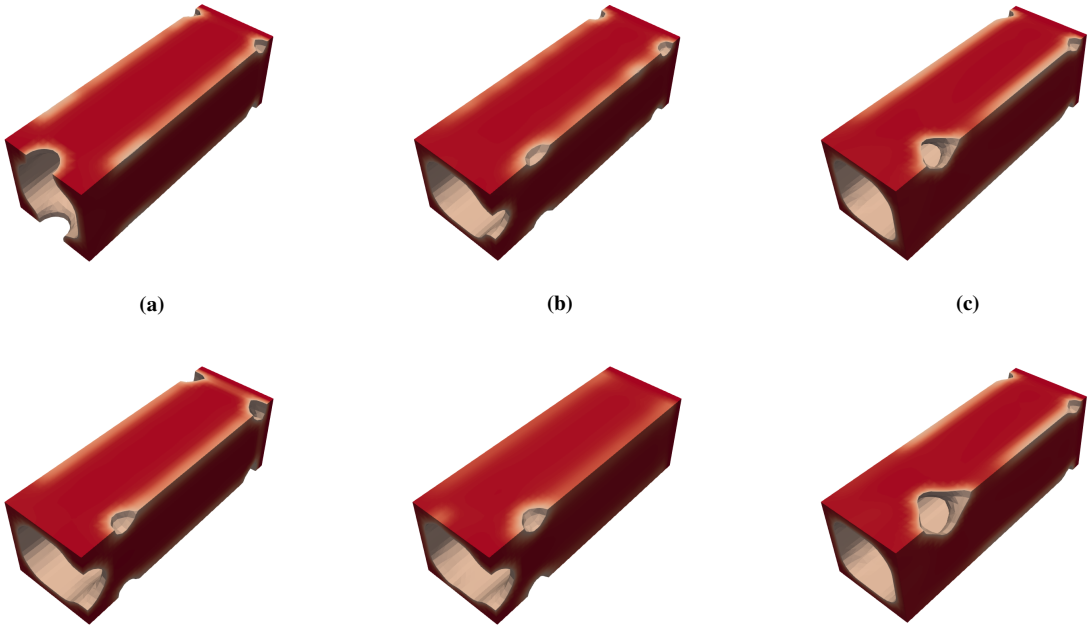
The initial tests show that the method is able to successfully eliminate enclosed voids in a 3D problem, however it currently does so using only an elimination feature of one element wide. Depending on how fine the mesh is this might not be a sufficient width to extract powder from the void, which is why the next step is to also apply the morphologic operators discussed in Section 4.4. The geometric optimization already shows how these should be able to properly achieve control over the void elimination feature size, although testing in 3D and with compliance is still necessary to confirm this.

Results using the filter, volume constraint and geometric constraint implementation with different morphologic operator radii are shown in Figure 4.29. For the volume and geometric constraint implementations the figure does show that morphologic operators can be used to accurately control the size of the void elimination feature, but for the filter implementation the compliance is higher due to the error introduced and an unnecessary void elimination feature has even been generated for a radius of one. This result shows once again one of the main issues of the flood fill algorithm, where it can get stuck in suboptimal void locations and numbers. The geometric constraint did require quite a high constraint value, due to the addition of the error in the smooth maximum of the morphologic operators. This is not a problem in the case of the torsion beam because there is only one large void area, but for other problems where smaller voids exist this could be abused to leave those enclosed.

Feature location control

Now that the size of the void elimination features can be controlled using morphologic operators, the final test is to also accurately control both the amount and locations of the features. These should be tested together as choosing the amount of features inherently also necessitates a choice on different feature locations. To create additional void elimination features the flood fill algorithm needs to be performed multiple times, each time with different boundary elements. This is in contrast to the eigenfrequency method, where a higher constraint value automatically created more features. The advantage of this is that there should be complete control over the amount, size and location of the void elimination features, this does however come at the cost of additional computational effort. Performing the flood fill multiple times in a row also opens up another decision on how that is best performed, as they can either be done in series or parallel. Performing the flood fill in series would mean that every subsequent flood fill takes the output of the previous one as input, which has a knock-on effect on the total error. Performing the flood fill in parallel could avoid the knock-on effect on the error, as all flood fills are performed using the same input. This does mean that additional constraints are necessary for each new density field, which makes it impossible in combination with the filter implementation.

Figure 4.30 shows the results of multiple flood fills for the filter, volume and geometric constraint implementation, where for the filter the flood fill is done in series but for the volume and geometric constraint it is done in parallel.

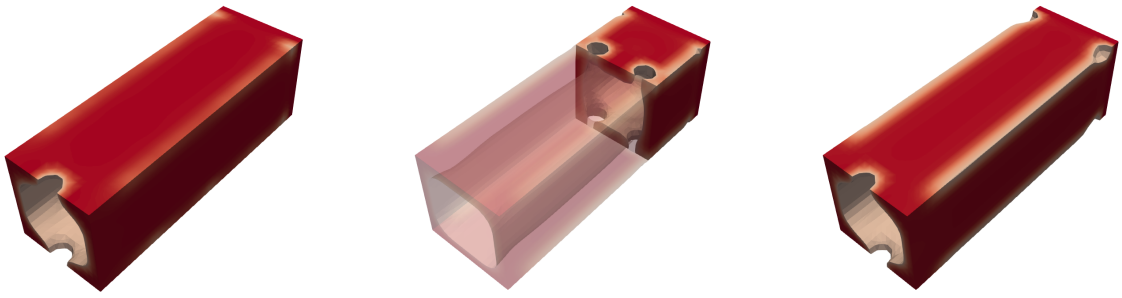


(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 4.29: Solutions to the torsion beam problem for different implementations and morphologic operator radii. (a) Filter implementation with r = 1 and $\frac{c}{c_{ref}}$ = 105.4%; (b) Volume constraint implementation with r = 1 and $\frac{c}{c_{ref}}$ = 100.6%; Geometric constraint implementation with χ = 1000, r = 1 and $\frac{c}{c_{ref}}$ = 101.1%; (d) Filter implementation with r = 2 and $\frac{c}{c_{ref}}$ = 110.4%; (e) Volume constraint implementation with r = 2 and $\frac{c}{c_{ref}}$ = 104.5%; (f) Geometric constraint implementation with χ = 1000, r = 2 and $\frac{c}{c_{ref}}$ = 102.6%.

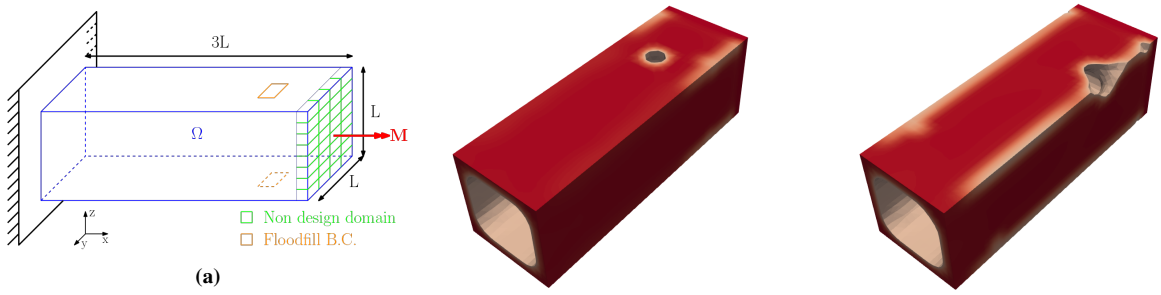
The results for all beams show how the method can succesfully create multiple void elimination channels, although the location of these channels has become less predictable. Furthermore the result for the volume constraint even contains two redundant void elimination channels, although it has not resulted in a large increase in compliance. The compliance for the filter implementation should once again be taken with a grain of salt, as it is now the result of two flood fills being performed in a row.



(a) **(b)** **(c)**

Figure 4.30: Solutions to the torsion beam problem where multiple void elimination channels are required with three flood fill implementations. (a) Filter with $\frac{c}{c_{ref}}$ = 96.6%; (b) Volume constraint with $\frac{c}{c_{ref}}$ = 103.4%; (c) Geometric constraint with χ = 200 and $\frac{c}{c_{ref}}$ = 100.6%.

Finally the method was also tested with two additional locations for the boundary elements, one of two specific hole locations and one of two surfaces with some overlapping elements. These were both only tested using the volume constraint implementation, because throughout all tests it performed the best. The results for both tests are shown in Figure 4.31 along with the problem description of the specific hole locations, which shows how the method is at least able to deal with very specific hole locations. The surfaces with overlapping elements do show how some precaution should be taken when prescribing boundary elements, as the optimizer has created one single hole connected to the overlapping elements.



(a) **(b)** **(c)**

Figure 4.31: Additional boundary element cases, results using the volume constraint implementation. (a) Problem description of torsion beam with specific boundary locations; (b) Result for specific boundary locations with $\frac{c}{c_{ref}} = 102.6\%$; (c) Result for boundary surfaces with overlapping elements with $\frac{c}{c_{ref}} = 101.8\%$.

5 | Discussion

5.1 Computational effort

How much additional computational effort the new methods cost for the test cases is known, but how that additional effort scales with the amount of elements or degrees of freedom should still be tested. It is also necessary to test how different settings, for instance locking the degrees of freedom discussed in Section 3.1 or the morphologic operators discussed in Section 4.5, affect this additional effort. The five tests then become the eigenfrequency method with and without a locked direction, the flood fill method with and without morphologic operators and finally determination of the compliance. The eigenfrequency method includes three notable operations, the assembly of the stiffness and mass matrix, as well as the generalized eigenvalue problem. Determination of the compliance includes two notable operations, the assembly of the stiffness matrix and a linear solve to determine the displacement vector.

By testing the algorithm on a cube with different mesh sizes, the computational effort can be visualized in Figure 5.1. They show that for lower amount of degrees of freedom all tested methods require similar amounts of computational effort, with the eigenfrequency method slightly more expensive than the objective calculation, which in turn is slightly more expensive than the flood fill. That the flood fill initially requires a similar amount of computational effort can be explained by the fact that this program is written in an interpreted coding language, but the linear solve and eigensolve both utilize extensions which have been written in a compiled language. Interpreted languages are known to be slower than compiled languages and at a low amount of degrees of freedom this has a more significant impact than the computational complexity of the methods. Increasing the amount of degrees of freedom is when the computational complexity more significantly influences the computational effort required. The flood fill is a simple loop which should scale linearly ($\mathcal{O}(n)$) with the amount of elements and therefore it has a large advantage over the eigenfrequency method or the linear solve for larger meshes. The theoretical estimation of the computational complexity depends on the chosen technique and is between $\mathcal{O}(n\log(n))$ and $\mathcal{O}(n^3)$ for a linear solve and between $\mathcal{O}(n^{2.4})$ and $\mathcal{O}(n^3)$ for an eigenvalue determination. These computational complexities clearly reiterate how the flood fill method is comparatively cheaper the larger the mesh becomes.

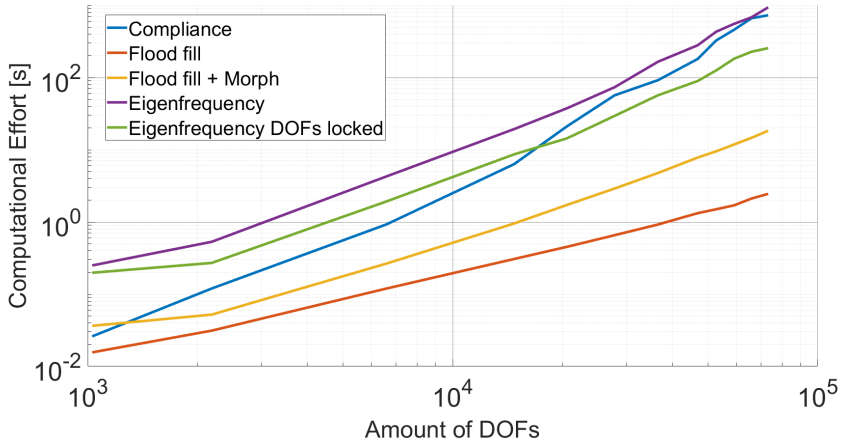


Figure 5.1: Computational effort for a cube of increasing size.

These results do have a small caveat though, as the calculation of the sensitivities was left out. This calculation is an expensive part of the flood fill method in the current direct approach and therefore the true computational complexity of the flood fill method is slightly higher than $\mathcal{O}(n)$. Figure 5.2 compares the flood fill method with and without sensitivity calculation, which shows that the computational complexity is indeed increased to approximately $\mathcal{O}(n^{1.5})$ when the sensitivity calculation is taken into account. An adjoint formulation of the sensitivity could possibly improve this complexity, although it would come at the cost of having to store the order of processing and subsequently running an additional loop.

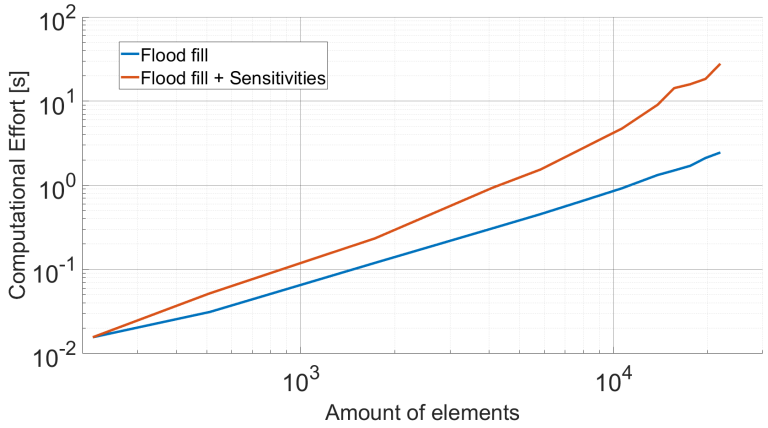


Figure 5.2: Computational effort of the flood fill method with and without sensitivities.

5.2 Overshoot error

As seen in Chapter 4, one of the biggest drawbacks of the flood fill algorithm is the error of the smooth maximum function. It artificially inflated the performance of the design in the filter implementation, unnecessarily put additional restrictions on the volume in the volume constraint method and caused suboptimal convergence behaviour for the geometric constraint implementation.

Figure 5.3 shows how this error evolves when multiple elements of the same density are evaluated sequentially, this is to get an idea of how this error is affected by different mesh sizes. It is important to note that this essentially assumes one large line of either void or solid material and that the actual problem has to be several orders of magnitude larger than depicted for that error to occur. This is due to the fact that a normal design has a mix of groups with either void or solid elements and that any boundary between these groups will have an error of zero as seen in Figure 4.2. The first subfigure clearly shows that the absolute error is negligible for void elements, while for solid elements this error can become unacceptably high. The second subfigure shows the error for void elements more accurately and it shows how all errors in the p-norm are the same when scaled with the element densities. How these tested errors relate to a specific mesh size is once again something that is difficult to determine, as it varies for specific load cases, their volume constraints and filter radii.

Since the robust formulation has been used to partly tackle the problem of this error, its cumulative error should also be tested. However, since all density values above one simply get projected back to one, this test is only done on a line of void elements. The result of this test should then show if the error in a void element can become so large that it reaches the point where the Heaviside steps. The amount of elements required for this to happen for any η and p can be determined analytically with the following equation:

$$(n_{el}\,(\rho_{el})^{p})^{\frac{1}{p}} = \eta \tag{5.1}$$

This equation reveals that with the common values of $\eta = 0.5$, $\rho_{el} = 0.1$ and $p = 15$ it would require more than 10^{10} uninterrupted void elements for one element density to reach 0.5. This again shows how the robust formulation can successfully mitigate the error of the smooth maximum function.

Now that this test revealed how the error on void elements has negligible impact, a final test should be done on solid elements. This test could reveal another critical aspect of the robust formulation in combination with the flood fill algorithm, namely how much the total error on a specific amount of solid elements is for different values of β. This

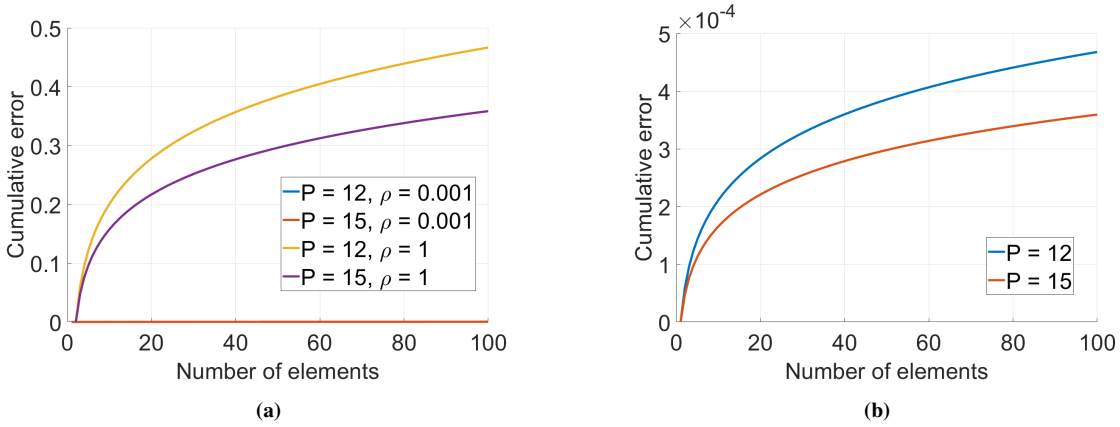


Figure 5.3: Cumulative error for 100 elements of the flood fill algorithm for different element types and values of P. (a) solid and void elements of $\rho = 1$ and $\rho = 0.001$, respectively; (b) Only void elements of $\rho = 0.001$ with different scale on y-axis.

could then be used to determine a specific value of β where this error no longer hampers the convergence of the geometric constraint implementation. For this purpose Figure 5.4 shows the total error of solid elements for different values of β, which shows that for any β past 7 for normal and 9 for very fine meshes the overestimation error is negligible. This result combined with the tests done in Section 4.6.2 allows us to have a better understanding of how the geometric constraint should be applied, mainly that the constraint should be tightened at these β values and not much higher, optionally while pausing the β continuation.

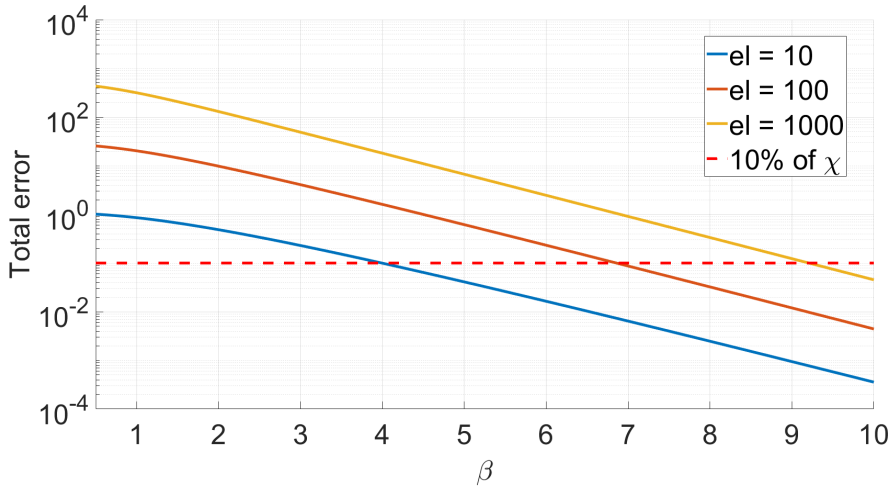


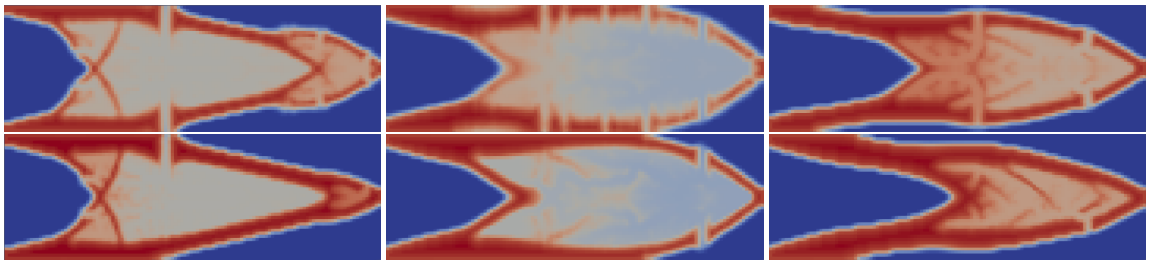
Figure 5.4: Total error of the flood fill algorithm with P = 15 for different values of β and number of solid elements.

5.3 Feature locations

During testing with the flood fill method it became apparent that it has a tendency to get stuck in an initial location of the void elimination channel, even though better locations can clearly be identified. This most notably happened during geometric optimization, where large void channels were sometimes created for every single void as opposed to simply connecting all voids together. This behaviour seemed to be solved when compliance is considered, because most of those results utilize one single void elimination channel. However, another question could then be raised on the location of this channel and whether or not it is a result of the aforementioned tendency or the optimal location. These issues are mainly caused by the discrete order in which elements are processed, which is not included in the sensitivity calculations.

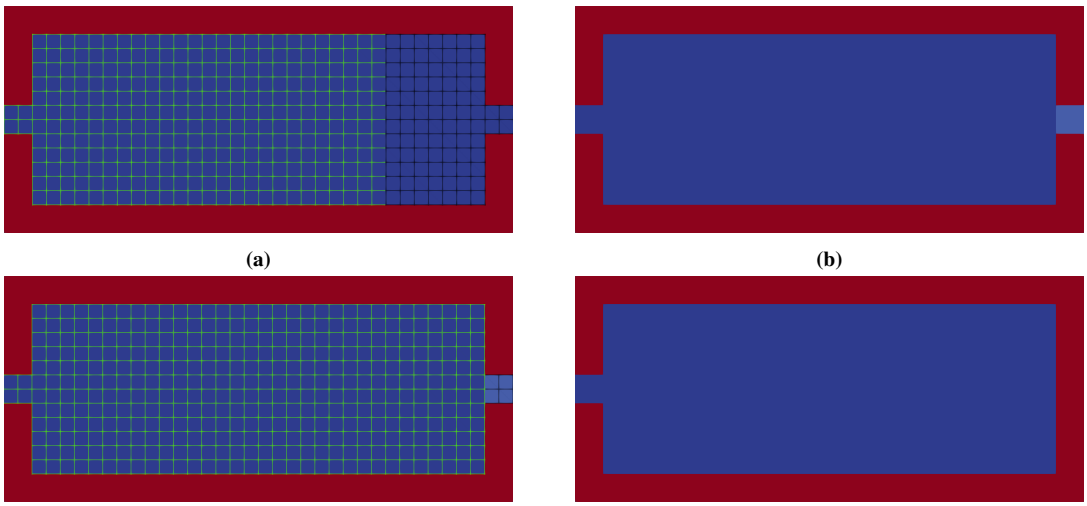
While the final designs offer no insight into this matter, their results in earlier iterations do show how this tendency is less problematic than initially thought. Figure 5.5 shows the early iterations of two results shown in Chapter 4, more

specifically in Figure 4.13, 4.15 and 4.16. These earlier iterations show how the optimizer generally opens up multiple void elimination channels, which are then closed in later iterations. The closing of these channels happens because of a small change in density in one of the channels and its exact working mechanism is visualized schematically in Figure 5.6, these. This phenomenon does show that multiple locations for void elimination channels are considered and that they are later closed due to their negative impact on the compliance, which indicates that the tendency to get stuck is not as prevalent as initially suspected, although more testing is necessary to confirm this.



(a) (b) (c)

Figure 5.5: Previous results show how multiple channels are used in early iterations. (a) Iteration 20 and 50 for volume constraint with increased penalization; (b) Iteration 15 and 23 for geometric constraint of 100 elements; (c) Iteration 25 and 100 for geometric constraint of 100 elements with increased penalization.



(a) (b) (c) (d)

Figure 5.6: Visual representation of the removal of redundant void elimination channels. Outlines indicate which element is influenced by which channel. (a) Initial state where both channels have different reaches of influence, caused by slight variations in density; (b) Next iteration where the channel with less influence increases in density to reduce effect on compliance; (c) Channel with increased density now has zero influence; (d) Channel with zero influence fills up over subsequent iterations as no void elements depend on it anymore.

Unfortunately closing void elimination channels does not always occur, most notably in the 3D results shown in Figure 4.29a and 4.30b. Here the small increase in density which subsequently closes the void elimination channel does not happen, because closing it in 3D has much less impact on compliance than in 2D. The void elimination channels in 3D instead find an equilibrium, where the internal void is split up in a way that each channel influences a significant portion of the void.

6 | Conclusion and Recommendations

6.1 Conclusion

The research aim of this thesis was to investigate new methods to prevent enclosed voids from forming during the topology optimization process, this has resulted in two new methods: the eigenfrequency method and the flood fill method.

The eigenfrequency method utilizes eigenfrequency analysis of an inverted density field and applies a minimum constraint to the eigenfrequencies to eliminate enclosed voids. It is difficult to predetermine the constraint value for specific properties of the void elimination feature, because for any given eigenfrequency constraint level, multiple combinations of properties like void size and void elimination channel width or length can be found. The main problem of this method is the occurrence of intermediate densities, as the eigenfrequency constraint could also be satisfied by them. Even with the use of multiple techniques to curb the use of these intermediate densities, the 2D compliance results were still either suboptimal or contained enclosed voids. In 3D the issue of intermediate densities was less prevalent and the method successfully eliminated enclosed voids in most tests, however the exact properties of the void elimination features were difficult to directly control with different eigenfrequency constraints. Finally, the additional computational effort was high due to the eigensolve used for the method, although this additional effort was successfully reduced by locking certain directions used in the calculation.

The flood fill method utilizes a modified flood fill algorithm, which is a filter with a resulting density field where elements monotonically increase from the starting boundary. This density field can then be incorporated as a filter, into the volume constraint or as a geometric constraint. This method successfully eliminates enclosed voids in geometric optimization by creating void elimination channels of 1 element wide. To directly control the width of these channels morphologic operators were then successfully incorporated into the method. In regular 2D compliance optimization the method utilized channels of intermediate densities to fill up all enclosed voids with more intermediate densities. Combined with the robust formulation all three implementations did successfully eliminate enclosed voids for multiple 2D problems with the same settings, which indicates that the method is universally applicable with these settings. The filter and geometric constraint implementation did show some suboptimal convergence behaviour, which can be solved for the latter by slowly tightening the constraint during the optimization. The method mostly worked well in 3D, although in some specific cases redundant void elimination channels were created. Direct control over the location, amount and size of the void elimination feature was also achieved by varying boundary elements, running additional flood fills and morphologic operators, respectively. Finally, the method requires very little computational effort for large meshes, as it scales linearly with the amount of elements.

After all tests done on both methods they can now be added to the rating table at the end of Chapter 2, which concisely compares these new methods to the existing methods. Table 6.1 shows that the eigenfrequency method did not necessarily perform better than the current methods. However the flood fill method has performed well on all criteria, making it a good choice for future topology optimization focused on AM.

6.2 Recommendations

With intermediate densities being the main drawback of the eigenfrequency method, the main recommendation is also on that topic. Testing the eigenfrequency method with topology optimization approaches that inherently create binary solutions, most notably the level set, ESO and BESO method is recommended. Using these approaches would eliminate the issue of intermediate densities and would allow for further research into the true effect of different eigenfrequency constraints, as in this thesis the results were mostly muddled by the inclusion of intermediate densities.

Method	User Control	Computational Effort	Ease of Use	Mechanical Performance	Geometric Complexity
Shortest path [20]	+	++	+−	−	++
Casting Constraint [11][12][13]	−	++	+	−−	−−
Machining Constraint [14][15][16]	+−	++	+	−	−−
Virtual Temperature Method [17][18]	−−	+	−−	+−	++
Side Constraint Method [21]	−	+	−	−	−
Void Projection Method [22]	++	+	+−	+	−
Eigenfrequency Method	−−	+−	+	+−	++
Flood fill Method	++	++	+	+	++

Table 6.1: Comparison of new methods with current void elimination methods

The flood fill algorithm allows a designer to specifically control the amount of void elimination features, their location as well as the width of these features. The one property that is missing from this list is the length of a void elimination feature, which would be a valuable parameter to also have control over, for instance because a longer path will have a larger chance of getting clogged with powder. One possible option to do this would be to add a specific offset based on the distance from the boundary, where combined with the robust formulation this offset would change any void element too far from the boundary to solid, regardless of whether or not a void elimination channel is connected to it. This boundary could even be the specific element at the start of the void elimination channel, since it is known for any specific element which other elements were processed before it. With this information of which internal element was influenced by a boundary element it could also be possible to have control over the size of internal voids, where some void size threshold can be introduced to decide whether a channel is made or it is changed to solid.

More alternatives to directly control void elimination features are also a possibility. The first is to also implement the coarser mesh method to control the size of void elimination features. This would be a possible alternative to morphologic operators as it offers a large decrease in computational effort at the cost of reduced geometric complexity. An alternative for when multiple holes are necessary is subdividing the mesh and then performing the flood fill with different boundary conditions for each subdivision. This would avoid the problem of additional computational effort added by running multiple flood fills at the cost of geometric complexity, although another advantage is that void elimination features are located closer to the actual void.

Additional tests should be done on the relation between the convergence behaviour of the geometric constraint and other parameters like mesh size, volume fraction constraint and filter radius. With the results from these tests it could be possible to determine a universal rule for scaling in the geometric constraint for different β continuation implementations.

Changes to the way the flood fill algorithm is written can also be useful. Firstly, the algorithm currently only works on simple square or cubic elements and therefore some alterations need to be made for it to also work on unstructured meshes. Sensitivities are currently calculated directly and in the same loop as the response, which was faster because the algorithm is written in an interpreted language. When using this method in a compiled language the disadvantage of calculating the sensitivities directly could outweigh the advantage of only using one loop. Therefore when this method is implemented with a compiled language an adjoint sensitivity formulation should be used, for which the algorithm would also need to save the order in which elements are processed.

Finally both the flood fill and eigenfrequency method could have another application next to eliminating enclosed voids. By simply inverting the methods they could be used to eliminate floating masses, which sometimes occur in topology optimization of eigenfrequencies or mixing features in fluid channels. The eigenfrequency method would then simply no longer be on the inverted but on the regular density field, while the flood fill method will work by utilizing a smooth minimum function and reversing the order in which elements are processed.

Bibliography

[1] D. Brackett, I. Ashcroft, and R. Hague, "Topology Optimization for Additive Manufacturing," tech. rep., Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, 2011.

[2] M. P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*. Berlin: Springer, 2003.

[3] E. C. Santos, M. Shiomi, K. Osakada, and T. Laoui, "Rapid manufacturing of metal components by laser forming," *International Journal of Machine Tools and Manufacture*, vol. 46, pp. 1459–1468, oct 2006.

[4] M. Attaran, "The rise of 3-D printing: The advantages of additive manufacturing over traditional manufacturing," *Business Horizons*, vol. 60, pp. 677–688, sep 2017.

[5] R. Liu, Z. Wang, T. Sparks, F. Liou, and J. Newkirk, "Aerospace applications of laser additive manufacturing," in *Laser Additive Manufacturing: Materials, Design, Technologies, and Applications*, pp. 351–371, Elsevier Inc., jan 2017.

[6] N. Hawaldar and J. Zhang, "A comparative study of fabrication of sand casting mold using additive manufacturing and conventional process," *International Journal of Advanced Manufacturing Technology*, vol. 97, no. 1-4, pp. 1037–1045, 2018.

[7] X. Guo, J. Zhou, W. Zhang, Z. Du, C. Liu, and Y. Liu, "Self-supporting structure design in additive manufacturing through explicit topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 323, pp. 27–63, aug 2017.

[8] D. Thomas, *The Development of Design Rules for Selective Laser Melting*. PhD thesis, University of Wales, 2009.

[9] S. N. Reddy, V. Maranan, T. W. Simpson, T. Palmer, and C. J. Dickman, "Application of topology optimization and design for additive manufacturing guidelines on an automotive component," in *Proceedings of the ASME Design Engineering Technical Conference*, vol. 2A-2016, 2016.

[10] K. Svanberg, "The method of moving asymptotes—a new method for structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 24, pp. 359–373, feb 1987.

[11] Y. Wang and Z. Kang, "Structural shape and topology optimization of cast parts using level set method," *International Journal for Numerical Methods in Engineering*, vol. 111, no. 13, pp. 1252–1273, 2017.

[12] Q. Xia, T. Shi, M. Y. Wang, and S. Liu, "A level set based method for the optimization of cast part," *Structural and Multidisciplinary Optimization*, vol. 41, no. 5, pp. 735–747, 2010.

[13] J. Lu and Y. Chen, "Manufacturable mechanical part design with constrained topology optimization," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 226, pp. 1727–1735, oct 2012.

[14] M. Langelaar, "Topology optimization for multi-axis machining," *Computer Methods in Applied Mechanics and Engineering*, vol. 351, pp. 226–252, jul 2019.

[15] S. L. Vatanabe, T. N. Lippi, C. R. Lima, G. H. Paulino, and E. C. Silva, "Topology optimization with manufacturing constraints: A unified projection-based approach," *Advances in Engineering Software*, vol. 100, pp. 97–112, oct 2016.

[16] J. Liu and Y. S. Ma, "3D level-set topology optimization: a machining feature-based approach," *Structural and Multidisciplinary Optimization*, vol. 52, pp. 563–582, sep 2015.

[17] S. Liu, Q. Li, W. Chen, L. Tong, and G. Cheng, "An identification method for enclosed voids restriction in manufacturability design for additive manufacturing structures," *Frontiers of Mechanical Engineering*, vol. 10, pp. 126–137, jun 2015.

[18] Q. Li, W. Chen, S. Liu, and L. Tong, "Structural topology optimization considering connectivity constraint," *Structural and Multidisciplinary Optimization*, vol. 54, pp. 971–984, oct 2016.

[19] Y. Luo, O. Sigmund, Q. Li, and S. Liu, "Additive manufacturing oriented topology optimization of structures with self-supported enclosed voids," *Computer Methods in Applied Mechanics and Engineering*, vol. 372, p. 113385, dec 2020.

[20] Y. Xiong, S. Yao, Z. L. Zhao, and Y. M. Xie, "A new approach to eliminating enclosed voids in topology optimization for additive manufacturing," *Additive Manufacturing*, vol. 32, p. 101006, mar 2020.

[21] L. Zhou and W. Zhang, "Topology optimization method with elimination of enclosed voids," *Structural and Multidisciplinary Optimization*, vol. 60, pp. 117–136, jul 2019.

[22] A. T. Gaynor and T. E. Johnson, "Eliminating occluded voids in additive manufacturing design via a projection-based topology optimization scheme," *Additive Manufacturing*, vol. 33, p. 101149, may 2020.

[23] T. E. Johnson and A. T. Gaynor, "Three-dimensional projection-based topology optimization for prescribed-angle self-supporting additively manufactured structures," *Additive Manufacturing*, vol. 24, pp. 667–686, dec 2018.

[24] J. Zhu, W. Zhang, and P. Beckers, "Integrated layout design of multi-component system," *International Journal for Numerical Methods in Engineering*, vol. 78, pp. 631–651, may 2009.

[25] T. E. Bruns and D. A. Tortorelli, "Topology optimization of non-linear elastic structures and compliant mechanisms," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 3443–3459, mar 2001.

[26] Z. D. Ma, N. Kikuchi, and I. Hagiwara, "Structural topology and shape optimization for a frequency response problem," *Computational Mechanics*, vol. 13, pp. 157–174, dec 1993.

[27] T. D. Tsai and C. C. Cheng, "Structural design for desired eigenfrequencies and mode shapes using topology optimization," *Structural and Multidisciplinary Optimization*, vol. 47, pp. 673–686, may 2013.

[28] G. Kreisselmeier and R. Steinhauser, "Systematic control design by optimizing a vector performance index," in *Computer Aided Design of Control Systems*, pp. 113–117, Elsevier, jan 1980.

[29] F. Wang, B. S. Lazarov, and O. Sigmund, "On projection methods, convergence and robust formulations in topology optimization," *Structural and Multidisciplinary Optimization*, vol. 43, pp. 767–784, jun 2011.

[30] G. Takács, "Smooth Maximum Based Algorithms for Classification, Regression, and Collaborative Filtering," Tech. Rep. 1, 2010.

[31] I. Zang, "A smoothing-out technique for min-max optimization," tech. rep., 1980.

[32] G. Zhao, Z. Wang, and H. Mou, "Uniform approximation of min/max functions by smooth splines," in *Journal of Computational and Applied Mathematics*, vol. 236, pp. 699–703, North-Holland, oct 2011.

[33] Y. Feng, L. Hongwei, Z. Shuisheng, and L. Sanyang, "A smoothing trust-region Newton-CG method for minimax problem," *Applied Mathematics and Computation*, vol. 199, pp. 581–589, jun 2008.

[34] J. A. Chatelon, D. W. Hearn, and T. J. Lowe, "A subgradient algorithm for certain minimax and minisum problems," *Mathematical Programming*, vol. 15, no. 1, pp. 130–145, 1978.

[35] O. Sigmund, "Morphology-based black and white filters for topology optimization," *Structural and Multidisciplinary Optimization*, vol. 33, pp. 401–424, apr 2007.

A | Eigenfrequency method

A.1 Increased penalization

Figure A.1 shows a complete overview of 2D results using the eigenfrequency method with slightly increased penalization ($q = 5$) and a varied eigenfrequency constraint. It is essentially an intermediate step between the regular penalization and extreme penalization used in Section 3.5.2.

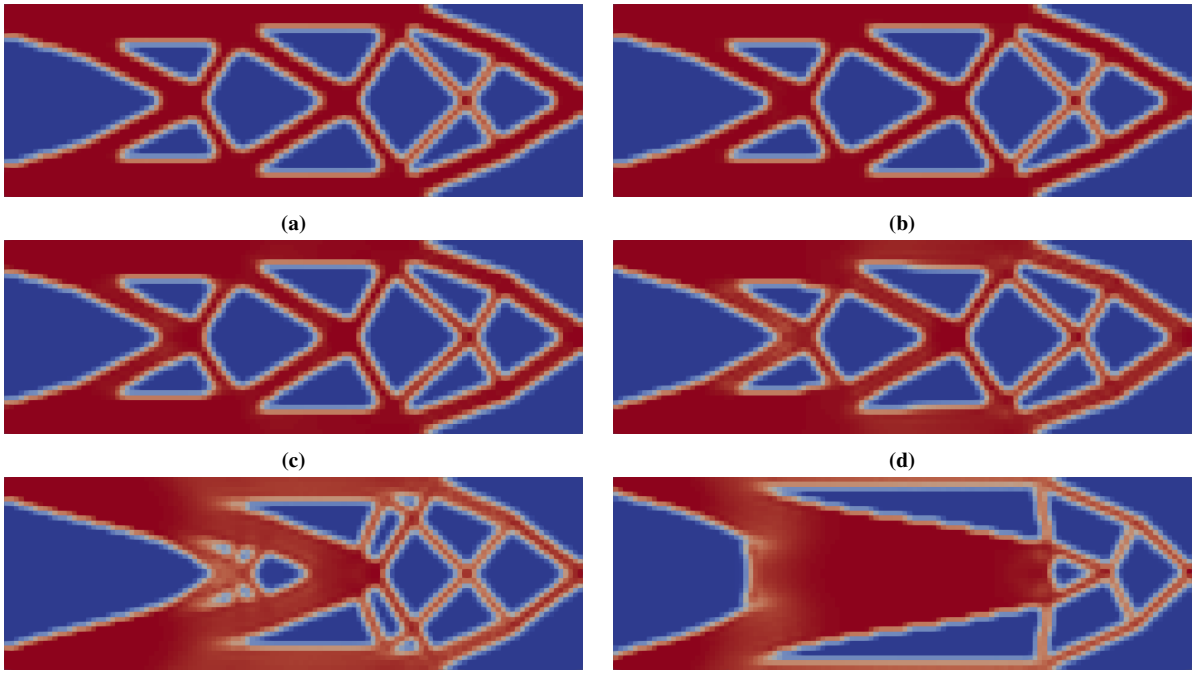
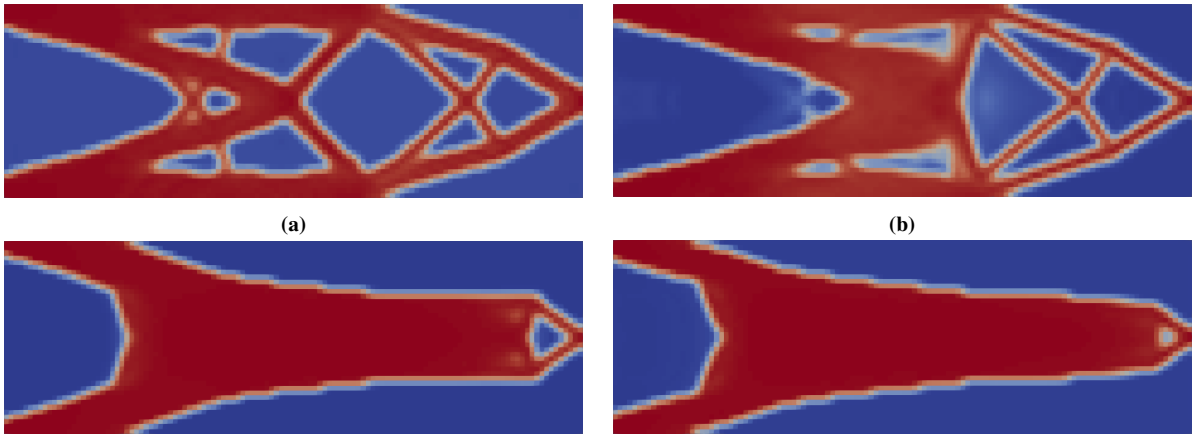


Figure A.1: Results with the eigenfrequency method with slightly increased penalization and a varied eigenfrequency constraint. (a) $\omega^2 = 10^7$; (b) $\omega^2 = 10^{7.5}$; (c) $\omega^2 = 10^8$; (d) $\omega^2 = 10^{8.5}$; (e) $\omega^2 = 10^9$; (f) $\omega^2 = 10^{9.5}$.

A.2 Extreme penalization

Figure A.2 shows some remaining 2D results using the eigenfrequency method with extreme penalization ($q = 10$) and a varied eigenfrequency constraint. Figure A.2c and A.2d have a slightly decreased and increased eigenfrequency constraint compared to the result shown in Section 3.5.2, respectively. This reconfirms how it is very difficult to determine the eigenfrequency constraint, as these two results do contain two small enclosed voids.

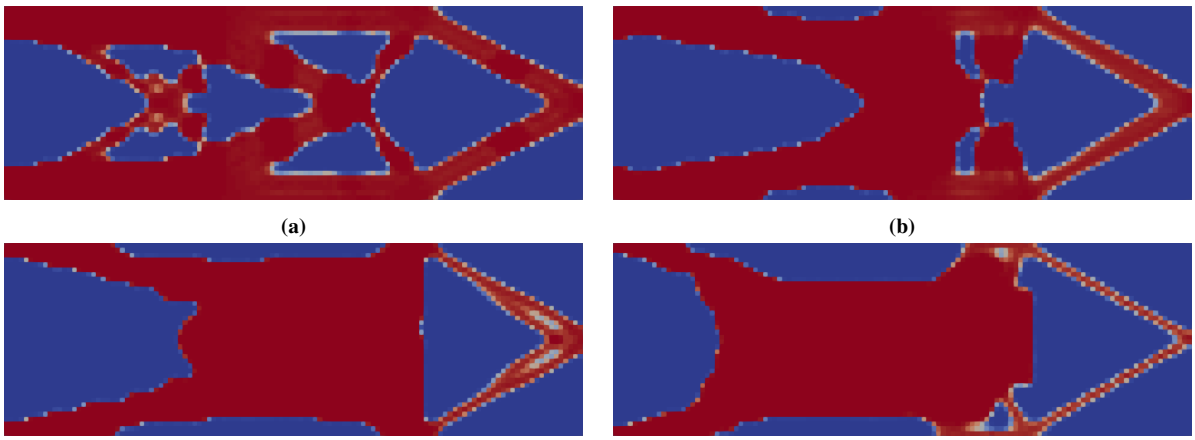


(a) **(b)** **(c)** **(d)**

Figure A.2: Results with the eigenfrequency method with slightly increased penalization and a varied eigenfrequency constraint. (a) $\omega^2 = 10^{8.5}$; (b) $\omega^2 = 10^9$; (c) $\omega^2 = 5 \cdot 10^9$; (d) $\omega^2 = 7 \cdot 10^9$.

A.3 Robust formulation

Figure A.3 shows additional results using the robust formulation with a β limit of 20 and slightly increased penalization ($q = 5$). Which shows that the void on the right is still deemed too important to remove.



(a) **(b)** **(c)** **(d)**

Figure A.3: Results with the eigenfrequency method with the robust formulation, slightly increased penalization and a varied eigenfrequency constraint. (a) $\omega^2 = 10^8$; (b) $\omega^2 = 10^{8.5}$; (c) $\omega^2 = 10^9$; (d) $\omega^2 = 10^{9.5}$.

B | Flood fill method

B.1 Formulation of the Geometric constraint

The geometric constraint used the following smooth approximation of an absolute value:

$$s_A = x\frac{e^{px}}{e^{px}+e^{-px}} - x\frac{e^{-px}}{e^{px}+e^{-px}}. \tag{B.1}$$

Figure B.1 plots the behaviour of s_A for different values of p, which shows how the approximation slightly underestimates the true absolute value at small values of x. This is also the reason why a relatively large cluster of elements could take an intermediate density in Figure 4.17a and 4.17b.

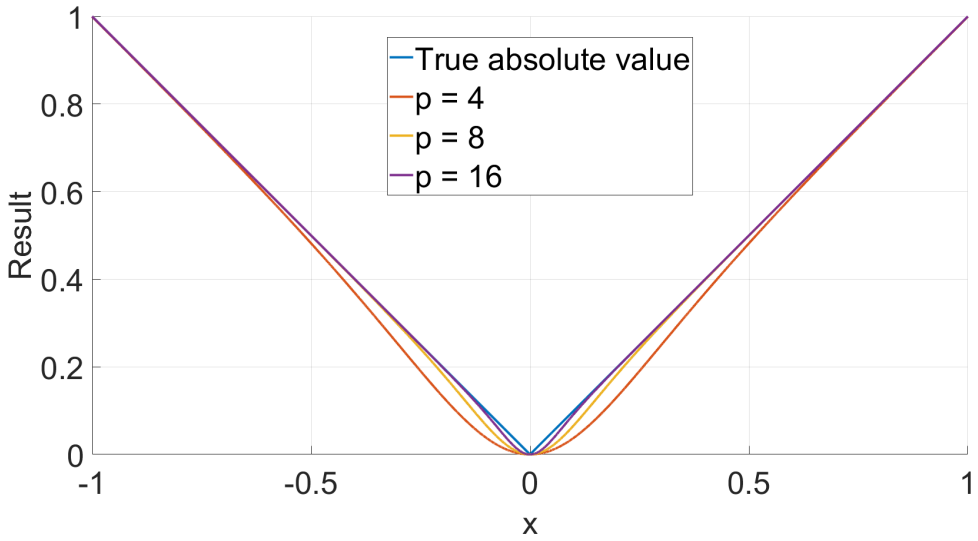


Figure B.1: Smooth approximation of the absolute value for different values of p.

The sensitivities of this smooth approximation are given in the following equation:

$$\frac{\partial s_A}{\partial x} = (xp)\left(1 - \frac{(e^{px}-e^{-px})^2}{(e^{px}+e^{-px})^2}\right) + \frac{(e^{px}-e^{-px})^2}{(e^{px}+e^{-px})^2} \tag{B.2}$$

B.2 Element processing order

Figure B.2 shows how the flood fill order of the result in Figure 4.10 changes in its first few iterations. It shows how there is some slight variation in the processing order, but the only really drastic step is from the first iteration to the second iteration.

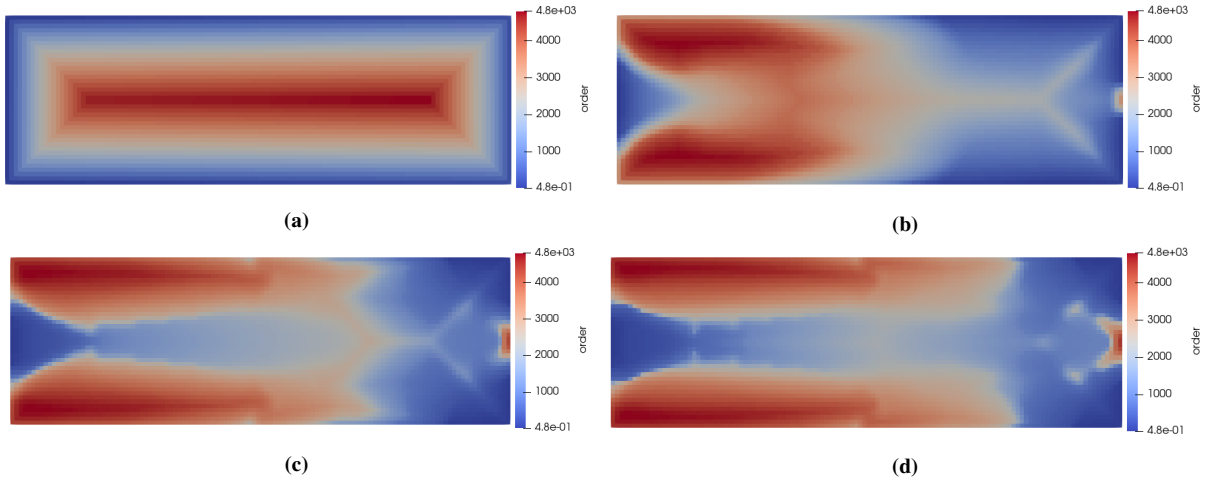


Figure B.2: How the flood fill order changes in early iterations of an optimization. (a) iteration 1; (b) iteration 2; (c) iteration 3; (d) iteration 4.

Figure B.3 shows the removal of redundant channels as discussed in Section 5.3 in closer detail on the result of Figure 4.10, where a slight variation in density causes a specific hole elimination channel to lose influence, which subsequently closes that channel. The comparison with the actual corresponding design at that iteration also shows how minute the differences actually are when these orders are altered. It is also important to note that the color scale was altered slightly for the latter two subfigures, as displayed in the legend bar next to it.

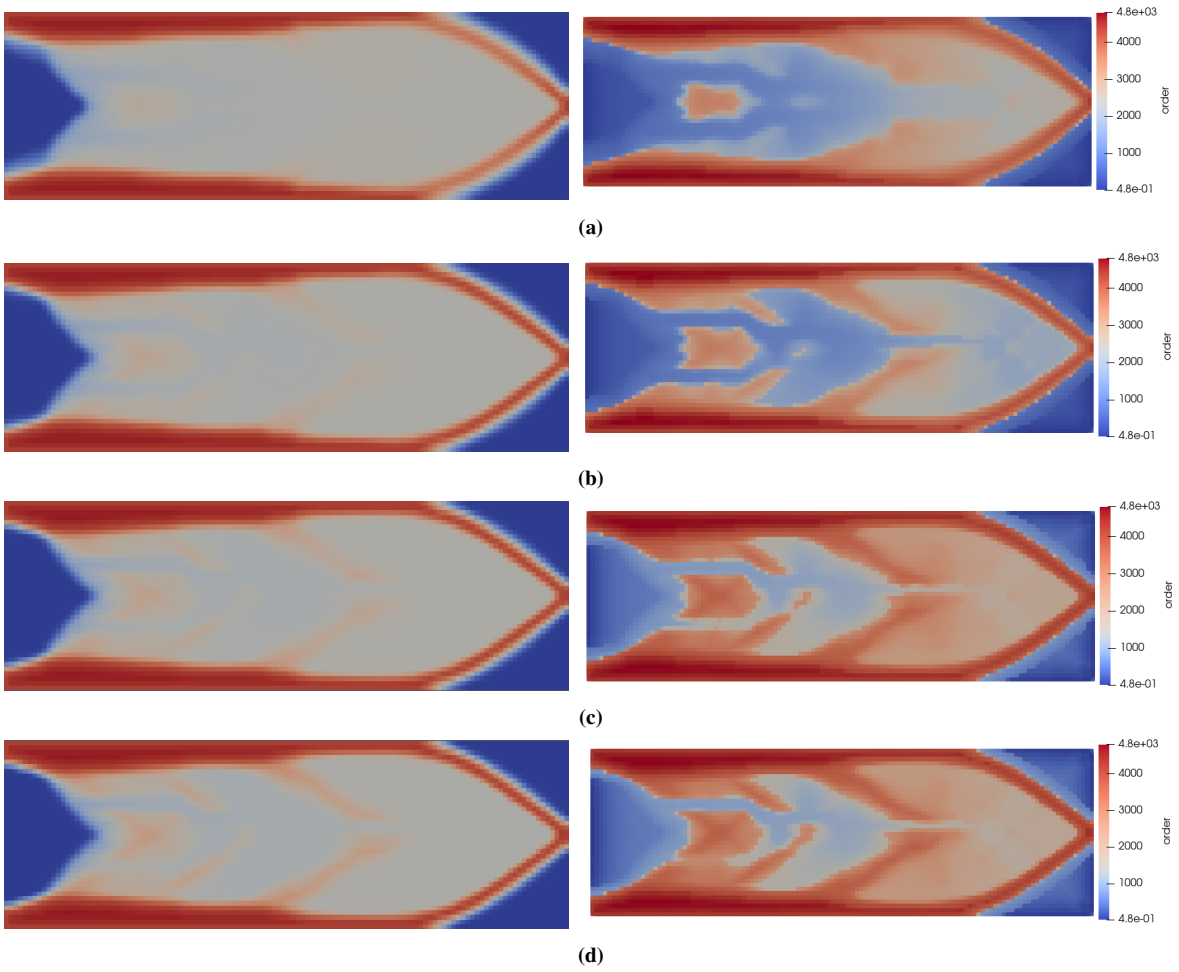


Figure B.3: How redundant void elimination channels are closed during the optimization, with the design at a specific iteration on the left and its corresponding order on the right. (a) Iteration 10 where two main access channels can be identified; (b) Iteration 12 where the lower access channel has a slight increase in density; (c) Iteration 13 where the redundant access channel is clearly lower in the processing order; (d) Iteration 14 where the redundant access channel is fully closed.

B.3 Additional results when varying p

Figure B.4, B.5 and B.6 shows designs for additional values of p.

B.3.1 Filter implementation

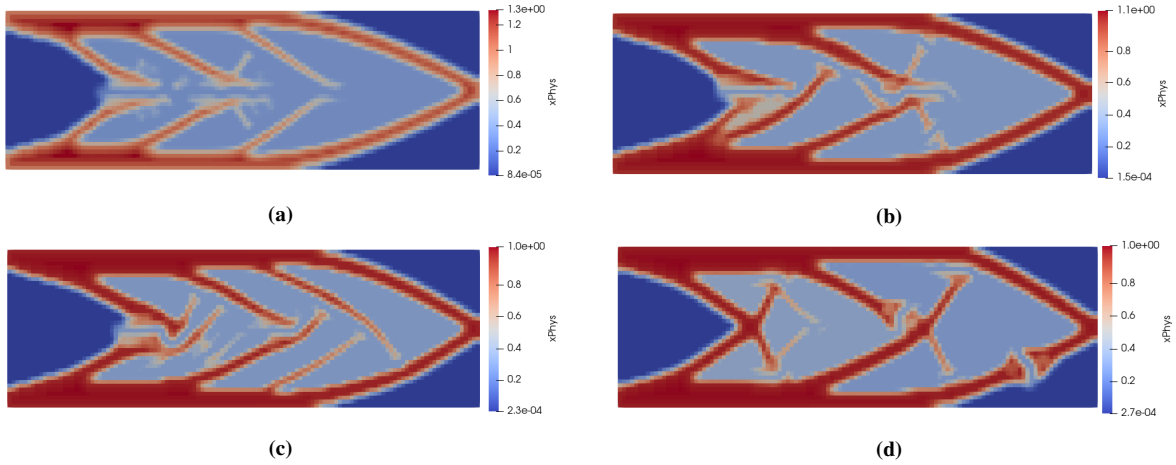


Figure B.4: Results for the cantilever problem with the flood fill filter implementation applied with different values of p. (a) $p = 5$; (b) $p = 30$; (c) $p = 40$; (d) $p = 50$.

B.3.2 Volume constraint implementation

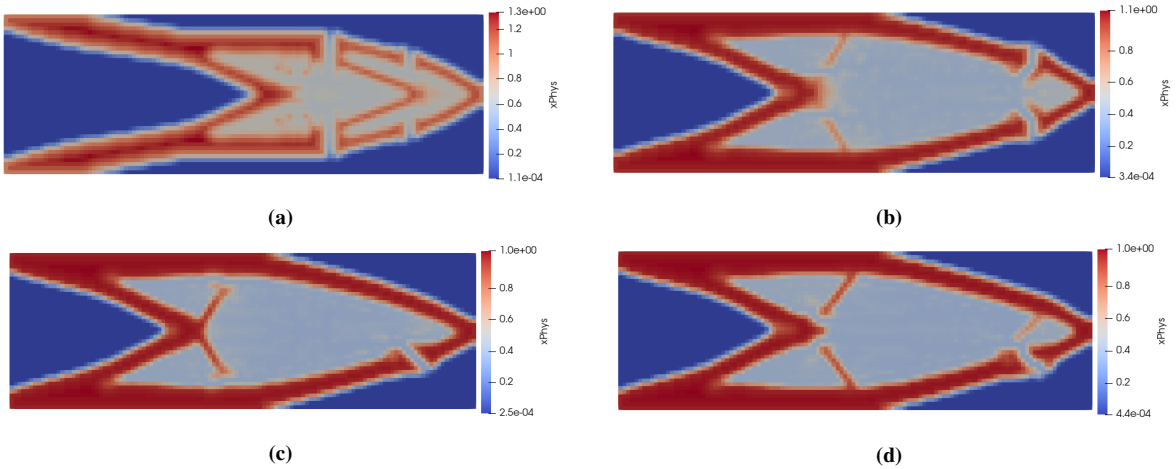


Figure B.5: Results for the cantilever problem with the flood fill volume constraint implementation applied with different values of p. (a) $p = 5$; (b) $p = 30$; (c) $p = 40$; (d) $p = 50$.

B.3.3 Geometric constraint implementation

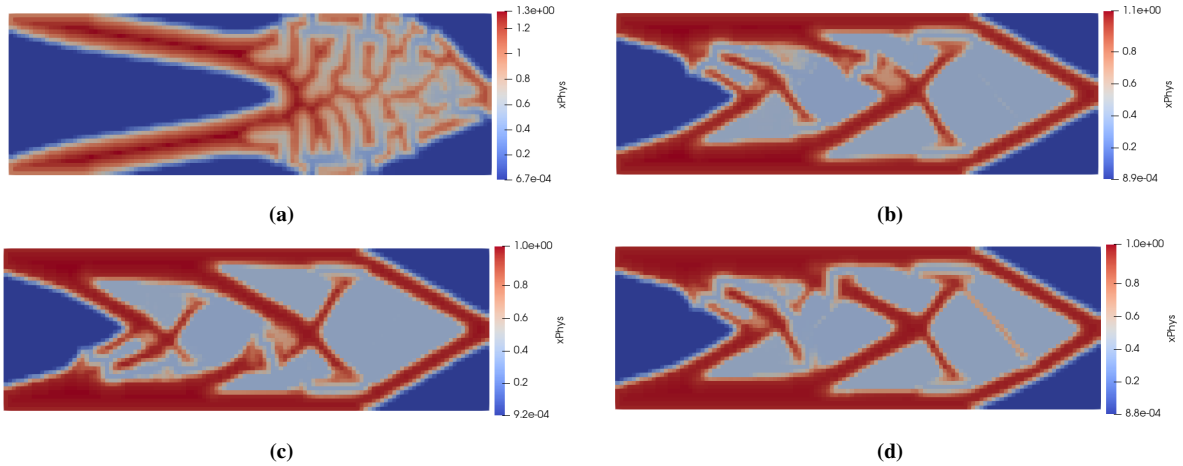


Figure B.6: Results for the cantilever problem with the flood fill geometric constraint implementation applied with $\chi = 300$ and different values of p. (a) $p = 5$; (b) $p = 30$; (c) $p = 40$; (d) $p = 50$.

B.4 Additional results when varying flood fill boundary

Currently 2D results use all outside surfaces as starting point for the flood fill. Excluding some boundaries is unnecessarily restrictive in 2D, although it could offer some information what boundaries are important for a specific problem. Some additional internal locations can also be specified as flood fill boundary, which could show large improvements on compliance. The four specific cases used and their results using the volume constraint implementation are shown in Figure B.7, which shows that excluding the left surface has caused the method to no longer successfully eliminate enclosed voids. Furthermore it shows that opening just a single hole already improves the compliance by a large amount.

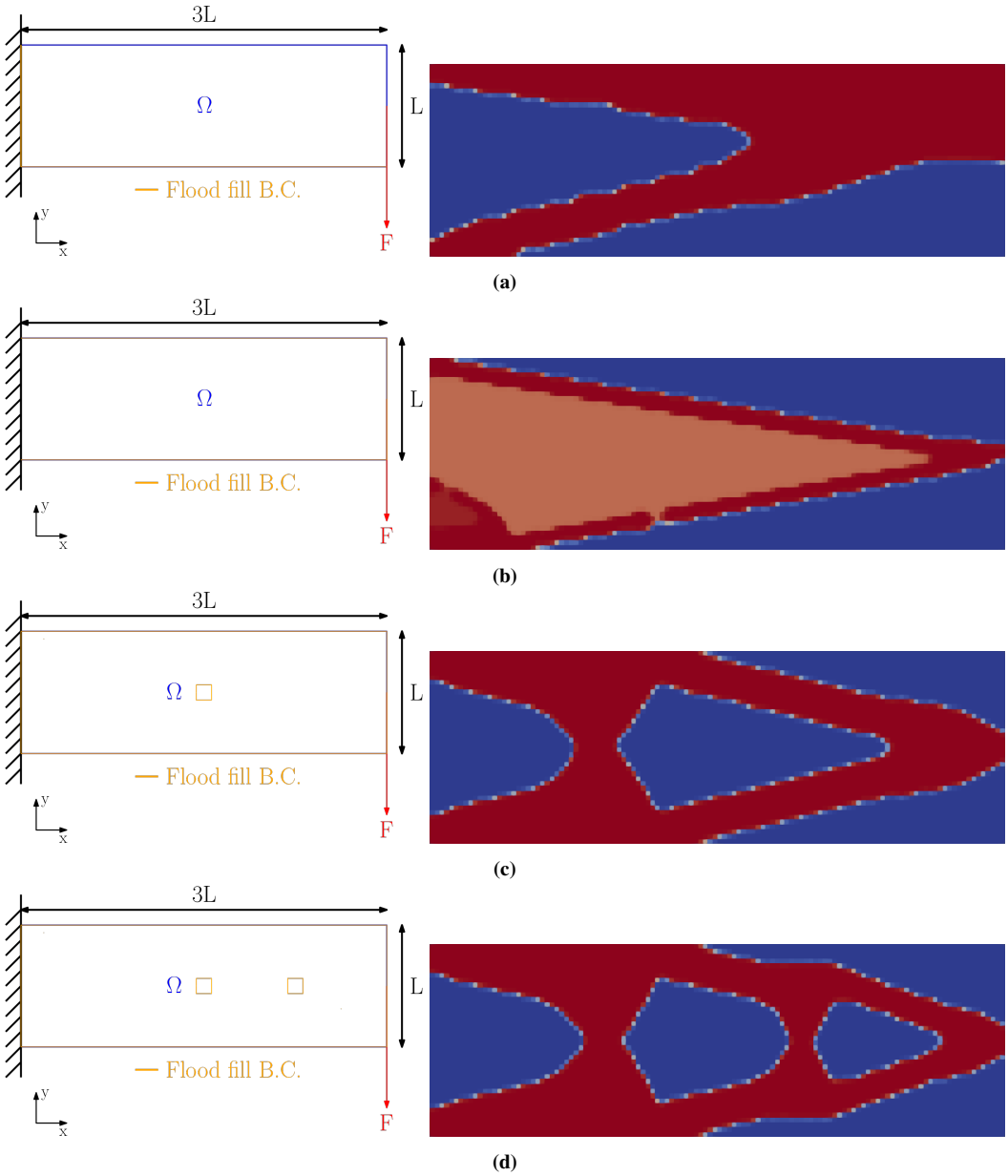


Figure B.7: Case descriptions and robust results of different flood fill boundaries with a β limit of 12. (a) Right and top surfaces excluded with $\frac{c}{c_{ref}} = 147.2\%$; (b) Left surface excluded; (c) Element in the middle included with $\frac{c}{c_{ref}} = 115.4\%$; (d) Elements in middle and on $\frac{3}{4}$ of x-axis included with $\frac{c}{c_{ref}} = 113.6\%$.

B.5 Early iterations of specific results

Figure B.8 shows the early iterations of the Mitchell cantilever results using the filter implementation, which shows why the result in Figure 4.24 got stuck in a slightly suboptimal shape.

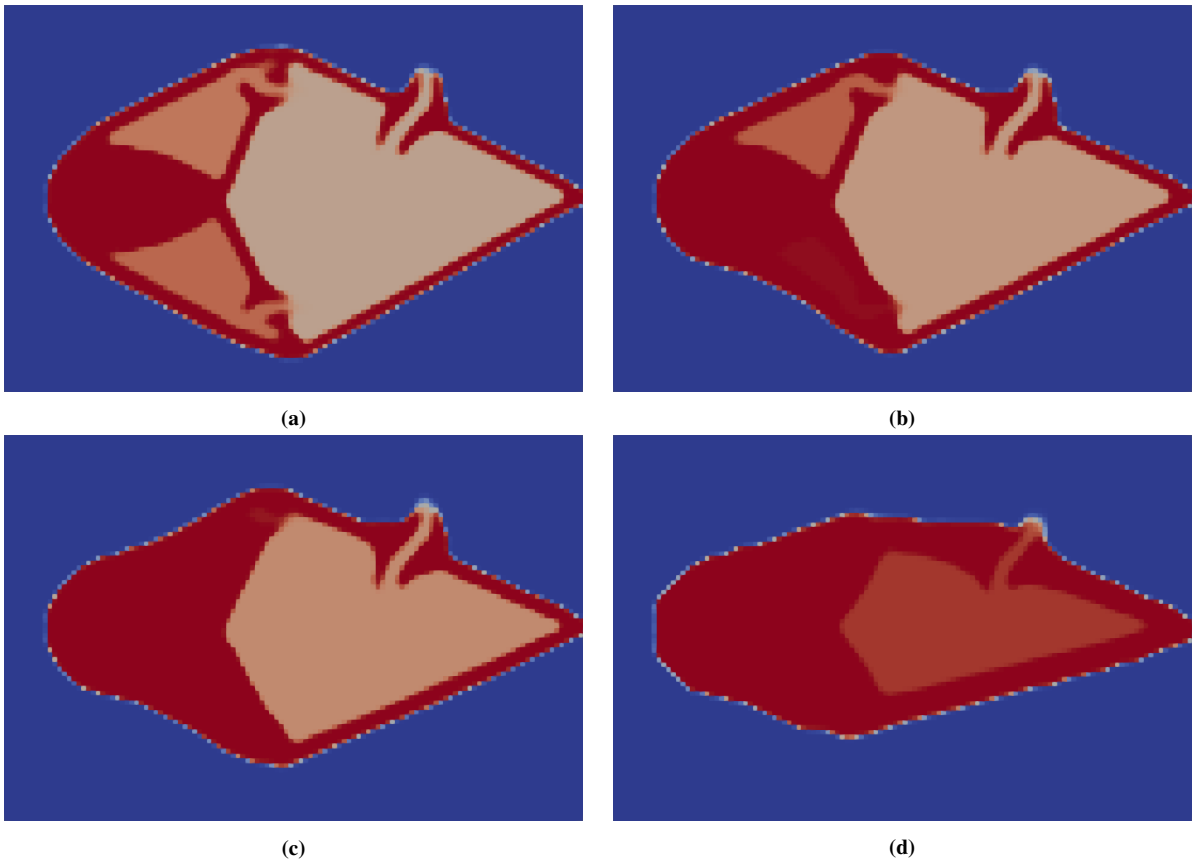


Figure B.8: Early iterations of the filter implementation used for the Mitchell cantilever problem. (a) iteration 90; (b) iteration 130; (c) iteration 160; (d) iteration 200.

Figure B.9 shows the early iterations of the table results for the using the geometric constraint implementation, which visualizes why the result in Figure 4.25 got stuck in a suboptimal shape.

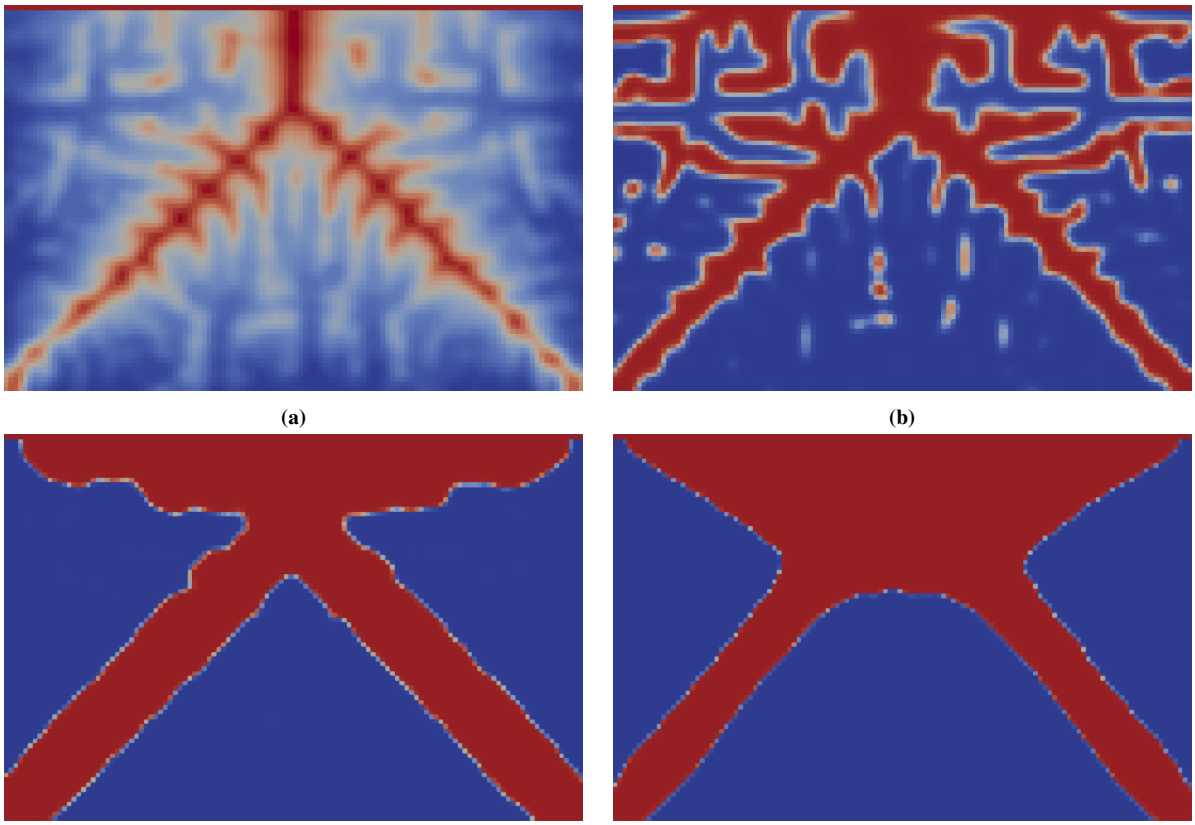


Figure B.9: Early iterations of the geometric constraint implementation used for the table problem. (a) iteration 10; (b) iteration 50; (c) iteration 70; (d) iteration 170.

C | Flood fill code

```
import numpy as np
from optiblock import OptiBlock, Signal

class Floodfill(OptiBlock):
   def _prepare(self, model, bc, P):
      #Initialize optimization constants
      self.model = model    #Load in FE model used
      self.bc = bc          #Load in prescribed boundary elements
      self.P = P            #Smooth maximum parameter

      #Initialize model dimensions
      self.nx = self.model.nx
      self.ny = self.model.ny
      self.nz = self.model.nz
      self.A = self.ny*self.nz

   def _response(self, x):
      #initialize lists:
      #Q is queue of elements, d is list of densities and result stores the result
      #self.sens is the Jacobian of the flood filled densities to the input densities
      x = self.sig_in[0]
      d = x.get_state().copy()
      Q = np.empty(self.model.nel)
      Q[:] = np.nan
      Q[self.bc] = d[self.bc]
      Result = np.copy(Q)
      self.sens = np.eye(self.model.nel)

      #Initialize adjacent element arrays
         if np.equal(self.nz, 0):
            Adj = np.array([-1, 1, -self.ny, self.ny])
         else:
            Adj = np.array([-1, 1, -self.nz, self.nz, -self.A, self.A])


      for i in range(self.model.nel):
         #Select new element
         Curr = np.nanargmin(Q)
         Xik = Result[Curr]
         NewEl = np.ones_like(Adj)*Curr + Adj

         #Filter out edge elements
         if np.equal(self.nz, 0):
            Moduy = np.remainder(Curr, self.ny)
```

```
            NewEl = NewEl[np.array([Moduy, Moduy+1-self.ny, np.greater(NewEl[2], -1),
             ↪  np.less(NewEl[3], self.model.nel)], dtype= np.bool_)]
         else:
            Moduz = np.remainder(Curr, self.nz)
            Moduyz = np.remainder(Curr, self.A)
            NewEl = NewEl[np.array([Moduz, Moduz+1-self.nz, np.greater(Moduyz,
             ↪  self.nz-1), np.less(Moduyz, self.A-self.nz),  np.greater(NewEl[4], -1),
             ↪  np.less(NewEl[5], self.model.nel)], dtype=np.bool_)]

         #Filter out elements already in Q
         NewEl = NewEl[np.isnan(Result[NewEl])]

         #Update densities and lists
         Rhoi = d[NewEl]

         ##p-norm smooth maximum
         Xii = np.power(np.add(np.power(Rhoi, self.P), np.power(Xik, self.P)), 1/self.P)

         #update lists
         Q[NewEl] = Xii
         Q[Curr] = np.nan
         Result[NewEl] = Xii

         #update sensitivities
         if np.logical_not(np.equal(NewEl.size, 0)):
            ##p-norm
            self.sens[:, NewEl] = np.copy(self.sens[:,
             ↪  [Curr]])*(Xik**(self.P-1))*(Xik**self.P + Rhoi**self.P)**(1/self.P - 1)
            self.sens[NewEl, NewEl] = (Rhoi**(self.P-1))*(Rhoi**self.P +
             ↪  Xik**self.P)**(1/self.P - 1)
      return [Result]

   def _sensitivity(self, dfdv):
      sensin = np.ones(self.model.nel)*np.copy(dfdv[0])
      sensout = np.matmul(self.sens, sensin)
      return [sensout]
```