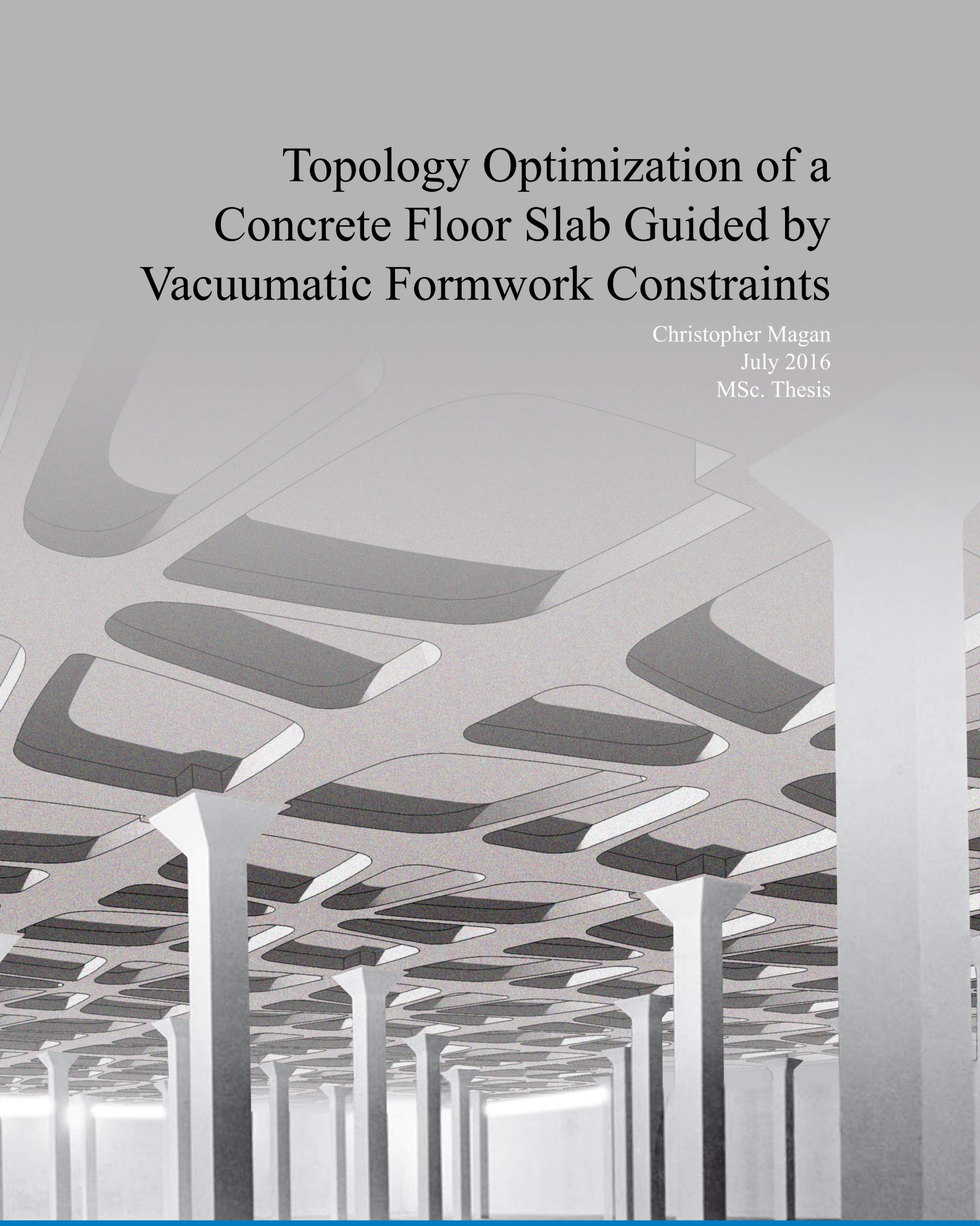
# Topology Optimization of a Concrete Floor Slab Guided by Vacuumatic Formwork Constraints

Christopher Magan
July 2016
MSc. Thesis



abt
BEM Next lab
TU Delft

*Cover photo based on Nervi (1963)*
*Electronic version thesis & topology optimization algorithm:*
`http://repository.tudelft.nl/`

*Author*

Christopher Magan
Cornelis Trompstraat 38, 2628RR Delft
Student No. 4009495
E: c.m.magan@gmail.com
T: +31 (0) 6 19 45 30 70

*Committee*

prof. ir. R. Nijsse
Delft University of Technology
E: r.nijsse@tudelft.nl

prof. ir. A.Q.C. van der Horst
Delft University of Technology
E: a.q.c.vanderhorst@tudelft.nl

dr. ir. J.L. Coenders
Delft University of Technology
BEMNext Laboratory
White Lioness technologies
E: jeroencoenders@white-lioness.com
T: +31 (0) 20 737 1997

dr. ir. arch. F.A.A. Huijben
ABT bv
E: f.huijben@abt.eu
T: +31 (0) 15 270 36 63

# Preface

This Master's thesis concludes my degree at Delft University of Technology. It was written as part of the curriculum of the Master Building Engineering with a specialization in Structural Design at Delft University of Technology. The research has been performed in collaboration with the Faculty of Civil Engineering, the BEMNext Lab, and ABT bv. The making of this thesis would not have been possible without the help and guidance of the following people.

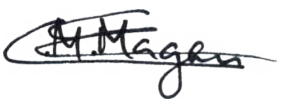
First of all, I would like to thank my supervisors prof. R. Nijsse for chairing the committee, prof. ir. A.Q.C. van der Horst for his insight in the construction industry, dr. ir. J.L. Coenders for sparking my interest in structural optimization, and dr. ir. arch. F.A.A. Huijben for his enthusiasm and vital knowledge of vacuumatic formwork.

Secondly, my colleagues at ABT bv were invaluable for their feedback and interest in my work. In particular, Chris van der Ploeg played a tremendous role in guiding me through the process, basically acting as an additional daily supervisor. The pleasant working environment and extensive knowledge at ABT bv contributed greatly to my motivation. My colleagues at the Provincie Zuid-Holland were also very important in that respect.

I would also like to thank Jeroen Coenders and Frank Huijben for giving me the chance to experience the International Association for Shell and Spatial Structures (IASS) symposium 2015 in Amsterdam. Together with Arjen Deetman, Jelmer Feenstra, Roel Schippers, and many others, they helped provide a memorable workshop and extraordinary week.

Last but not least, I would like to express my gratitude towards my friends and family. Specifically, I would like to thank my girlfriend Lara Blom, my parents Lilian and Michael Magan, and my brother Alexander Magan for their endless support and care.

Thank you!



*Christopher Magan*

*Delft, July 2016*

# Abstract

This thesis presents a design for a topology-optimized concrete floor slab, of which the structural optimization process is guided by manufacturability constraints from a vacuumatic formwork. The design has been obtained using an open-source, three-dimensional topology optimization algorithm developed in Python.

Optimization plays a significant role in both this thesis and, increasingly, in the building industry as a whole. Improving the quality of buildings is something many engineers strive for, but there are many different aspects to optimize for. For example, the sustainability can be improved or the amount of sunlight entering a building can be optimized. Structurally, it is often attempted to reduce the self-weight of the structure, since this can contribute to a reduction in material usage and foundation costs. According to Georgopoulos and Minson (2014), floor slabs can comprise up to 85% of the self-weight of a concrete structure. Therefore, the choice in flooring system can be very influential in terms of structural dimensioning.

A common method for reducing the structural weight of a floor is by means of a sizing optimization. By changing the slab thickness, the concrete type and/or the amount of reinforcement, a floor slab can be optimized until it reaches the standard set by the building codes. Other methods for reducing the floor slab weight include shape optimization and topology optimization. In this thesis, topology optimization is applied since it offers the most design freedom and, consequently, may provide greater advantages.

Traditional floor slabs systems are cost-optimized but can have the disadvantage of being structurally inefficient. The excessive weight has negative consequences in sustainability and structural dimensioning of both the floor slab itself and the supporting structure consisting of beams, walls, columns, foundation, et cetera. Topology optimization allows for efficient material distribution, and thus a reduction in weight. Topology-optimized floors are typically regarded as being difficult to produce, however, and as such cost too much to be considered in building designs.

In order to reach a compromise between a low self-weight and low production costs, two features are included in the optimization process in this thesis. First, manufacturability is directly incorporated in the optimization, rather than afterward. Secondly, the highly malleable vacuumatic formwork system has been used as a premise. Stabilizing sand formwork by means of a vacuum allows the sand to assume more complex shapes than its own angle of repose will support. Major advantages of this technique are reusability and adaptability. These advantages may cause the formwork costs to be reduced considerably when producing floor slabs. Vacuumatic formwork provides a method for producing complex shapes with

a low-tech and sustainable procedure, and accordingly strengthens the advantages of topology optimization while counteracting the disadvantages.

In order to utilize topology optimization for designing a concrete floor slab, several features are incorporated in the topology optimization algorithm. In order to correctly model concrete, self-weight and a reduced Young's modulus for tensile elements are included. From a functional point of view, the top of the floor is required to be flat. Lastly, a minimum concrete thickness and casting constraint are applied in the iterative optimization process to make the design more suitable for manufacturing with vacuumatic formwork.

By developing an open-source topology optimization algorithm in Python, the final result consists of a preliminary design for a topology-optimized concrete floor slab to be produced with a vacuumatic formwork system. The advantages of this slab compared to a traditional, monolithic slab include a weight reduction of 24%, the potential for a cost decrease, a $CO_2$ emission reduction of at least 17%, and increased architectural value.

# Table of contents

# 1 Introduction

When a client wants to acquire a new building in order to fulfill a specific function, a team of engineers, architects, contractors etc. are assembled to make a design that best satisfies all the requirements set by the client. The conception this design consists of several stages in which there is an iterative process of optimizing the building as the final model becomes more and more clear. The optimization goal of a project can be structural, cost, or aesthetics-based for example depending on the requirements set by the client.

The development of a new project starts with a preliminary design and the load-bearing structure of the building is no different. In this phase, there are still many options to fulfill the safety requirements regarding the structural system, material, shape, member cross-section and many more. Traditionally, before computational power was not readily available, choices made by structural engineers were largely based on experience, rules of thumb, simplified hand calculations, or in worse cases based on subconscious motives, random selections, or even mere superstition (Iyengar, 2004). Several iterations later this would result in a final design that is more refined when compared to the original preliminary design.

Design optimization in this thesis is largely based on the same traditional philosophy. However, the recent advances in computational power have led to considerable differences. Nowadays, choices can be made based on the results of sophisticated Finite Element Analysis software. The number of iterations can be increased considerably, meaning that instead of five or ten, there can be hundreds. The process by which the best options are selected differs also. Rather than basing choices on experience, the choices are automatically made by the evaluation algorithms using the specified boundary conditions and design variables; there has been a shift from qualitative to quantitative decision-making. In conclusion, the vague traditional decision-making aspects have made way for clear constraints, which can result in a structure with a higher level of optimization.

Provided that setting up the parametric structural model does not take too long, the advantages of structural optimization may include cost saving by making the rest of the design process more time-efficient. Reducing material usage has the potential to help reduce expense and can result in a more sustainable building too. Lastly, as computers allow for more design options to be evaluated, structural optimization may add a new dimension of creativity and thus value for the client.

Structural optimization can be summarized as a design process in which there are three distinctive types of optimization. After choosing a structural system, the process begins with topology optimization, which can be described as the optimal placement of material in terms of stress/stiffness in a design space. For a concrete floor, this can be streamlined to the optimal placement of the ribs, which stiffen the relatively thinner floor slab, resulting in an overall lower weight. While this brings disadvantages in terms of fire safety and sound insulation, however, from a structural, sustainable, and aesthetic point of view, in this thesis, it is assumed that the light-weight floor is an improvement. Usually, the final design in the construction stage differs from the topology-optimized structure because of manufacturability. In theory, a structural topology-optimized floor is very efficient, but in practice, it can be too expensive to produce. Complex connections and irregular members cause the price to increase considerably. Therefore, what usually happens is that, after the structural topology optimization of an element, the design is manually modified in order to be suitable for manufacturing. This implies that the element is designed in terms of manufacturability or costs and not entirely optimized in terms of its structural performance anymore.

Concrete floors are a common component of buildings and over the years these building elements have been greatly optimized for costs instead of weight. For almost every span or support, there is a floor system that can satisfy those requirements for a reasonable price. The majority of standard floor systems are flat plates because formwork takes up roughly sixty percent of the production costs in the US (Fanella and Alsamsam, 2007). Other Western countries will show similarities in this cost distribution. The superfluous concrete that is used consequently is not enough to justify complex surfaces. However, with the recent increase in interest in sustainability and architectural value, there is potential in a topology-optimized floor system, provided there is additional funding.

Up to now, the focus of sustainable building has been on reducing energy consumption in heating, cooling, et cetera. While this has resulted in considerable improvements in energy consumption, a truly sustainable building should also take structural material usage into account. Material usage is becoming relatively more important as energy consumption is being reduced more and more (Anink et al., 2015). In The Netherlands, an environmental performance check called Milieuprestatieberekening Gebouwen (MPG) has been required for the permit of new buildings larger than $100\,\mathrm{m}^2$ as of January 2013. The method is based on the Life Cycle Analysis (LCA), which takes the entire life of an element into account when assigning it a certain sustainability level. The regulation does not include a minimum quality level yet, however, allowing construction firms to get used to the new method (Rijksoverheid, 2015; Boon et al., 2014). It can be expected that in the near future there will be stricter rules on the environmental impact of material use, so a lighter optimized

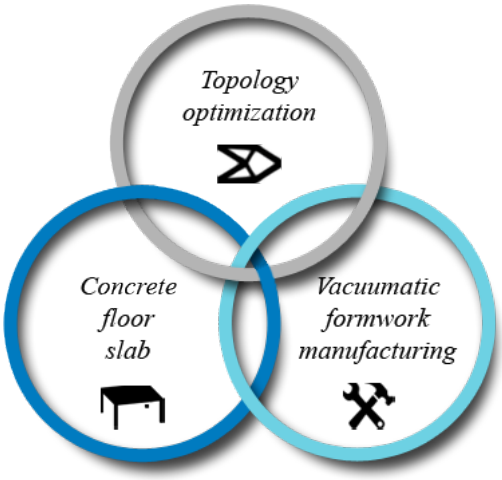



***Figure 1.1:*** *Three main components of this thesis*

floor can have a positive effect on the MPG. Together with the additional aesthetic enhancements, this gives a building more value for its users. Figure 1.1 displays the three main components that make up this thesis. A concrete floor slab has been topology-optimized with manufacturability constraints from a vacuumatic formwork.

## 1.1 PROBLEM DEFINITION

Traditional cost-optimized concrete floor systems can have the disadvantage of being structurally inefficient and, therefore, using a lot of material. The excessive weight has negative consequences in sustainability and structural dimensioning of both the floor slab itself and its supporting structure consisting of beams, walls, columns, foundation, etc. Topology optimization allows for efficient material distribution in floor designs, and thus a reduction in weight. Topology-optimized floors are typically regarded as being difficult to produce, however, and as such cost too much to be considered in building designs.

## 1.2 RESEARCH QUESTION

*How can the topology of a concrete floor slab be optimized when taking manufacturability constraints from a vacuumatic formwork into account, and how would it compare to a traditional, monolithic floor slab in terms of weight, costs, sustainability, and appearance?*

## 1.3 RESEARCH OBJECTIVE

The result of this research should help make topology-optimized floors more feasible in terms of manufacturability and design, resolving the problem of designing lightweight floors that are too complex to produce. By developing an open-source topology optimization algorithm in Python, the final result should consist of a design for a lightweight concrete floor to be produced with a vacuumatic formwork system.

## 1.4 HYPOTHESIS & EXPECTATIONS

The problem of diminishing benefits of topology optimization due to manufacturability constraints can be solved by reaching a compromise between a topology-optimized lightweight floor and a cost-optimized standard floor. To achieve this, first, manufacturability is incorporated in the optimization loop, rather than afterward. Secondly, the highly malleable vacuumatic formwork system allows a sand formwork to assume more complex shapes than its own angle of repose will support. Major advantages of this technique are reusability and adaptability. These advantages mean that the formwork cost may be reduced severely when producing floors, especially since floors have a high rate of repetition in most multi-story buildings. By combining vacuumatic formwork with topology optimization, a floor slab can be designed that has advantages over a traditional, monolithic floor slab in weight, costs, sustainability, and architectural value.

## 1.5 RESEARCH METHODOLOGY

In order to reach the research objective, this thesis starts with an introduction to structural optimization. Three distinct optimization types can be distinguished: cross-section optimization, shape optimization, and topology optimization. Topology optimization, being part of the research objective, is analyzed in more detail. This iterative design method is applied to a concrete floor slab, so several alternatives have been researched. At this point, both the optimization method and floor slabs have been clarified, so the topology optimization algorithm is introduced. In order to clarify how the algorithm works, pseudo code, an extensive step-by-step example, and several smaller examples are employed.

Validation of the algorithm has been performed with examples from literature, after which it is ready to be applied to a floor slab. The problem that arises with regard to manufacturability has been described in problem definition. Since vacuumatic formwork forms one part of the solution to making the design more suitable for manufacturing, the method itself is researched and the ensuing required additions to the algorithm are defined. Research of the method has been performed with a literature study, first, and with a practical workshop, second.

The final part of this thesis brings vacuumatic formwork and the additional features together in a modified topology optimization algorithm. In order to show the influence of each feature, the topology-optimized floor slab design is gradually improved until the final floor slab is presented. In order to check whether the objective has been reached, the topology-optimized floor slab is compared to a traditional, monolithic floor slab.

## 1.6 THESIS OUTLINE

*Chapter 1 – Introduction*
The current chapter includes an introduction to the research subject. Figure 1.2 displays how the chapters are related, and what is discussed in each one.

*Chapter 2 – Structural optimization*
In the first part of the literature study, both structural optimization in general and topology optimization, in particular, have been described.

*Chapter 3 – Ribbed floor slab systems*
In this chapter, several ribbed floor slab systems are analyzed. Floor slabs with isostatic ribs are examined more extensively. While this system is vastly different from topology optimization, the method can also be applied to the considered floor slab of this thesis to provide some degree of comparability.

*Chapter 4 – Topology optimization algorithm*
The topology optimization algorithm is demonstrated here. The input variables are clarified, so the algorithm can be used by anyone for any topology optimization problem. The chapter concludes with a standard topology-optimized floor slab design.

*Chapter 5 – Vacuumatic formwork*
The highly malleable vacuumatic formwork system can help make the complex floor slab design more feasible. The chapter includes a description of the method and an analysis of the strengths, weaknesses, opportunities, and threats.

*Chapter 6 – Topology-optimized floor slab design*
Using the topology optimization algorithm developed in Chapter 4, a design is made for a concrete floor slab that can be produced with vacuumatic formwork.

*Chapter 7 – Discussion*
The discussion verifies the hypothesis, offers advice on advantages and disadvantages of the optimizations, and proposes several applications for both the algorithm and the topology-optimized floor slab.

*Chapter 8 – Conclusions*
The conclusion includes the answer to the research question and various general conclusions.

*Chapter 9 – Recommendations*
The recommendations are split into three parts. There are suggestions for the algorithm, for topology optimization and manufacturability, and for topology optimization and manufacturability of a concrete floor slab.

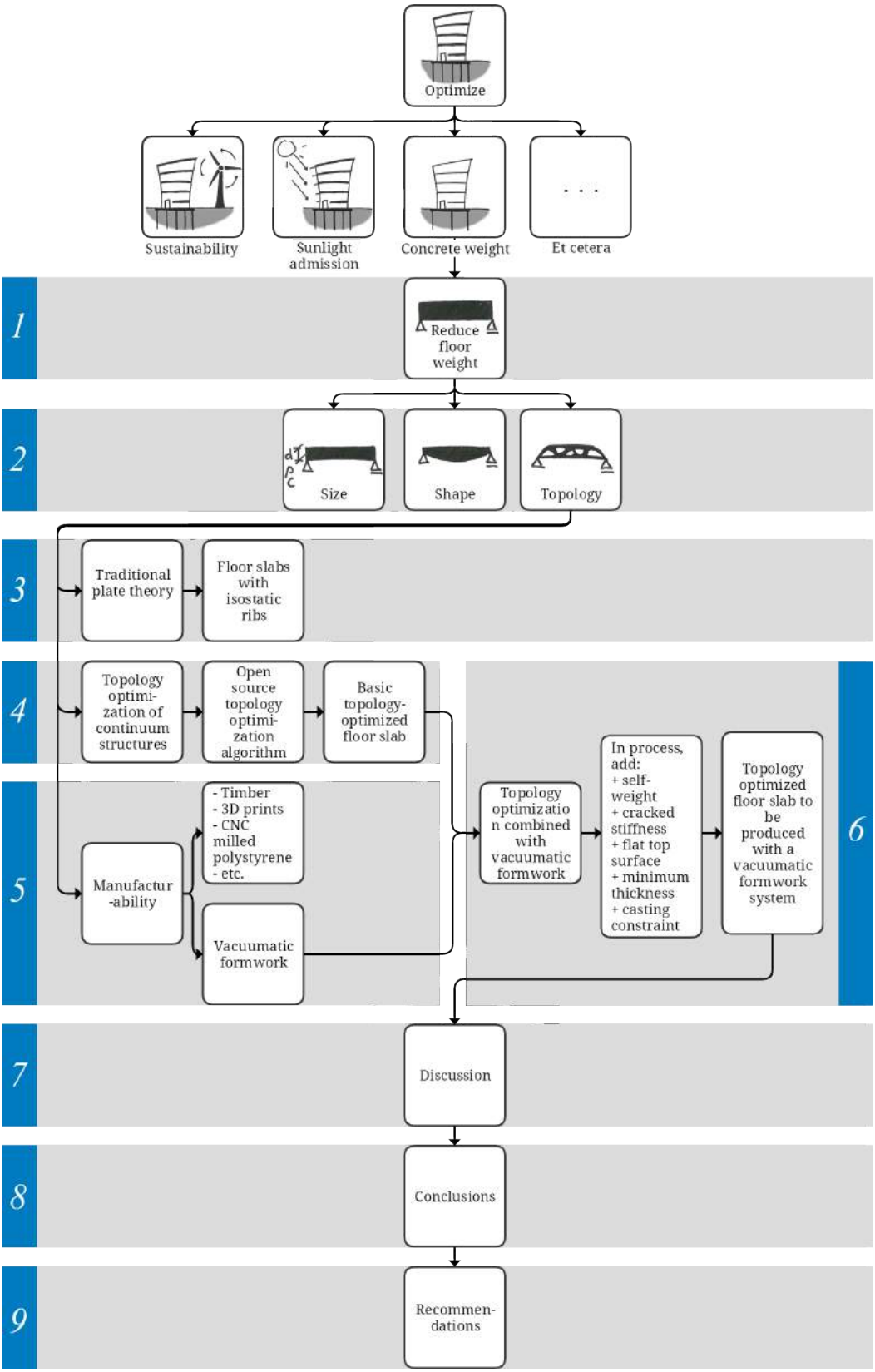



***Figure 1.2:*** *Thesis structure and chapters*

# 2

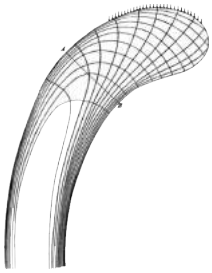
# Structural optimization

According to Merriam-Webster (2015), the definition of optimization is the act, process or methodology of making something (e.g., a design, system, or decision) as fully perfect, functional, or effective as possible. In nature, this can refer to the optimal conditions of, for example, light, temperature, or altitude for a species to thrive. As in nature, in architecture, the optimal conditions are also sought-after.

The process of optimization is about finding an optimal solution within a given model. This model can be, among others, biological, architectural, or structural and is often restricted by boundary conditions (Burry and Burry, 2010). Structural optimization and biological evolution are closer connected than one might think. According to Skedros and Baucom (2007), an example can be found in the way bone grows: tiny beams inside of bone called trabeculae seem to follow the stress trajectories.

When applying optimization to the design of a structure, we are basically asking ourselves: when given a design space with loads and boundary conditions, how should

***(a)*** *Photograph (Yale Bone Lab, nd)*



***(b)*** *Stress trajectories (Meyer, 1867)*

***Figure 2.1:*** *Bone trabeculae*

the material be distributed within this space in order to achieve the most efficient structure, in accordance with the optimization goal?

First, the structural model needs to be established using design variables (or parameters) when starting the structural optimization process. These parameters can be related to geometry, such as height or thickness, but also to material properties or other design aspects. Next, boundary conditions define the domain of the solution. Lastly, the optimal solution depends on the objective the engineer wishes to reach. Examples include the minimization of weight or the maximization of stiffness. Combinations of objective functions are also possible in a multi-objective optimization.

After making this parametric model and defining boundary conditions, the structural optimization process can begin. Using optimization tools each design iteration can be checked according to the constraints and parameters until the 'best' design follows.

The Iron Triangle is a model of showing how all projects are dependent on three attributes: time, quality, and cost. This triple constraint symbolizes how one side of the triangle cannot be improved without negatively affecting the others. Similar to the Iron Triangle, three factors need to be taken into account when optimizing a structural model. First, there needs to be knowledge of the problem at hand. This knowledge can be used to increase the *optimization algorithm efficiency*. Each model takes a certain *time* to run. After the algorithm is finished the *quality* of the result can be measured. This triangle is visualized in Figure 2.2. From the figure we can conclude that concessions need to be made when developing a new optimization algorithm: it is either too slow, not smart enough, or the found solution is not the optimal one. This does not mean structural optimization is without merit, however, just that the perfect optimization algorithm does not exist.

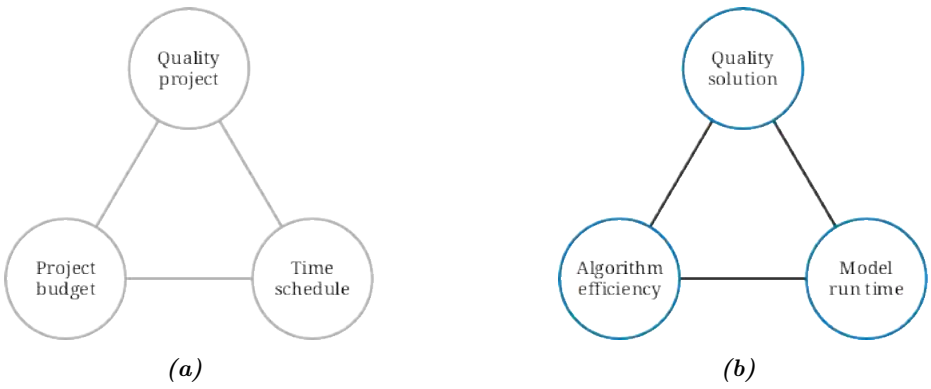



***Figure 2.2:*** *(a) Iron Triangle; (b) Optimization Triangle*

## 2.1 STRUCTURAL OPTIMIZATION TYPES

The structural optimization process can be subdivided intro three different types: member cross-section, shape, and topology optimization. All types can be applied on various scales and in all design phases, but some are more effective than others

or more reflect the choices to be made in this phase. Using a truss bridge as an example, each type is visualized in Figure 2.3. This diagram also shows the relation between the ease of formulation and the generality of each optimization type. It indicates that a topology optimization problem has a general application, but as a result, the problem can be challenging to define.

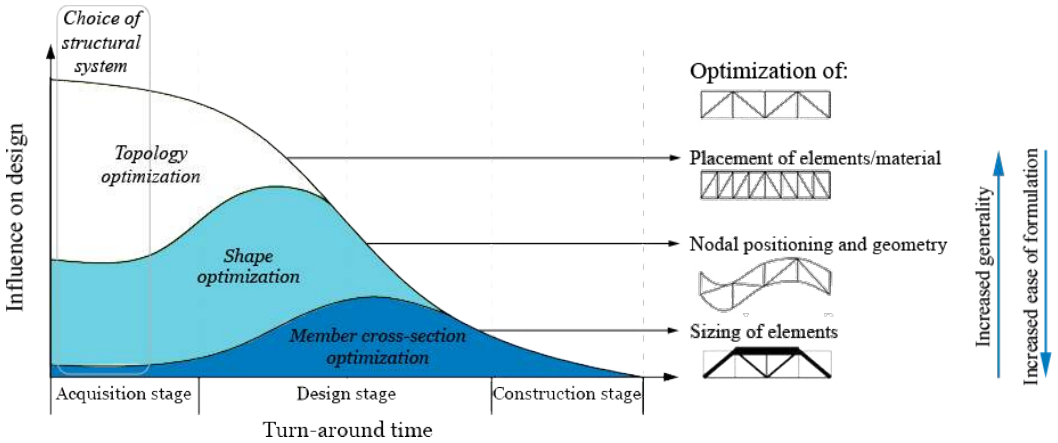



***Figure 2.3:*** *Optimization types per phase, examples of each optimization type (ABT bv, 2014), and simplicity versus generality (EDC, 2015)*

### 2.1.1 Member cross-section optimization

This type of optimization is best used near the end of a project, as shown in Figure 2.3, when the shape of the structure is already determined. By changing the shape or size of a cross-section, small gains can be made, which ultimately add up. This type can also be referred to as sizing optimization.

#### National Military Museum

The roof structure of the National Military Museum in Soesterberg, The Netherlands (Figure 2.4) is an example where member cross-section optimization has been utilized. In the acquisition stage, the design space was already determined by the architect. After optimizing the center to center distance and height of the truss (topology and shape optimization respectively), there were still gains to be made in the member cross-sections.

***Figure 2.4:*** *Nationaal Militair Museum (Uponor, 2015; Nationaal Militair Museum, 2015)*

That is where ABT bv further optimized the structure, resulting in a weight reduction of 40 kg per square meter compared to the first design, which translates to a cost-saving of 2.5 million euros, as reported by Henket (2013).

### 2.1.2 Shape optimization

Shape optimization refers to changes in nodal positioning and element surface/curvature. Both shape and topology optimization are most effective in an earlier design stage. At the beginning of a project, the final shape or topology is not yet entirely defined so this means that substantial changes to the design will have smaller consequences for the project costs.

#### Communication tower

The communication tower near Geleen (Figure 2.5) is a project that had its shape optimized by ABT bv in collaboration with BroekBakema architecture. After the topology was set by the architect the exact curvature of the structure was determined in an optimization process. This involved structural challenges, like fatigue, but also practical constraints, such as the size of the staircase. After the shape, the structure was also optimized in terms of the member's cross-section.

***Figure 2.5:*** *Communication tower (BroekBakema, 2014; Living Projects, 2014)*

### 2.1.3 Topology optimization

This refers to the placement of members and material in relation to the flow of forces and stress levels. It is the broadest of the optimization types and, therefore, maximizes its use in the earliest design phase. Several examples from practice are mentioned in Section 2.5. In a basic sense, the topology optimization model consists of three formulations:

- *The design space* is the area in which the optimal solution must be found, and it can be divided into either pixels, voxels, or beams;
- *The optimization objective* is the type of optimal solution, which can, for example, be based on stress or stiffness;
- *The optimization constraint* indicates the fraction of the design space that is allowed to be filled.

The process of finding an optimal solution using these formulations can be described as a hill climbing problem. It can be visualized with a surface area and corresponding axes, a landscape on top of this plane, and a search algorithm that attempts to find the top of the highest hill. The top of the hill symbolizes the point where the

parameters are such that the optimization objective reaches its greatest potential. Figure 2.6 shows how in topology optimization three components are necessary to find a solution:

- *The model* describes the design space in which the solution may be found, the way in which this space is divided, and the parameters that are allowed to change. These parameters give the x, y-axis.
- *The landscape* is defined by a formula, which is based on the optimization objective. This formula may be called $f$ and is dependent on the parameters $x$, $y$.
- *The optimization method* describes the method of finding the highest point and thus the optimal solution. In other words: finding the optimal values of $x$, $y$ for which $f$ is maximized.

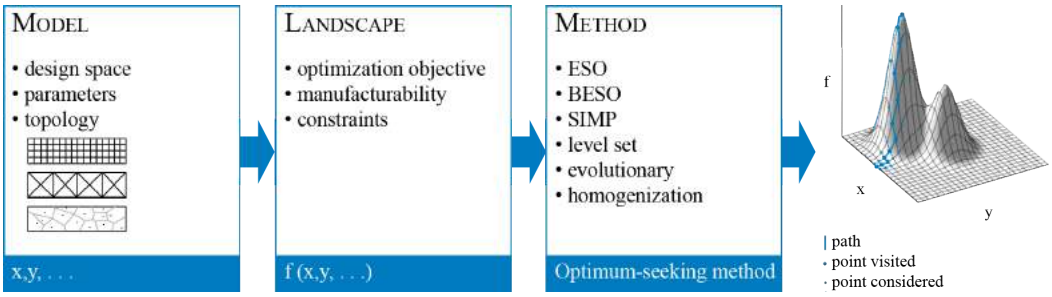
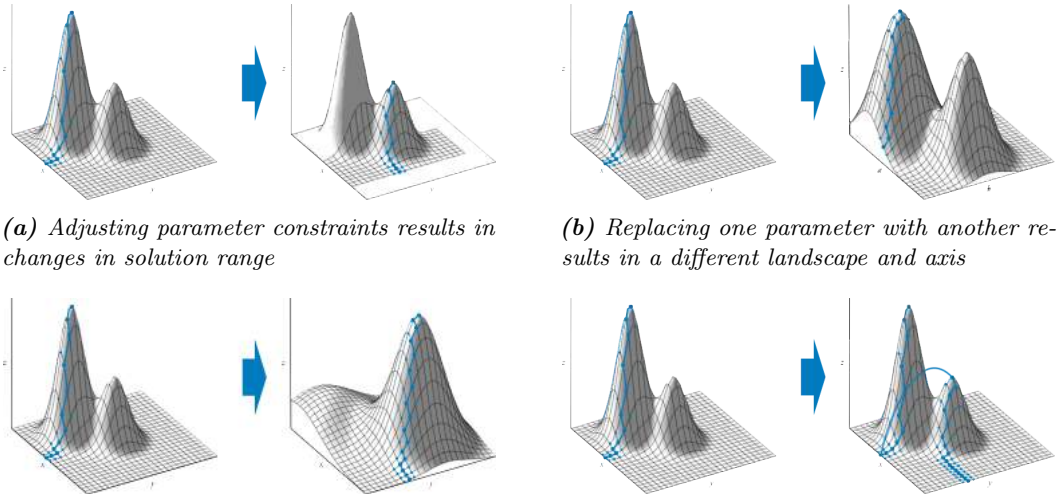



***Figure 2.6:*** *Topology optimization and the required components*

Figure 2.7 visualizes the way modifying certain parts of the components may affect the outcome of the optimization algorithm using the method of steepest ascent as search method. It should be noted that the problem in this thesis is a lot more complex than the examples of Figure 2.6 and Figure 2.7. There are more parameters, and the landscape will consist of many hills making it more challenging to find the global maximum instead of a local maximum. Because of the complexity of this problem, the triangle visualized in Figure 2.2 plays a greater role. The calculation time can become lengthy and as such knowledge of the problem and applying a smart algorithm is important.

## 2.2 TOPOLOGY OPTIMIZATION MODEL

The model for a structural topology optimization problem can be defined in three different ways, namely as a space of virtual elements, a truss, or a Voronoi diagram. Figure 2.6 shows these options in the left-hand box. Which one of these is the most effective in solving the requirements depends on the used production technique or material. The first option can add and subtract elements in order to reach an optimum. The Voronoi model optimizes the location of the nodes such that the resulting Voronoi diagram functions the best, according to the optimization objective. The truss model is more commonly referred to as the growing ground structure method, and it allows for truss elements to be turned on or off in a fixed grid.



*(a) Adjusting parameter constraints results in changes in solution range*

*(b) Replacing one parameter with another results in a different landscape and axis*

*(c) Modifying the optimization objective changes the landscape to find a solution which better suits the new objective*

*(d) Modifying the search method results in a change in the way the optimum is found and sometimes a different optimum. Here, simulated annealing is included*

***Figure 2.7:*** *Adjustments in the components and the corresponding changes in the hill climbing model using the method of steepest ascent*

## 2.3 TOPOLOGY OPTIMIZATION LANDSCAPE

The optimization objective displayed in the second column of Figure 2.6, decides which values of the parameters result in the 'best' solution. The value of each combination of parameters (as related to the optimization objective) can be visualized with a landscape. Subsequently, a search method described in Section 2.4 can find an optimum.

## 2.4 TOPOLOGY OPTIMIZATION METHODS

### 2.4.1 Evolutionary Structure Optimization method

The Evolutionary Structure Optimization (ESO) method by Xie and Steven (1993) removes material from the design space with each iteration, resulting in an optimum where the remaining material is the most efficient. By combining the ESO method with a finite element analysis, the optimization is usually based on maximum stress. The specific stress level of an element in the model is usually based on the Von Mises stress, which is a method to translate the complex stresses of an element into one number. In the Von Mises yield criterion, this stress is then compared to the material's yield strength (Burns, 2002). When then optimization objective is based on overall stiffness, then compliance is considered. Compliance is the inverse of the overall stiffness of the model and it represents the total strain energy of the structure by the applied loading (Huang and Xie, 2010). When an element turns from a solid to an empty element, the change of mean compliance is the same as the strain energy, which in turn is called the elemental sensitivity number. The elements with the lowest elemental sensitivity number are removed with each iteration.

The ESO method is a simple but quick concept that can easily be linked to finite element method software. The output of such a model shows a clear design with no porous elements, only empty or fully solid. There are several disadvantages, however. Since material can only be removed, this means that an element erased in an early iteration might still be of use in the final design. The ESO method, therefore, finds a local optimum, instead of the global optimum (see Figure 2.8). The Bi-directional ESO method aims to alleviate this problem by allowing the model to recover material after it has been deleted.

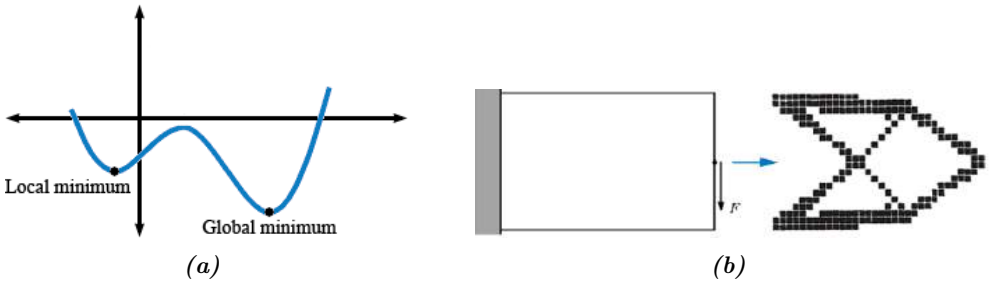



***Figure 2.8:*** *(a) Global vs. local minimum; (b) An example of an ESO model showing the division of the model in elements (Huang and Xie, 2010)*

### 2.4.2 Bi-directional Evolutionary Structure Optimization method

The Bi-directional Evolutionary Structure Optimization (BESO) method, also by Xie and Steven (1997), is similar to the ESO method except that material can be both added and discarded. For the optimal stress design, this means that the elements with the lowest Von Mises stresses are removed using a Rejection Ratio while the empty elements close to the maximum Von Mises stresses are returned to their fully solid state using an Inclusion Ratio. The elemental sensitivity number, which is used in the ESO method on solid elements, is also used for the empty elements in the BESO method. Figure 2.9 shows an example of the BESO method.

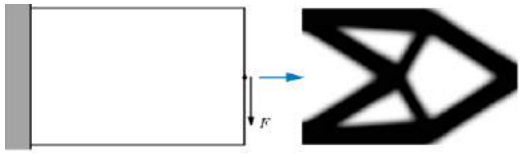



***Figure 2.9:*** *An example of a BESO model showing the porous elements (Huang and Xie, 2010)*

The Extended ESO (XESO) method is a variation on the BESO method which has the optimization goal of minimizing the stress levels (Ohmori et al., 2005). It has been used in the design of several constructed buildings and as such has proven itself to be a valuable BESO method. These buildings include the Akutagawa River Side Project in Japan and the Qatar Education City which will both be discussed later in Section 2.5.

### 2.4.3 Solid Isotropic Material with Penalization method

The Solid Isotropic Material with Penalization (SIMP) method by Bendsøe (1989) does not remove elements that have a relatively low stress, but it changes their stiffness to such a small number the element either practically has no use anymore or it becomes porous. It is regarded as an efficient and effective method and has proven itself in a wide range of models. A disadvantage compared to other methods is that usually a local optimum is reached. This local optimum is very near the global optimum, so for practical purposes, this does not necessarily pose a problem. Similar to the ESO and BESO topology optimization method, an elemental sensitivity number is utilized in each iteration to determine which elements should increase their stiffness, and which ones should decrease their stiffness. The SIMP method is applied in the developed tool in this thesis. Chapter 4 elaborates on this choice.

### 2.4.4 Ant colony method

The ant colony optimization (ACO) concept by Flint (2008) is based on the behavior of ants. When looking for food, ants cover an area at random and upon finding a meal, they leave behind pheromones. These pheromones attract other ants, which also leave behind pheromones creating a positive feedback loop where more and more ants are drawn towards the food. When using ACO for structural problems the design space is colonized by artificial ants who travel every element and leave pheromones when a solution performs well. Flint (2008) has shown this is a feasible method to be used as the basis for a structural design.

### 2.4.5 Homogenization method

The homogenization method by Bendsøe and Sigmund (2003) also divides the design space into many elements, and like the SIMP method, it alters the binary problem into a continuum problem. In other words: the elements can also be porous instead of only full or empty. It differs from the SIMP method in the way the porous elements are modeled. Instead of adjusting the stiffness of an element, the homogenization method appoints each individual element as part full and part empty, so there is a hole in each one, shown in Figure 2.10. The hole size or porosity and orientation angle dictate the stiffness of an element in an iterative process.

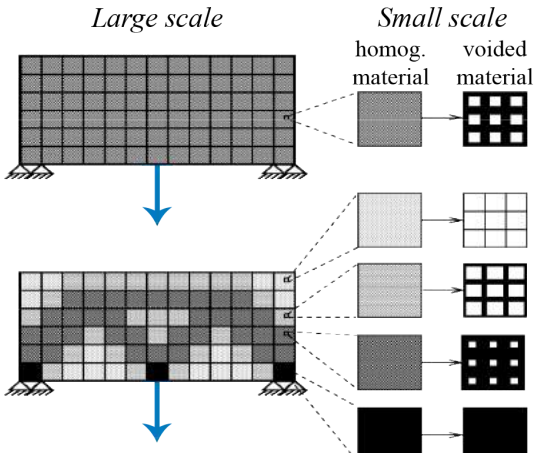



***Figure 2.10:*** *An example of a homogenization model (Orlando and Luege, 2014)*

### 2.4.6 Level set method

This is also widely accepted as a useful method and Huang and Xie (2010) summarize it as a steepest descent method using a shape sensitivity analysis as well as the Hamilton-Jacobi equation. Figure 2.11 shows how this method works.

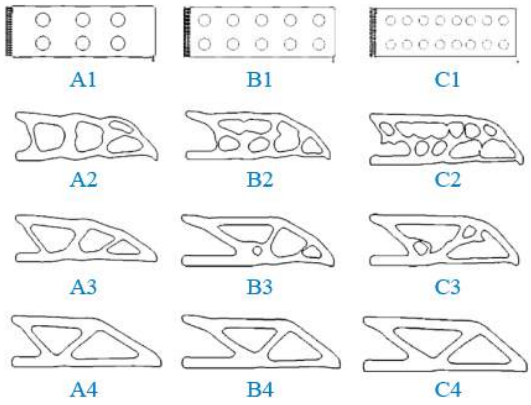



***Figure 2.11:*** *An example of a level set model (Huang and Xie, 2010)*

## 2.5 TOPOLOGY OPTIMIZATION IN PRACTICE

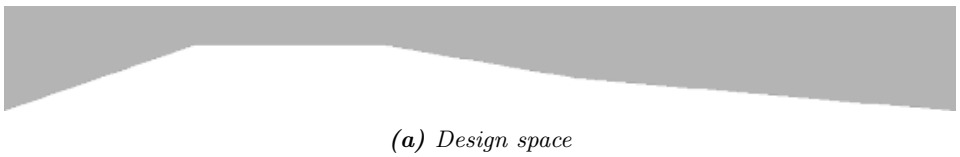
### 2.5.1 National Military Museum

The National Military Museum in Soesterberg, The Netherlands, also mentioned in Section 2.1.1, is an example of a structure that had its topology optimized. The spans and spacing (vertical and horizontal) between the members of the truss, based on orthogonal placement, while keeping the design space constant, is a result of this analysis.

### 2.5.2 Civil Engineering & Geosciences hall

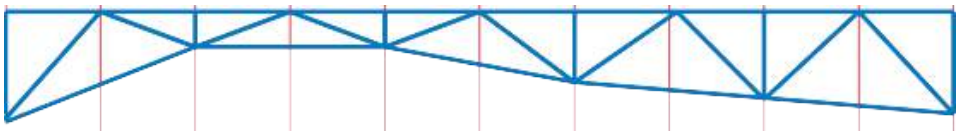
Another example based on topology optimization is the design of the hall between the lecture rooms of the Civil Engineering & Geosciences building in Delft (Figure 2.12). The building itself was built in the sixties and as part of a redevelopment project in 2012 the new roof would create a new space between the already existing lecture rooms. This project is currently on hold, but several designs were made, including a preliminary design with a traditional Pratt truss by ABT bv and a topology-optimized truss by Van der Ploeg (2013) using the SIMP method. Both designs also had their member cross-sections optimized. Compared to the preliminary design his final configuration in Figure 2.13 lowered both the number of elements and the number of connections and reduced self-weight with 1000 kg, which is more than 10% of its original weight.

***Figure 2.12:*** *Render of the new CiTG hall (OIII architecten, 2012)*



***(a)*** *Design space*

***(b)*** *Topology-optimized structure*

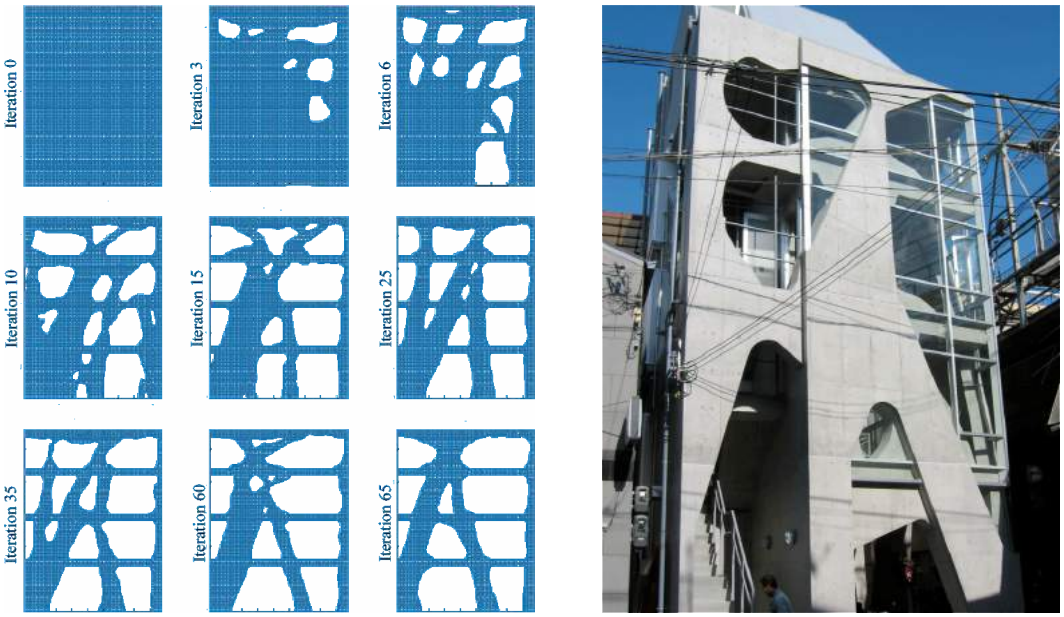


***(c)*** *Inspired configuration structure*

***Figure 2.13:*** *The design process of the CiTG hall (Van der Ploeg, 2013)*

### 2.5.3 Akutagawa River Side Project

The Akutagawa River Side Project is part of a shopping mall in Takatsuki, Japan. The mall has been realized as part of a redevelopment plan because of the large growth of this commuter town. Two walls of the building have an organic shape based on the topology-optimized design using the Extended ESO method. Dead weight acted as the vertical load while earthquakes acted as the horizontal load. The boundary conditions include the floors of the building and the east side wall, which both were not allowed to be removed. In the final iteration, it is clear they are both included in the design presented in Figure 2.14.



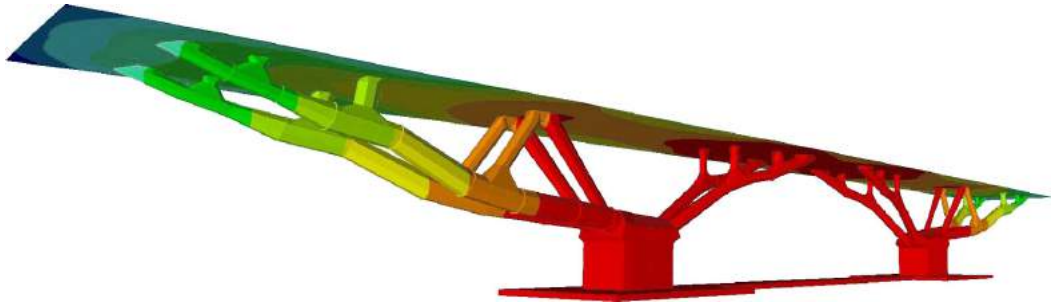

**(a)** *Design process (south wall)* **(b)** *Resulting construction (west wall)*

***Figure 2.14:*** *Akutagawa River Side Project (Ohmori et al., 2005)*

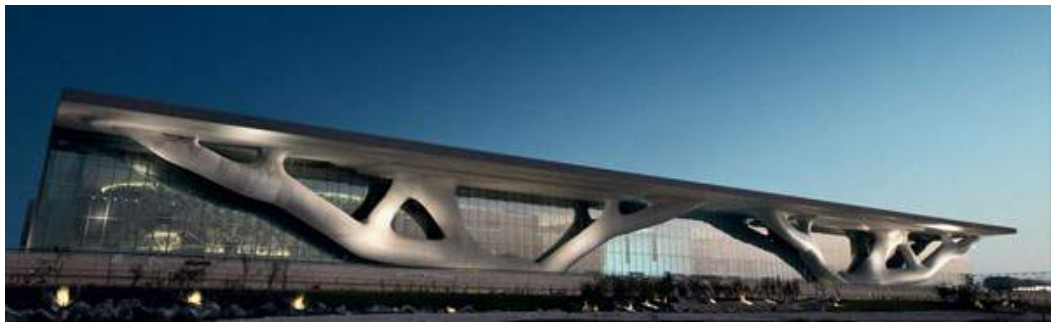
### 2.5.4 Qatar Education City

The Extended ESO method has been successfully introduced in the Qatar Education City (QEC) in Doha, Qatar. The huge structure forms the entrance to the 250 m wide convention center. The design is based on the Sidra tree, which symbolizes knowledge of the divine (Burry and Burry, 2010). The support of the roof resembles the native plant, but structurally it is also efficient since the shape almost exactly follows the topology-optimized design as can be seen in Figure 2.15. The smooth organic surface of the support is not the load-bearing structure, however. The actual structure is hidden inside and was multi-objective optimized where the goal was to keep the aesthetic outer surface very close, but at the same time keep a straight structural profile over as long a distance as possible. The result is visible in Figure 2.15b.

***(a)*** *Several iterations with the Extended ESO method (Sasaki, 2008)*



***(b)*** *Structure of QEC entrance (Burry and Burry, 2010)*



***(c)*** *Picture of QEC entrance (wikimapia, nd)*

***Figure 2.15:*** *Qatar Education City entrance*

# 3

# Ribbed floor slab systems

A structure consists of several elementary building components. Space for living, working, etc. can be created by combining columns, walls, beams, and slabs. Columns and/or load bearing walls serve as the vertical supports. Horizontal spans are traversed by floors, which have multiple functions that all need to be considered when designing a structure. There are structural, functional, building physical, and durability requirements. Structurally, floors have strength, stiffness, and stability requirements. Functionally, floors need to be level and have a smooth finish, while at the same time allowing space for installations. The building physics require soundproofing, insulation, and fire resistance. A durable floor needs to be resistant to exposure from light, wind, and water. In this report, the only requirements the floor will need to fulfill are the structural ones. The others are not focused on or otherwise neglected.

Traditional, monolithic floor slabs can have the disadvantage of being structurally inefficient, so, naturally, there already exist solutions for producing light-weight floor slabs. In a general sense, floor systems achieve this by adopting ribs that increase the cross-sectional stiffness due to their height, while at the same time removing material in between. Examples of external ribs or open structures include the waffle slab and TT-slab. Internal ribs are employed by, e.g., hollow core slabs and BubbleDeck floors. Appendix A describes several concrete floor systems that are often utilized, nowadays.

## 3.1 FLOOR SLABS WITH ISOSTATIC RIBS

An alternative to the modern floor slab systems in Appendix A are floor slabs with isostatic ribs. Several examples are displayed in Section 3.3. These designs are based on the same principal of the tiny beams inside of bone called trabeculae, mentioned earlier in the introduction to Chapter 2 and displayed in Figure 2.1. According to Culmann (1875), the trabeculae are oriented towards the principal stress trajectories.

Just as with bones, plates also generate principal stress trajectories when loaded. According to traditional plate theory, a two-dimensional plate under stress from a bending moment produces two types of orthogonal curves. These curves are called isostatics and follow the principal bending moment trajectories where the torsion is equal to zero.

The first person to truly combine the theory of trabeculae and apply it to a structure was the Italian structural artist Pier Luigi Nervi (1891-1979). He was an acclaimed engineer, famed for his highly effective structures that were constructed using minimum means while still allowing for aesthetics at the same time. He graduated from the University of Bologna in 1913 before founding a construction company ten years later. His initial work mostly consisted of contracting, but he especially excelled from the 1950s onward when he started working as a structural designer.

While the isostatic aligned ribs are an integral part of Nervi's floors, it was actually one of his colleagues, Aldo Arcangeli, who proposed this structural improvement. According to his findings, when the ribs in a floor are aligned to the isostatics, then the floor is at its most efficient when given the same load and support conditions.

In comparison, topology optimization has a greater design freedom than the theory behind the ribbed floor slabs with isostatic ribs and can be applied to a greater range of structural challenges. While floor slabs with isostatic ribs are, therefore, fundamentally different from topology optimized designs, for a floor slab, it can be seen as an alternative.

## 3.2 PRODUCTION METHOD

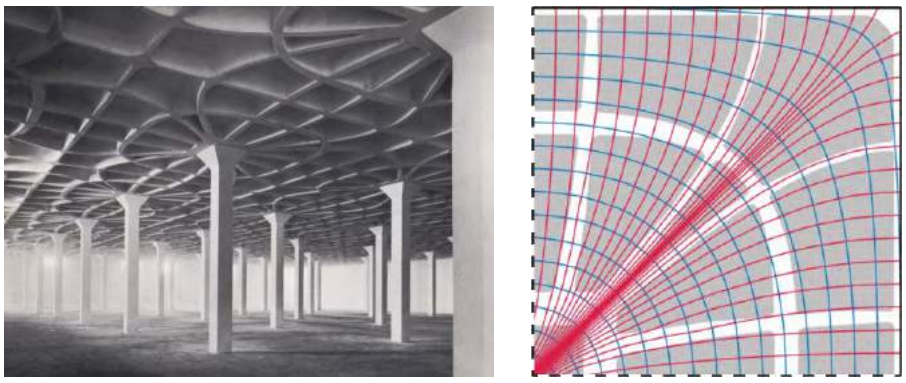
Pier Luigi Nervi realized prefabrication has several advantages when compared to in-situ concrete: the reduction of building time and the possibility of mass production. His research on the feasibility of prefabrication was mostly focused on slabs, vaults, and domes (Powell, 2011). Nervi's ribbed floor slabs are actually very similar to the waffle slab mentioned in Appendix A except that in the waffle slab design the ribs are all square because it makes production easier. This forces the loads around corners, which is not necessarily a problem. However, a more organic shape guides the loads in a more natural way to the nearest support. This results in a lighter structure and more architectural value.

Producing the organic shapes is a challenge that Nervi solved by making use of ferrocement, which can be seen as the precursor of reinforced concrete. Ferrocement is a composite material consisting of several layers of metal reinforcement, such as wire mesh covered with enough cement to cover it all (Chiorino, 2012). It is not the same as regular reinforced concrete because it is highly elastic and moldable. Ferrocement was used to make prefabricated concrete elements in plaster casts. Regular reinforcement was placed inside the ferrocement elements acting as a permanent mold and afterward they were covered in concrete creating one single slab. This seemingly complex method of producing floors was actually a cheap option as steel usage was restricted from 1935. Iori and Poretti (2005) published that the Fascist Italian administration envisioned a self-sufficient nation, which meant rationing the import of steel.

## 3.3 DESIGNS WITH ISOSTATICS

### 3.3.1 Gatti wool factory

Nervi's first application of the floor system with isostatic ribs was the Gatti wool factory in 1951, where the floor was required to span a big distance and to support heavy loading from wool-spinning machines (Gargiani, 2012). The floor, visible in Figure 3.1, was made using the ferrocement molds explained in Section 3.2 and each column carries a $5 \times 5$ meter part of the floor slab. The image on the right of Figure 3.1 shows the maximum principal bending moments in red and the minimum principal bending moments in blue. It can be concluded that the floor of the Gatti wool factory shows a close resemblance to the isostatics.

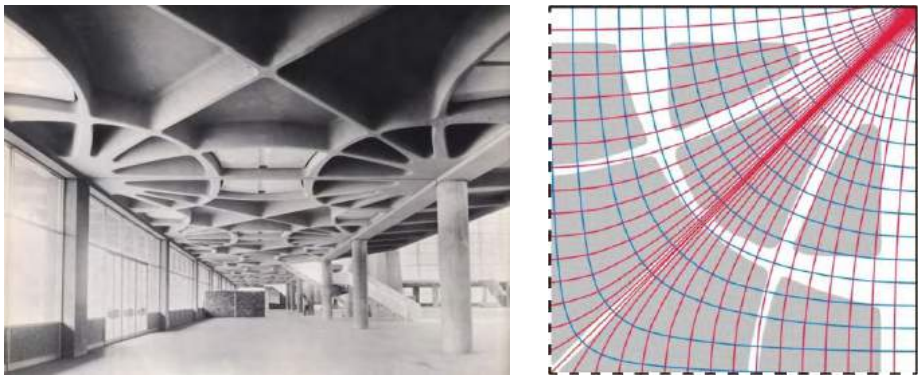


***(a)*** *Photograph* ***(b)*** *Isostatics*

***Figure 3.1:*** *Floor of the Gatti wool factory (Nervi, 1963; Halpern et al., 2013)*

### 3.3.2 Palazzo del Lavoro

Built in 1961 for the Turin exhibition which celebrated Italy's unification Palazzo del Lavoro also features a ribbed floor slab by Nervi. Figure 3.2 shows the outer perimeter of the building where this floor is situated and the similarities between the structure and isostatics.



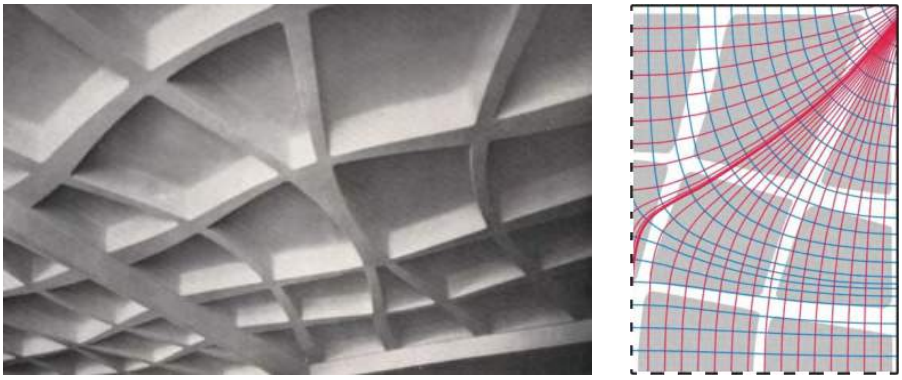
***(a)*** *Photograph* ***(b)*** *Isostatics*

***Figure 3.2:*** *Floor of the Palazzo del Lavoro (Nervi, 1963; Halpern et al., 2013)*

Nervi's floor was selected in the competitive tender because of its quick building time (Sharp, 2002). Like the Gatti wool factory, it consists of a square floor plan supported by columns. The center to center distance is increased to 10 meters, however, and the shape of the isostatic ribs is also slightly different.

### 3.3.3 Palazzo dello Sport

Constructed along with the more renowned Palazzetto dello Sport, the Palazzo dello Sport was built in 1961 to host the Olympic basketball tournaments. The ribbed floors can be found in the outer perimeter of the building. The difference of the slabs in the sports arena compared to the Gatti and Lavoro floors is the rectangular shape of the plan. This small adjustment results in a drastic change in isostatics in Figure 3.3. While the maximum and minimum principal bending moments don't follow the structure exactly, according to Halpern et al. (2013), it shows Pier Luigi Nervi's own creativity in designing the floor.

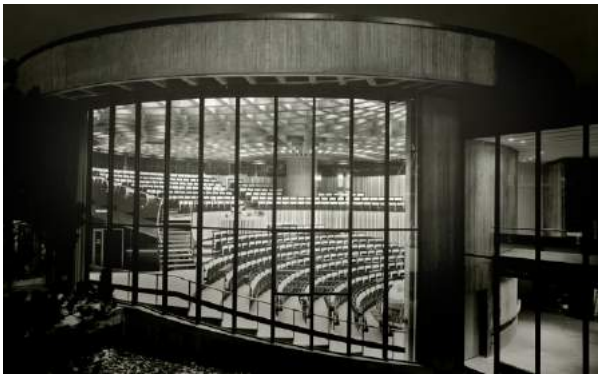


***(a)*** *Photograph* ***(b)*** *Isostatics*

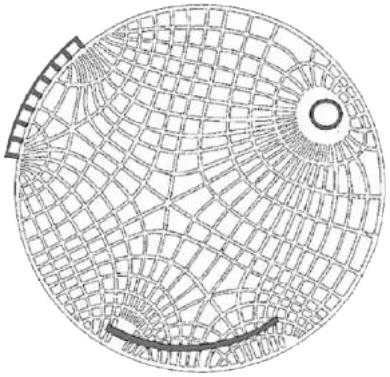
***Figure 3.3:*** *Floor of the Palazzo dello Sport (Nervi, 1965; Halpern et al., 2013)*

### 3.3.4 Former zoology lecture hall at Freiburg University

The architect of the ceiling in the former zoology lecture hall at Freiburg University was inspired by the floors Nervi had designed. Displayed in Figure 3.4, the floor was built in the 1960s with isostatic ribs to minimize material input. The design space is clearly more complex than the other examples since it combines a round shape with several different support types. Antony et al. (2014) report that this floor is significantly more sustainable in terms of greenhouse gas emissions and uses less concrete and reinforcement steel than modern alternatives.



*(a) Photograph*



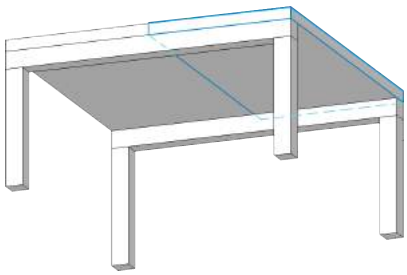
*(b) Floor plan*

***Figure 3.4:*** *Floor of the former zoology lecture hall at Freiburg University (Antony et al., 2014)*

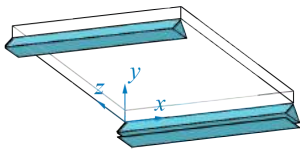
## 3.4 ISOSTATICS FOR THE CONSIDERED FLOOR SLAB

Figure 3.5 displays the shape and boundary conditions of the plate that has been topology optimized in the next chapter. In this section, the theory of isostatic ribs beneath a thin floor slab been applied to this considered plate. Differences between these two methods are described more extensively in Chapter 6.

In the beginning of last century, Stephen Timoshenko published several books on mechanics and materials. His work can be viewed as the foundation of plate theory. In this section, the theoretical stresses in a plate are derived, calculated, and plotted, with the help of his book on plates and shells. First, the theory behind a simple beam is addressed, which is the most obvious simplification of a plate. From there, the step to a two-dimensional plate and the principal stress trajectories for the plate considered in Chapter 6 can be made. In accordance with the design method of isostatics, the principal stress lines can be used to make predictions about an optimal topology of the floor slab.



*(a) Context*

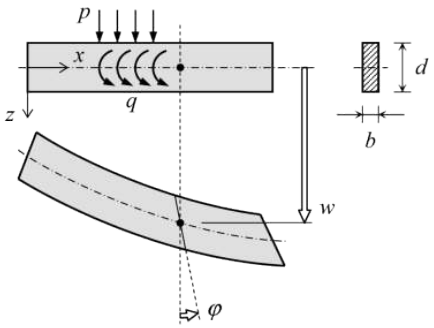



*(b) Model*

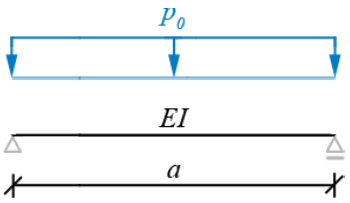
***Figure 3.5:*** *Design space of the floor with line supports*

### 3.4.1 Two-dimensional beam

A beam differs from a slab in the way the loads are transferred to the supports. While a slab can carry loads in two directions, a beam spans only one direction. The considered beam is displayed in Figure 3.6a, with a certain width $b$ and depth $d$. At this point, it is assumed there are no normal forces because $w << d$, flat cross-sections stay flat, and the Poisson's ratio is neglected. The beam has degrees of freedom $\phi$ and $w$ and the loads are a distributed load $p$ and distributed torque $q$. The loads cause a bending moment and a shear force, which in turn cause a curvature and shear deformation. This relationship is displayed in Equation (3.1). The relation between the deformations and the degrees of freedom are called the kinematic equations. Next, the relation between the degrees of freedom and stress resultants, are called the constitutive equations. Lastly, the relations between the stress and loads are defined in the equilibrium equations. For slender beams, the shear deformation can be neglected. So the equations are simplified in Equation (3.2).




*(a) Definition variables (Blaauwendraad, 2010)*




*(b) Considered problem for a floor beam*

***Figure 3.6:*** *Two dimensional beam*

$$\begin{Bmatrix} w \\ \phi \end{Bmatrix} \xrightarrow[equations]{kinematic} \begin{Bmatrix} \kappa \\ \gamma \end{Bmatrix} \xrightarrow[equations]{constitutive} \begin{Bmatrix} M \\ V \end{Bmatrix} \xrightarrow[equations]{equilibrium} \begin{Bmatrix} p \\ q \end{Bmatrix} \tag{3.1}$$

$$\{w\} \xrightarrow[equations]{kinematic} \{\kappa\} \xrightarrow[equations]{constitutive} \{M\} \xrightarrow[equations]{equilibrium} \{p\} \tag{3.2}$$

Rotations are defined in Equation (3.3). From the rotations, the curvature is defined in Equation (3.4). These are the kinematic equations, and relate the deformations with the bending curvature $\kappa$. From the curvature, the stress resultant is defined in the constitutive relation Equation (3.5). Here, the plate stiffness $D$ is introduced in order to account for the material properties of the slab. The equilibrium equations are the last step. They are defined in Equation (3.6).

$$\phi = -\frac{\delta w}{\delta x} \tag{3.3}$$

*Kinematic equation:*

$$\kappa = -\frac{\delta^2 w}{\delta x^2} \tag{3.4}$$

*Constitutive equation:*

$$M = EI\kappa \tag{3.5}$$

*Equilibrium equation:*

$$-\frac{\delta^2 M}{\delta x^2} = p \tag{3.6}$$

By combining all these relationships, the fourth-order differential equation may be stated in Equation (3.7).

$$EI\frac{\delta^4 w}{\delta x^4} = p \tag{3.7}$$

Assuming the problem of a floor is as displayed in Figure 3.6b, according to Blaauwendraad (2006), Equation (3.8) can be assumed for the deflection $w$. The boundary conditions are defined in Equation (3.9). By combining Equation (3.10) and Equation (3.7), the unknown value of $C$ can be solved in Equation (3.11), resulting in the deflection function in Equation (3.12).

$$w(x) = C\left(x^4 - 2ax^3 + a^3x\right) \tag{3.8}$$

$$\begin{aligned} &\text{at } x = 0 \quad \begin{cases} w = 0 \\ M = -EI\dfrac{\delta^2 w}{\delta x^2} = 0 \end{cases} \\ &\text{at } x = a \quad \begin{cases} w = 0 \\ M = -EI\dfrac{\delta^2 w}{\delta x^2} = 0 \end{cases} \end{aligned} \tag{3.9}$$

$$\begin{aligned} \frac{\delta w}{\delta x} &= C\left(4x^3 - 6ax^2 + a^3\right) \\ \frac{\delta^2 w}{\delta x^2} &= C\left(12x^2 - 12ax\right) \\ \frac{\delta^3 w}{\delta x^3} &= C\left(24x - 12a\right) \\ \frac{\delta^4 w}{\delta x^4} &= C \cdot 24 \end{aligned} \tag{3.10}$$

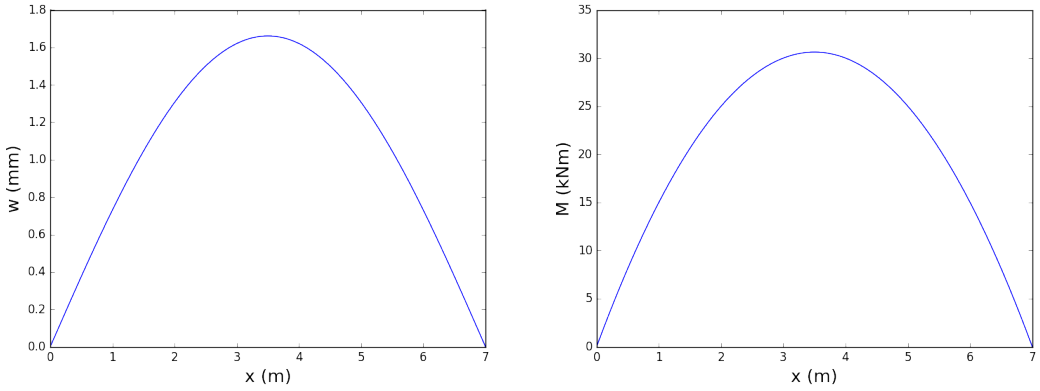
$$\begin{aligned} EI \cdot C \cdot 24 &= p_0 \\ C &= \frac{p_0}{24EI} \end{aligned} \tag{3.11}$$

$$w_1(x) = \frac{p_0}{24EI}\left(x^4 - 2ax^3 + a^3x\right) \tag{3.12}$$

The final function for the deflection in Equation (3.12) conforms to the boundary conditions, as displayed in Equation (3.13).

$$\begin{aligned} w_1(0) &= \frac{p_0}{24EI}\left(0^4 - 2a0^3 + a^30\right) = 0 \\ w_1(a) &= \frac{p_0}{24EI}\left(a^4 - 2a^4 + a^4\right) = 0 \\ M(0) &= -EI \cdot \frac{p_0}{24EI}\left(12 \cdot 0^2 - 12a0\right) = 0 \\ M(a) &= -EI \cdot \frac{p_0}{24EI}\left(12a^2 - 12a^2\right) = 0 \end{aligned} \tag{3.13}$$

At this point, the results for a beam can be plotted in Figure 3.7. The span is assumed to be 7 m, the distributed load 5.0 kN/m, the Young's modulus 37.000 MPa, and the second moment of area $\frac{1}{12}bh^3 = \frac{1}{12}1000 \cdot 312.5^3$. For these values, the maximum deflection is found to be 1.66 mm, which also conforms to the result found by the so-called *vergeet-mij-nietjes* (in Dutch). The maximum bending moment is found to be 30.625 kN m, which conforms to the theoretical maximum bending moment of $\frac{1}{8}pa^2$.



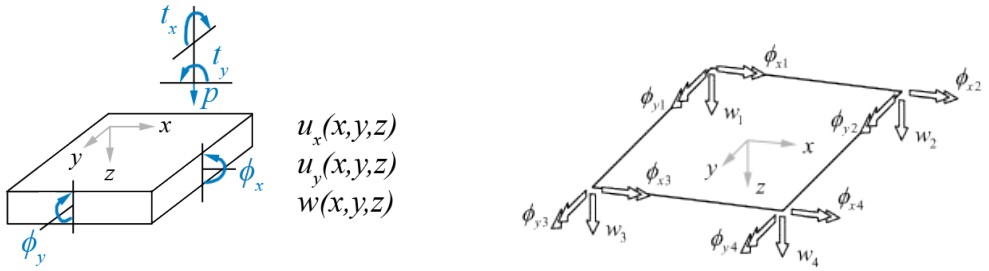

***(a)*** *Deflection* ***(b)*** *Bending moment*

***Figure 3.7:*** *Results for the beam*

### 3.4.2 Three-dimensional slab

This section will expand on the theory behind slabs, which is a type of plate that is loaded perpendicular to its plane. The stresses in these slabs follow from bending moments and shear forces. According to traditional plate theory, two models can be considered: a thin slab or a thick slab. Since the deflection is much smaller than the height of the slab ($w << h$), the thin slab theory will be applied. Several assumptions follow:

- Flat cross-sections stay flat after applying a load;
- Shear deformation is neglected, so $t_x = t_y = 0$;
- Deflections are independent of the z-coordinate;
- No stress $\sigma_{zz}$ (Figure 3.8).




*(a) Slab* *(b) FEM element*

***Figure 3.8:*** *Sign conventions (Blaauwendraad, 2006)*

The relation between the deformations and the degrees of freedom are called the kinematic equations, again. Next, the relation between the degrees of freedom and stress resultants, are called the constitutive equations. Lastly, the relations between the stress and loads are defined in the equilibrium equations, as displayed in Equation (3.14) for general problems. These equations are simplified for thin plates in Equation (3.15).

$$\begin{Bmatrix} w \\ \phi_x \\ \phi_y \end{Bmatrix} \xrightarrow[equations]{kinematic} \begin{Bmatrix} \kappa_{xx} \\ \kappa_{yy} \\ \rho_{xy} \\ \gamma_x \\ \gamma_y \end{Bmatrix} \xrightarrow[equations]{constitutive} \begin{Bmatrix} m_{xx} \\ m_{yy} \\ m_{xy} \\ v_x \\ v_y \end{Bmatrix} \xrightarrow[equations]{equilibrium} \begin{Bmatrix} p \\ q_x \\ q_y \end{Bmatrix} \tag{3.14}$$

$$\{w\} \xrightarrow[equations]{kinematic} \begin{Bmatrix} \kappa_{xx} \\ \kappa_{yy} \\ \rho_{xy} \end{Bmatrix} \xrightarrow[equations]{constitutive} \begin{Bmatrix} m_{xx} \\ m_{yy} \\ m_{xy} \end{Bmatrix} \xrightarrow[equations]{equilibrium} \{p\} \tag{3.15}$$

Rotations are defined in Equation (3.16). From the rotations, three curvatures are defined in Equation (3.17). These are the kinematic equations, and relate the deformations with the bending curvatures $\kappa$ and torsional curvature $\rho$. From the three curvatures, the stress resultants are defined in the constitutive relation Equation (3.18). Here, the plate stiffness $D$ is introduced in order to account for the material properties of the slab. The equilibrium equations are the last step. They are defined in Equation (3.19) and follow from Figure 3.9.

$$\begin{aligned} \phi_x &= -\frac{\delta w}{\delta y} \\ \phi_y &= -\frac{\delta w}{\delta x} \end{aligned} \tag{3.16}$$

*Kinematic equations:*

$$\begin{aligned} \kappa_{xx} &= -\frac{\delta^2 w}{\delta x^2} \\ \kappa_{yy} &= -\frac{\delta^2 w}{\delta y^2} \\ \rho_{xy} &= -2\frac{\delta^2 w}{\delta x \delta y} = 2\kappa_{xy} \end{aligned} \tag{3.17}$$

*Constitutive equations:*

$$\begin{aligned} m_{xx} &= D(\kappa_{xx} + \nu\kappa_{yy}) \\ m_{yy} &= D(\kappa_{yy} + \nu\kappa_{xx}) \\ m_{xy} &= D(1-\nu)\kappa_{xy} \\ \text{In which: } D &= \frac{Et^3}{12(1-\nu^2)} \end{aligned} \tag{3.18}$$

*Equilibrium equations:*

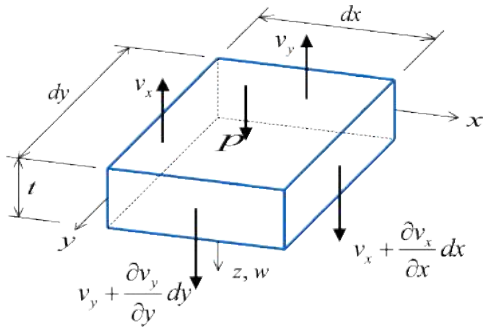
$$\begin{aligned} v_x &= \frac{\delta m_{xx}}{\delta x} + \frac{\delta m_{yx}}{\delta y} \\ v_y &= \frac{\delta m_{yy}}{\delta y} + \frac{\delta m_{xy}}{\delta x} \\ \frac{\delta v_x}{\delta x} &+ \frac{\delta v_y}{\delta y} + p = 0 \\ \text{Resulting in: } &- \left( \frac{\delta^2 m_{xx}}{\delta x^2} + 2\frac{\delta^2 m_{xy}}{\delta x \delta y} + \frac{\delta^2 m_{yy}}{\delta y^2} \right) = p \end{aligned} \tag{3.19}$$

By combining all these relationships, the biharmonic equation may be stated in Equation (3.20). This equation was formulated by Lagrange in 1811.

$$D\left( \frac{\delta^4 w}{\delta x^4} + 2\frac{\delta^4 w}{\delta x^2 \delta y^2} + \frac{\delta^4 w}{\delta y^4} \right) = p \tag{3.20}$$

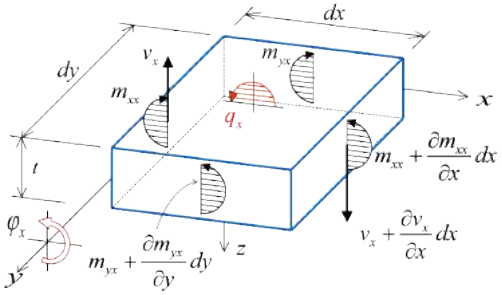
Now that the biharmonic equation is defined, finding the deformation function is the next step in the process, similar to how the two-dimensional problem was solved in the previous section. Levy (1899) recommends finding the solution via the series in Equation (3.21). Here, $Y_m$ is a function of $y$. The slab is displayed in Figure 3.10, with hinged supports at two opposing sides. Therefore, it can already be stated that Equation (3.21) conforms to the boundary conditions in Equation (3.22). Now, $Y_m$ needs to be determined in order to conform to the boundary conditions at $y = 0.5b$ and $y = -0.5b$ and to Equation (3.20).

$$w = \sum_{m=1}^{\infty} Y_m \sin\left(\frac{m\pi x}{a}\right) \tag{3.21}$$



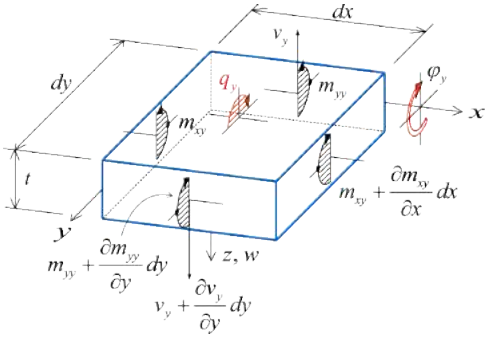

$$\sum f_z = 0 \;:\; \left(\frac{\partial v_x}{\partial x}dx\right)dy + \left(\frac{\partial v_y}{\partial y}dy\right)dx + p\,dx\,dy = 0$$

*(a) Vertical forces*




$$\sum m_y = 0 \;:\; \left(\frac{\partial m_{xx}}{\partial x}dx\right)dy + \left(\frac{\partial m_{yx}}{\partial y}dy\right)dx - \left(v_x dy\right)dx + q_x\,dx\,dy = 0$$

*(c) Moments over y*




$$\sum m_x = 0 \;:\; \left(\frac{\partial m_{yy}}{\partial y}dy\right)dx + \left(\frac{\partial m_{xy}}{\partial x}dx\right)dy - \left(v_y dx\right)dy + q_y\,dx\,dy = 0$$

*(e) Moments over x*

***Figure 3.9:*** *Equilibrium of an element (Blaauwendraad, 2006)*

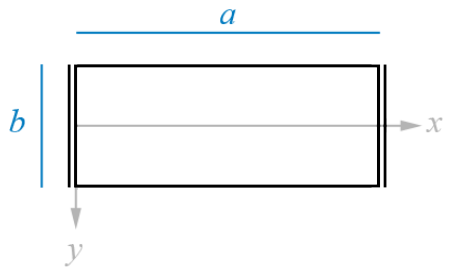



***Figure 3.10:*** *Considered problem for the slab, top-down perspective*

$$\begin{aligned} &\text{at } x = 0 \quad \begin{cases} w = 0 \\ \dfrac{\delta^2 w}{\delta x^2} = 0 \end{cases} \\ &\text{at } x = a \quad \begin{cases} w = 0 \\ \dfrac{\delta^2 w}{\delta x^2} = 0 \end{cases} \end{aligned} \tag{3.22}$$

According to Nadai (1915), the solution of the biharmonic equation can be simplified into Equation (3.23). Here, $w_1$ is the solution of the middle strip at $y = 0$, which has already been defined in Equation (3.12), Section 3.4.1. This solution of the middle strip has been rewritten as a trigonometric series in Equation (3.24) by Timoshenko (1956). Since $w_1$ already satisfies both the boundary conditions at $x = 0$ and $x = a$ and the biharmonic equation, $w_2$ needs to be chosen so it complies with Equation (3.25). By taking $w_2$ like the series in Equation (3.21), in which $m = 1, 3, 5...$, and applying it in Equation (3.25), Equation (3.26) is determined.

$$w = w_1 + w_2 \tag{3.23}$$

$$w_1(x) = \frac{p_0}{24EI}\left(x^4 - 2ax^3 + a^3x\right) = \frac{4p_0a^4}{\pi^5 D}\sum_{m=1}^{\infty}\frac{1}{m^5}\sin\frac{m\pi x}{a} \tag{3.24}$$

$$\frac{\delta^4 w_2}{\delta x^4} + 2\frac{\delta^4 w_2}{\delta x^2 \delta y^2} + \frac{\delta^4 w_2}{\delta y^4} = 0 \tag{3.25}$$

$$\begin{aligned} &\sum_{m=1}^{\infty}\left(Y_m'''' - 2\frac{m^2\pi^2}{a^2}Y_m'' + \frac{m^4\pi^4}{a^4}Y_m\right)\sin\frac{m\pi x}{a} = 0 \\ &\text{Therefore: } Y_m'''' - 2\frac{m^2\pi^2}{a^2}Y_m'' + \frac{m^4\pi^4}{a^4}Y_m = 0 \end{aligned} \tag{3.26}$$

Timoshenko and Woinowsky-Krieger (1959) suggest that by integrating Equation (3.26), Equation (3.27) can be derived. Because the plate in Figure 3.10 is symmetrical over the x-axis, only the even expressions need to be preserved. Therefore, it can be assumed that $C_m = D_m = 0$. Finally, by substituting Equation (3.27) into Equation (3.26), and Equation (3.26) into Equation (3.23), the total deflection is found in Equation (3.28).

$$\begin{aligned} Y_m = \frac{pa^4}{D}\Big(&A_m\cosh\frac{m\pi y}{a} + B_m\frac{m\pi y}{a}\sinh\frac{m\pi y}{a} + \\ &C_m\sinh\frac{m\pi y}{a} + D_m\frac{m\pi y}{a}\cosh\frac{m\pi y}{a}\Big) \end{aligned} \tag{3.27}$$

$$w = w_1 + w_2 = \frac{4p_0a^4}{\pi^5 D}\sum_{m=1}^{\infty}\frac{1}{m^5}\sin\frac{m\pi x}{a} + \frac{pa^4}{D}\sum_{m=1}^{\infty}\left(A_m\cosh\frac{m\pi y}{a} + B_m\frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right)\sin\frac{m\pi x}{a} \tag{3.28}$$

Only the integration constants of $A_m$ and $B_m$ still need to be determined. They follow from the boundary conditions set in Equation (3.29). Since the two remaining edges are free, both the bending moments in the y-direction and the sum of the shear force and derivative of the torsional moments are nil. From the boundary conditions, Timoshenko and Woinowsky-Krieger (1959) solved the integration constants that are displayed in Equation (3.30). The final deflection function is finally displayed in Equation (3.31).

$$\text{at } y = \pm\tfrac{1}{2}b \quad \begin{cases} m_{yy} = \dfrac{\delta^2 w}{\delta y^2} + \nu\dfrac{\delta^2 w}{\delta x^2} = 0 \\ v_y + \dfrac{\delta m_{yx}}{\delta y} = D\left(\dfrac{\delta^3 w}{\delta y^3} + (2-\nu)\dfrac{\delta^3 w}{\delta x^2 \delta y}\right) = 0 \end{cases} \tag{3.29}$$

$$A_m = \frac{4}{m^5\pi^5}\cdot\frac{\nu(1+\nu)\sinh\alpha_m - \nu(1-\nu)\alpha_m\cosh\alpha_m}{(3+\nu)(1-\nu)\sinh\alpha_m\cosh\alpha_m - (1-\nu)^2\alpha_m}$$
$$B_m = \frac{4}{m^5\pi^5}\cdot\frac{\nu(1-\nu)\sinh\alpha_m}{(3+\nu)(1-\nu)\sinh\alpha_m\cosh\alpha_m - (1-\nu)^2\alpha_m} \tag{3.30}$$
$$\text{In which: } \alpha_m = \frac{m\pi b}{2a}$$

$$w = w_1 + w_2 = \frac{p_0a^4}{D}\sum_{m=1}^{\infty}\left(\frac{4}{\pi^5 m^5} + A_m\cosh\frac{m\pi y}{a} + B_m\frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right)\sin\frac{m\pi x}{a} \tag{3.31}$$

When applying Equation (3.31) to Equation (3.32) and Equation (3.33), the results that are displayed in Figure 3.11 can be found. The maximum deflection $w_{max}$ is obtained at the center of the span in the x-direction and at the boundaries in the y-direction. It measures 1.68 mm. The distributions of the bending moments can be derived from Equation (3.18). They are presented in Equation (3.32) and Figure 3.11. The figure shows that the boundary conditions have been satisfied for $x = 0$ and $x = a$ since both the deflection and the bending moment is equal to zero. Additionally, the shear forces are obtained via Equation (3.33). They are also displayed in Figure 3.11, and they show that the boundary conditions at $y = \pm\frac{b}{2}$ are correct. The validation of the results is displayed in Table 3.1 and Figure 3.17, which has been performed by modeling the slab in Scia Engineer using Kirchhoff theory.

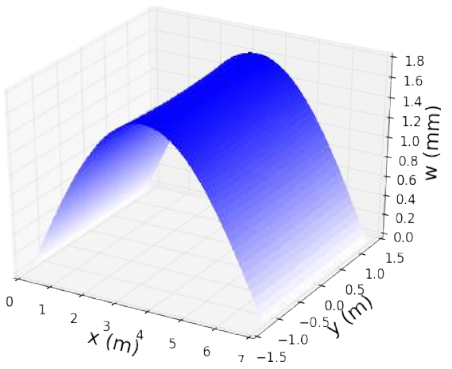
The shapes of the diagrams in Figure 3.17 are similar to the ones in Figure 3.11 and the minimum and maximum values in Table 3.1 are also practically the same.

$$\begin{aligned} m_{xx} &= -D\left(\frac{\delta^2 w}{\delta x^2} + \nu\frac{\delta^2 w}{\delta y^2}\right) \\ m_{yy} &= -D\left(\nu\frac{\delta^2 w}{\delta x^2} + \frac{\delta^2 w}{\delta y^2}\right) \\ m_{xy} &= -(1-\nu)D\frac{\delta^2 w}{\delta x \delta y} \end{aligned} \tag{3.32}$$

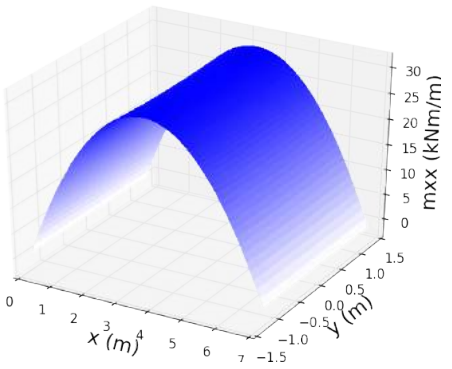
$$\begin{aligned} v_x &= D\left(\frac{\delta^3 w}{\delta x^3} + \frac{\delta^3 w}{\delta y^2 \delta x}\right) \\ v_y &= D\left(\frac{\delta^3 w}{\delta y^3} + \frac{\delta^3 w}{\delta x^2 \delta y}\right) \end{aligned} \tag{3.33}$$

***Table 3.1:*** *Validation of the results for the slab*

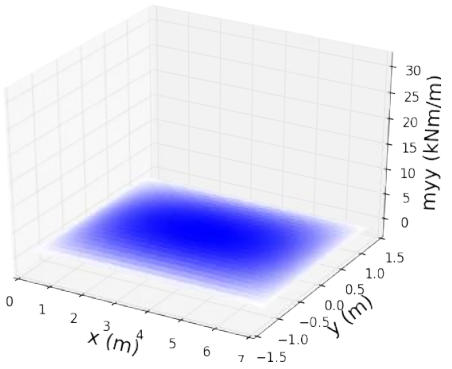
| | | Minimum plate theory | Minimum Scia Engineer | Maximum plate theory | Maximum Scia Engineer | Difference (%) |
|---|---|---|---|---|---|---|
| $w$ | (mm) | 0.00 | 0.00 | 1.68 | 1.68 | 0.00 |
| $m_{xx}$ | (kNm/m) | 0.00 | 0.00 | 30.95 | 30.95 | 0.00 |
| $m_{yy}$ | (kNm/m) | 0.00 | 0.00 | 1.22 | 1.22 | 0.00 |
| $m_{xy}$ | (kNm/m) | -2.73 | -2.73 | 2.73 | 2.73 | 0.00 |
| $v_x$ | (kN/m) | -15.50 | -15.49 | 15.50 | 15.49 | 0.06 |
| $v_y$ | (kN/m) | -0.99 | -0.95 | 0.99 | 0.95 | 4.21 |



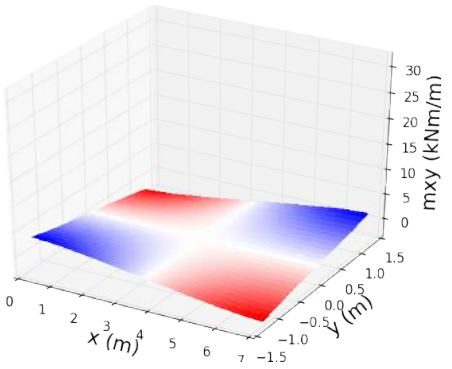

*(a) Deflection*



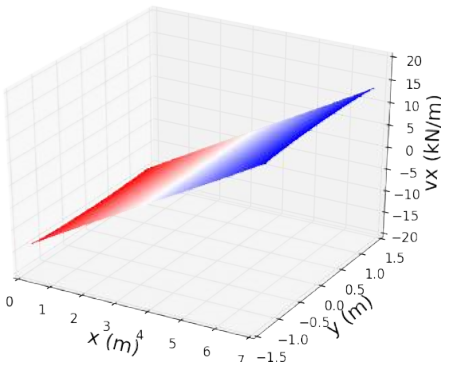

*(b) Bending moment $m_{xx}$*



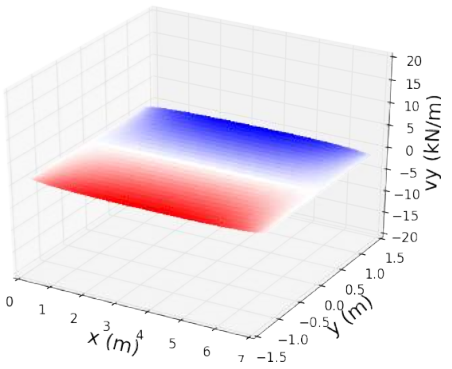

*(c) Bending moment $m_{yy}$*




*(d) Twisting moment $m_{xy}$*




*(e) Shear forces $v_x$*

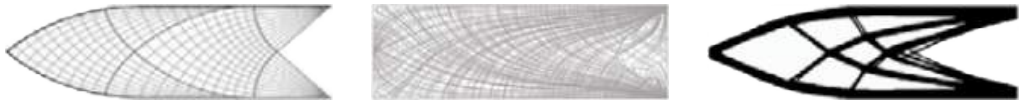



*(f) Shear forces $v_y$*

***Figure 3.11:*** *Results for the slab*

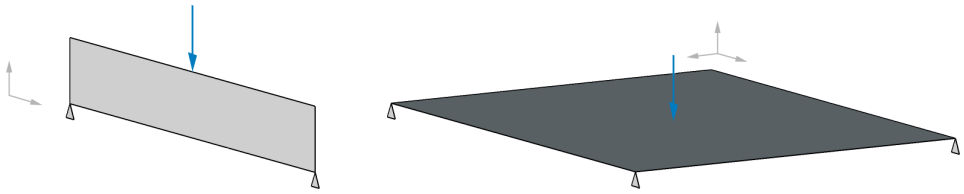
### 3.4.3 Principal stress trajectories

One of the first researchers in structural optimization was an Australian mechanical engineer called Anthony Michell. In 1904, he defined several optimal truss structures (Michell, 1904). The designs were formulated by placing the truss members in such a way that their strains are all equal, resulting in a minimization of the volume of material. His results very closely resemble the principal stress trajectories, as displayed in Figure 3.12. They are only applicable to two-dimensional problems, however. A floor is loaded perpendicular to its plane so, as Figure 3.13 illustrates, an additional dimension is required.



***(a)*** *Michell structure* ***(b)*** *Principal stress lines* ***(c)*** *Topology optimization*

***Figure 3.12:*** *Optimal results for a cantilever beam according to different theories (Tam et al., 2015)*



***(a)*** *In-plane* ***(b)*** *Out-of-plane*

***Figure 3.13:*** *Differences in loading*

As mentioned earlier, Pier Luigi Nervi and his colleague Aldo Arcangeli utilized principal stress lines as a design tool for their floors. The lines visualize the way forces flow to their supports. Therefore, according to them, when the ribs in a floor are aligned to these principal stress lines, the floor is at its most efficient when given the same loads and support conditions. Contrary to the Michell structures, this method also works for out-of-plane loading. There are several assumptions to be made at this point. First of all, the considered material is isotropic, and secondly, only the elastic range is regarded. While this might seem to be contradictory to reinforced concrete, Nervi showed that even under these conditions, an efficient structure in concrete can still be realized.

As documented by Chen and Li (2010), with these assumptions, stress lines are not affected by the material stiffness or applied forces, but by the geometry of the design space and properties of the boundary conditions. Now that all the stresses in the slab have been determined in the last section, the principal stress trajectories according to traditional plate theory can be plotted. Up until now, the stresses have been calculated in the global x- and y-direction because it simplifies the calculation. By changing the angle $\alpha$ of an element, the stresses are transformed as displayed in Figure 3.14. For every element, there is a special value of this angle called $\alpha_0$, at which the shear stress is zero and the normal stresses are at their highest or lowest.

At this angle, the principal stresses $\sigma_1$ and $\sigma_2$ are found alongside their principal stress direction $\alpha_0$.

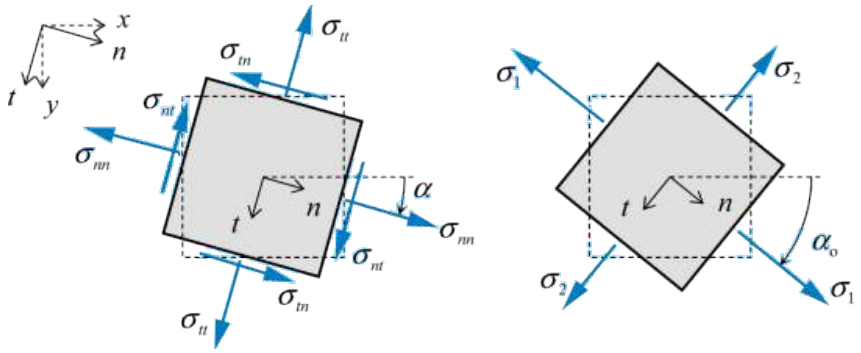


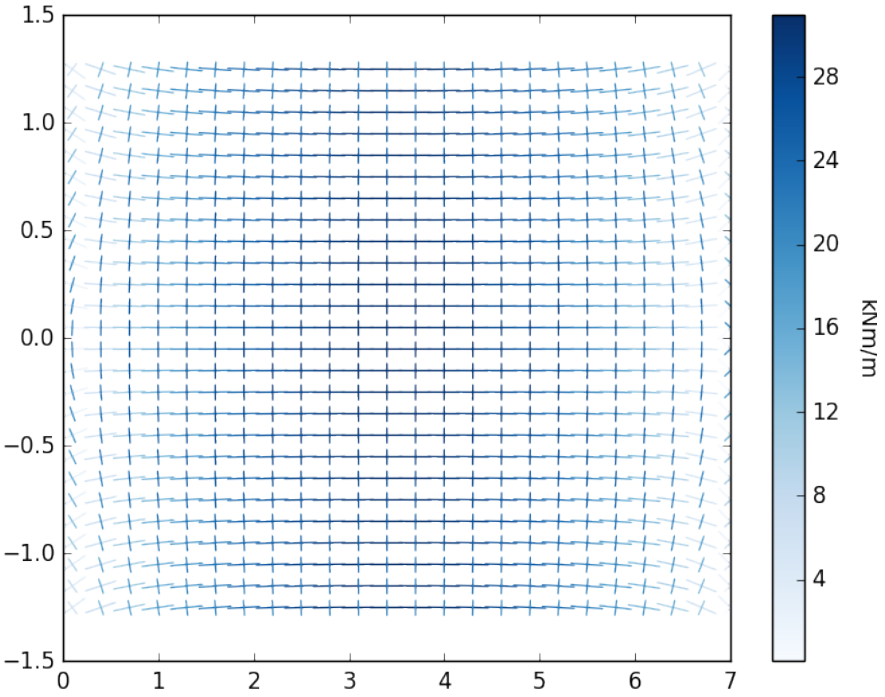

***Figure 3.14:*** *Principal direction $\alpha_0$ and corresponding stresses $\sigma_1$ and $\sigma_2$ (Blaauwendraad, 2010)*

This method also works for bending moments and shear forces. Then, the extreme values are called the principal moments and principal shear force, respectively. Blaauwendraad (2010) recommends using the expressions in Equation (3.34) and Equation (3.35). They follow from simple transformation rules based on Figure 3.14. The maximal principal moment $m_1$ always appears with a minimum principal moment $m_2$, but without a twisting moment. For the maximal principal shear force $v_1$, the accompanying minimal shear force is always equal to zero.

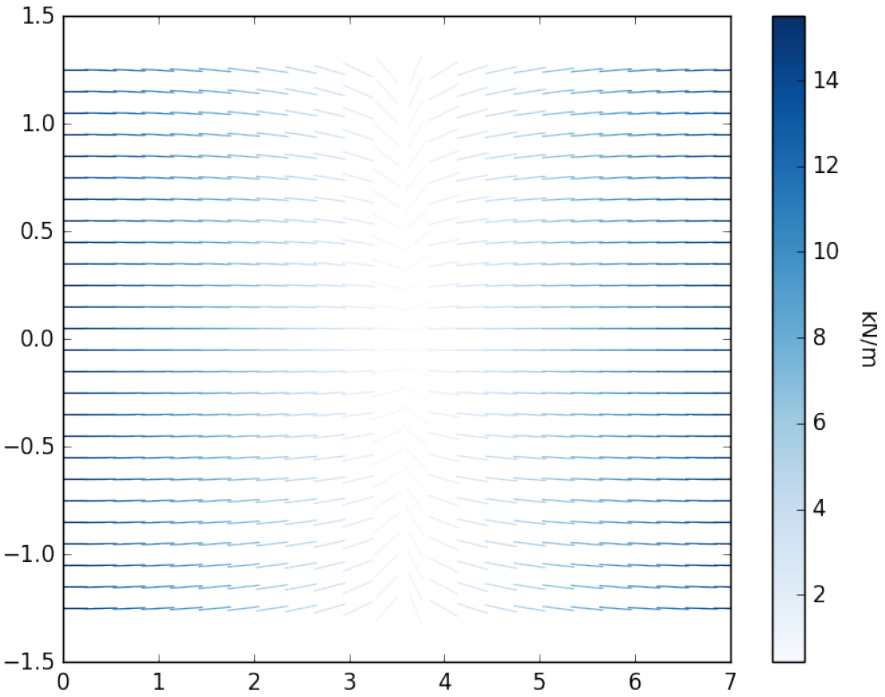
$$\begin{aligned} \tan(2\alpha_0) &= \frac{2m_{xy}}{m_{xx} - m_{yy}} \\ m_1 &= m_{xx}\cos^2\alpha_0 + m_{xy}\sin(2\alpha_0) + m_{yy}\sin^2\alpha_0 \\ m_2 &= m_{xx}\sin^2\alpha_0 - m_{xy}\sin(2\alpha_0) + m_{yy}\cos^2\alpha_0 \end{aligned} \tag{3.34}$$

$$\begin{aligned} \tan(\beta_0) &= \frac{v_y}{v_x} \\ v_1 &= \sqrt{v_x^2 + v_y^2} \\ v_2 &= 0 \end{aligned} \tag{3.35}$$

The resulting trajectories are displayed in Figure 3.15 and validated in Figure 3.17. The maximum principal bending trajectories are indicated by the longer dashes, the minimum principal bending trajectories by the shorter dashes. From them, predictions for an optimized floor slab design with isostatic ribs can be made. An interpretation has been sketched in Figure 3.16.




*(a) Bending moments*



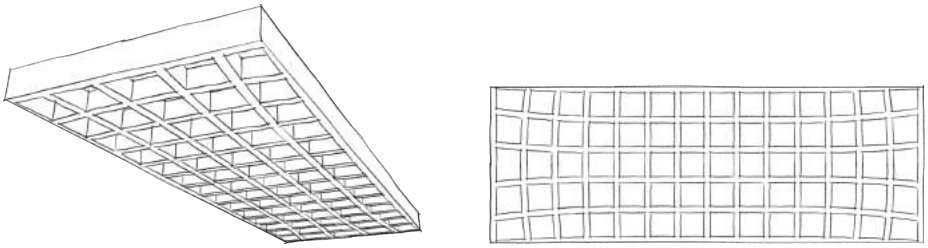

*(b) Shear forces*

***Figure 3.15:*** *Principal stress trajectories*

The predictions include:

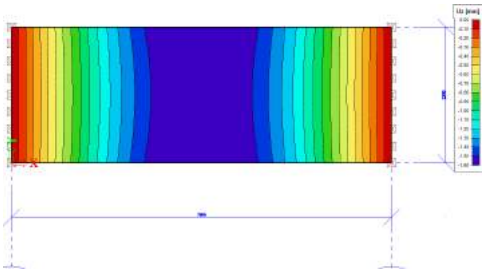
- Ribs are expected to 'flow' along the principal moment trajectories. For the considered design space and supports, this means that the main ribs will span between the supports in an almost straight manner. This is not surprising since TT-slabs, for example, show a similar design.
- Secondary ribs, following the minimum principal moment lines and perpendicular to the main ribs, are also expected according to the theory of isostatics.
- The main ribs are expected to be located near the sides of the slab where the bending moment in the main direction is the greatest.

The next chapter will elaborate on the topology optimization algorithm that has been developed for this thesis. It has been used to make a design for the same floor slab that was introduced in Section 3.4. The optimal result from topology optimization can be compared to the predictions from isostatics that have been made, here.

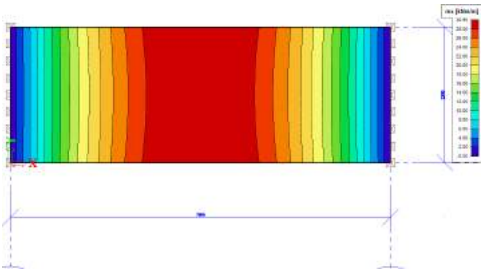


***(a)*** *Perspective* ***(b)*** *Top-down view*

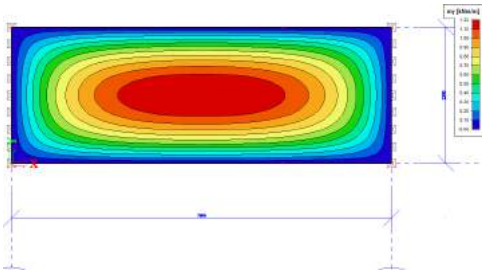
***Figure 3.16:*** *Interpretation of a ribbed floor slab with isostatic ribs for the considered design space*



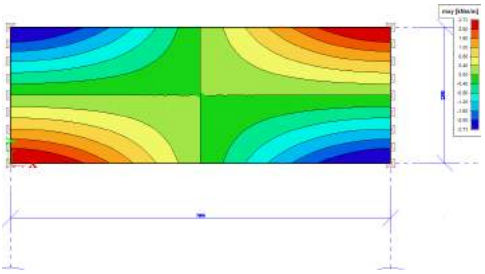
*(a) Deflection Uz*



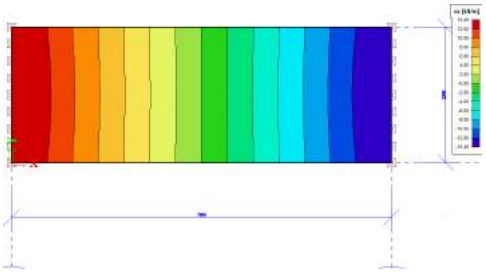
*(b) Bending moment mx*



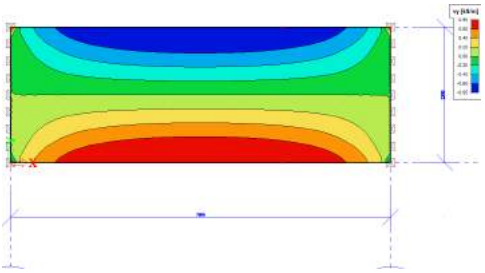
*(c) Bending moment my*



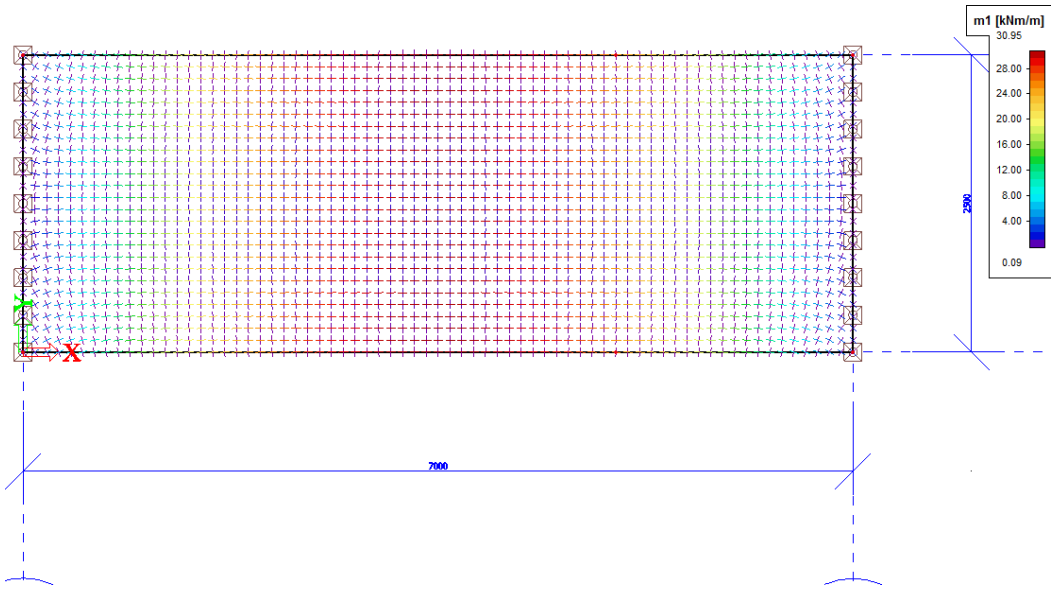
*(d) Torsional moment mxy*



*(e) Shear forces vx*



*(f) Shear forces vy*




*(g) Principal moment trajectories*

***Figure 3.17:*** *Validation in Scia Engineer*

# 4

# Topology optimization algorithm

As indicated in Chapter 2, there are many different ways to approach a topology optimization problem, such as ESO, BESO, SIMP, etc. There are also many different software packages available that use different methods. Furthermore, each method can be applied in both two- and three-dimensional problems. For this thesis, the combination of several requirements made the production of a new algorithm necessary, as their combination is not found in other software packages. The first requirement is the option of three-dimensional analysis. A simplification of the real scenario into a two-dimensional analysis would not include load dispersion in two directions and thus give a very unfavorable result. The second requirement is that the algorithm is available as open-source software. Closed-source software, such as Matlab, Rhinoceros, Altair, or Abaqus, is closed in a sense that they are closed to anyone who doesn't pay the steep price, but also, they are often not transparent. Transparency is essential for adding manufacturability constraints, for example.

Similar to the three-dimensional topology optimization algorithm presented in this thesis, are the studies and their resulting algorithms by Liu and Tovar (2014) and Hunter (2009). Compared to the work from Liu and Tovar, this algorithm is open-source. Compared to the work from Hunter, it is more transparent, consists of a single file only, and does not require additional software for viewing the result. It can be concluded that the advantages of the topology optimization algorithm presented in this report over other packages include:

- *Three-dimensional analysis*: for the given design problem of a floor, three-dimensional analysis is necessary.
- *Open-source software*: the algorithm is free to use for everyone.
- *Single file*: alternative open-source, three-dimensional topology optimization algorithms are complex and consist of multiple files and computer programs.

The topology optimization algorithm, introduced in Appendix F, is elaborated upon in this chapter. It builds on research by Sigmund (2001), Andreassen et al. (2011), and Liu and Tovar (2014). Figure 4.1 displays the topology optimization flowchart as it is applied in the algorithm. Each step is clearly indicated in the code, and will be explained in detail with the use of pseudo-code and an example. A small guide on how to get started is presented in Appendix E.

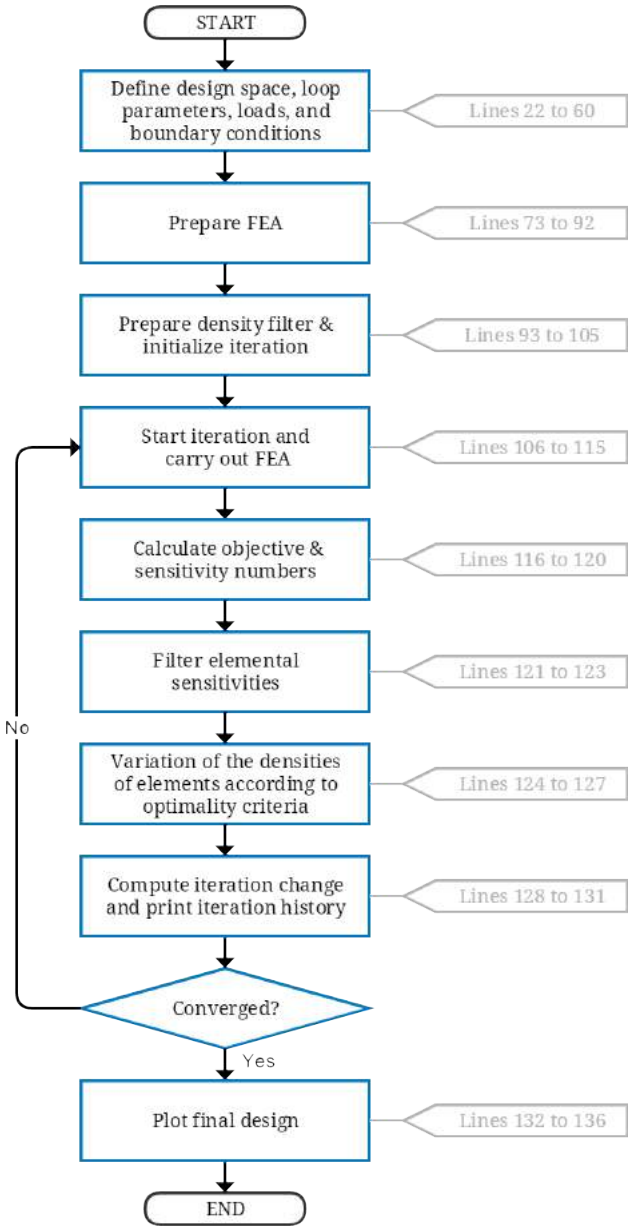



***Figure 4.1:*** *Topology optimization algorithm TopOpPy.py flowchart*

## 4.1 TOPOLOGY OPTIMIZATION METHOD AND ASSUMPTIONS

In Section 2.4, several different topology optimization methods were mentioned. These include ESO, BESO, SIMP, ant colony method, homogenization method, and level set method. The most common and proven methods are BESO by Xie and Steven (1997) and SIMP by Bendsøe (1989). Because of the way BESO works, it is called an evolutionary design method. According to Bendsøe and Sigmund (2003), evolutionary methods can lead to invalid results since gradient information (in the sensitivities) is used to formulate a discrete decision (is an element turned 'on' or 'off'). This is not an issue in the SIMP method since the gradient information of the sensitivities is applied to elements that also have gradient properties in their densities. Additionally, the SIMP method is easier to adapt to alternative objectives and Sigmund (2001) tells us it is equally computationally efficient. Therefore, the SIMP method is applied in the topology optimization algorithm of this thesis.

There are several assumptions made in the algorithm that need to be borne in mind. First, the design space that makes up the topology optimization problem is a rectangular cuboid or box. Secondly, the elements that make up the design space are cubes and, as is common in topology optimization, they have linear and isotropic properties. Both the shape of the design space and the elements simplify the numbering of the elements/nodes significantly in the Python script.

In a topology optimization problem, the objective is to place the material in such manner that the optimal solution is found. In the discretized version of the problem, this objective can be interpreted as the search of finding which elements are filled and which elements are empty. This 'on/off' problem is, in the SIMP method, replaced with continuous variables in the elemental stiffnesses combined with a penalty function that prevents intermediate stiffnesses in the optimal solution. Section 4.4 discusses the steps that are taken in the topology optimization algorithm in more detail with the help of a small example.

## 4.2 INPUT BY THE USER

The input of the algorithm consists of all the factors that may affect the end result. In topology optimization software this generally consists of the design space, volume fraction, load definition, boundary conditions, relative material properties, and, for this thesis, manufacturability constraints. Two factors that also influence the final result are the optimization objective and the optimization method. Here, they are fixed as compliance minimization and the SIMP method respectively. In Table 4.1, the definitions of the input variables are elaborated. They are very similar to the names in the notable Matlab topology optimization algorithms by Sigmund (2001), Andreassen et al. (2011), and Liu and Tovar (2014). This makes understanding the code easier for users that are already familiar with those algorithms. In this report, expressions in the typewriter font refer to the names that are used in the topology optimization algorithm.

***Table 4.1:*** *Input variables*

| Input variable | Description |
|---|---|
| `nelx` | Number of elements in the x-direction. |
| `nely` | Number of elements in the y-direction. |
| `nelz` | Number of elements in the z-direction, as displayed in Figure 4.2. |
| `volfrac` | Percentage of elements that are allowed to be fully solid. |
| `penal` | Penalty factor, to penalize low stiffness elements and arrive at a more black-and-white solution (Figure 4.4). |
| `rmin` | Minimum radius of a sphere for which neighboring elements are taken into account in the analysis (Figure 4.5a), in order to prevent checkerboarding (Figure 4.5b), and to apply the minimum thickness constraint. |
| `maxloop` | Maximum number of iterations before the loop stops. |
| `tolx` | The change per iteration below which the loop stops. |
| `Ecmax` | Maximum Young's modulus, applied in a fully solid element. |
| `Emin` | Minimum Young's modulus, applied in an 'empty' element. The value is required to be greater than zero to prevent singular matrices. |
| `nu` | Poisson's ratio. |
| `i, j, k` | Global node coordinates for loads and boundary conditions. Figure 4.6 displays several examples. |
| `nid` | Global node ID's for loads and boundary conditions. These are numbered column wise up-to-bottom first, left-to-right second, and back-to-front third (Figure 4.3). |
| `dof` | Node degrees of freedom for loads and boundary conditions, where a load/b.c. over the x-axis is defined as *3*loadnid.T-2*, over the y-axis as *3*loadnid.T-1*, and over the z-axis as *3*loadnid.T*. |

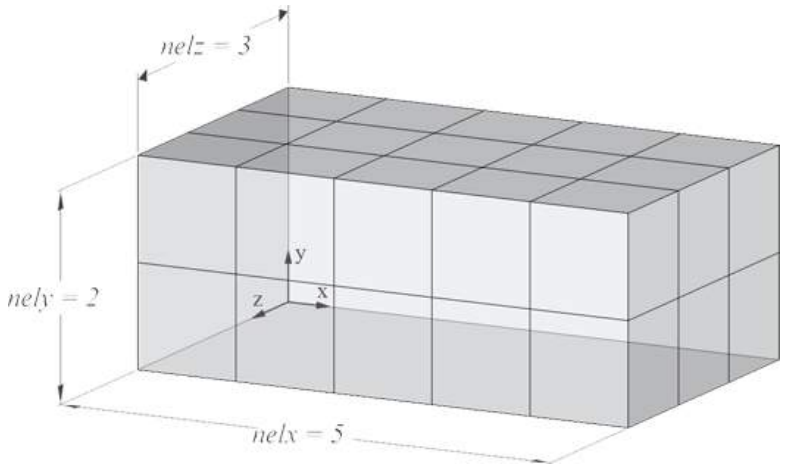



***Figure 4.2:*** *Definition of the number of elements* `nelx`*,* `nely`*, and* `nelz` *for a 5x2x3 design space*

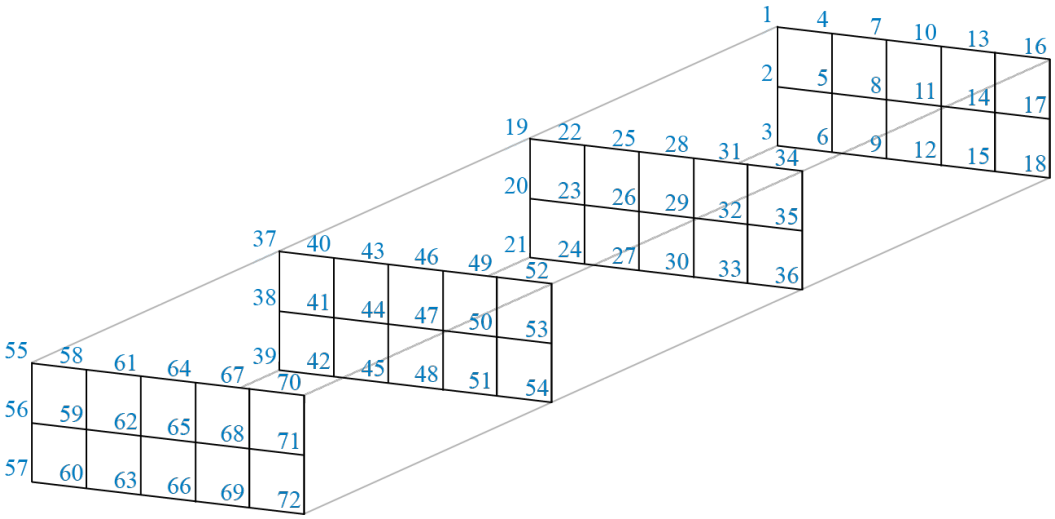



***Figure 4.3:*** *Definition of the global node ID's* `nid` *for a 5x2x3 design space*

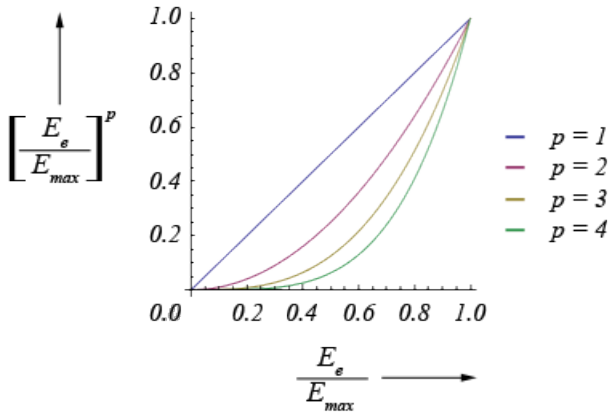
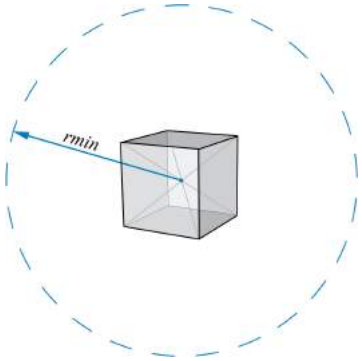
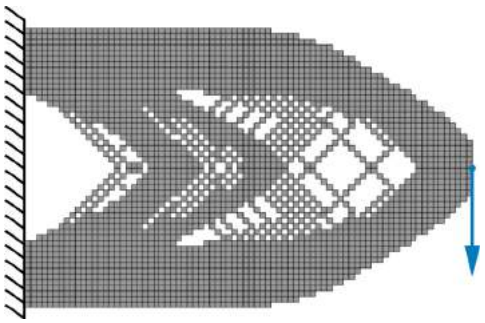



***Figure 4.4:*** *Visualization of the penalty factor* `penal`

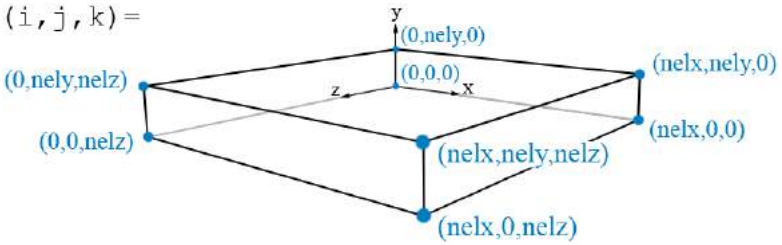



***(a)*** *Visualization of the* `rmin` *factor*

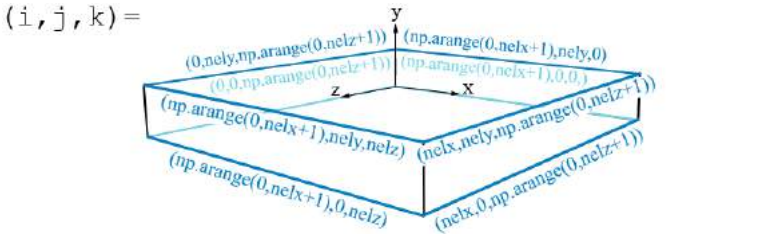


***(b)*** *Example of a checkerboard pattern, prevented by the* `rmin` *factor (Huang and Xie, 2010)*

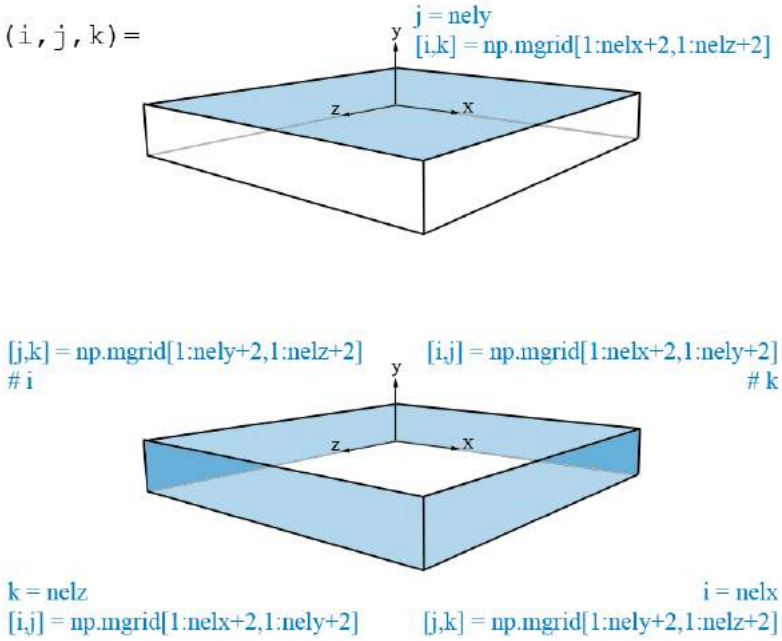
***Figure 4.5:*** *The* `rmin` *factor (Huang and Xie, 2010)*




*(a) For single elements*




*(b) For elements in a line*

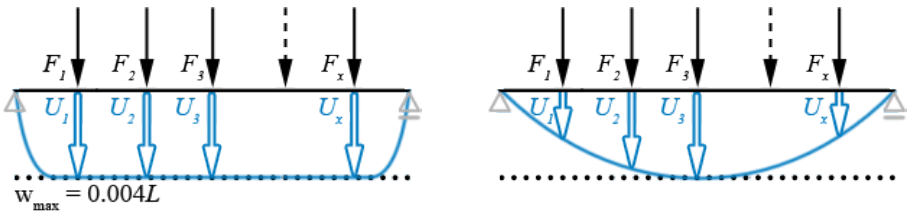



*(d) For elements on a surface*

***Figure 4.6:*** *Definition of the node coordinates used in* `il`, `jl`, *and* `kl`, *as well as in* `ibc`, `jbc`, *and* `kbc`

## 4.3 OPTIMIZATION OBJECTIVE

The Eurocode prescribes a maximum deflection of $0.004L$ in the Serviceability Limit State (SLS), where $L$ represents the span of the floor. As a result of this requirement, it would make sense to limit the acceptable deflection of the model. This would, however, result in structures that are considered 'optimal' when the entire deflection is at this limit. These undesirable deflections, displayed in Figure 4.7a, can be prevented by considering the total stiffness of the structure. One common method of defining the total stiffness is by expressing it in a total strain energy, which is the external work done by the applied load. For Figure 4.7, the strain energy is defined as $\sum \frac{1}{2} F_i U_i$ and is thus expressed in force · length. In the example of Figure 4.7, it is clear that Figure 4.7b has a lower strain energy than Figure 4.7a and is thus the 'better' structure. This corresponds with the structural engineer's preference, especially when taking potential structures such as walls on top of the floor into account that do not benefit from the extreme curvatures that are visible in Figure 4.7a near the supports.




*(a) Deflection objective* *(b) Compliance objective*

***Figure 4.7:*** *Deflections of hypothetical optimal floor slabs under different objectives*

Compliance is similar to the total strain energy of the structure, which, for SIMP, is defined as $\sum F_i U_i$ by Huang and Xie (2010). Compliance minimization is a common optimization objective in topology optimization literature. The algorithm in this thesis can be validated accurately with existing optimal solutions since they are both optimized for compliance. A minimum compliance problem with the SIMP method is defined in Equation (4.1) by Huang and Xie (2010). The objective function is subject to a maximum volume fraction $V^*$, that is composed of the volumes of individual elements $V_i$ multiplied with their relative densities $x_i$. The relative densities are anywhere between a minimum density $x_{min}$ and 1.

$$\begin{aligned} &\text{Minimize } c = f^T u \\ &\text{Subject to } V^* - \sum_{i=1}^{N} V_i x_i = 0 \\ &0 < x_{min} < x_i < 1 \end{aligned} \tag{4.1}$$

In which:

| | |
|---|---|
| $c$ | *compliance* |
| $f$ | *force vector* |
| $u$ | *displacement vector* |
| $V^*$ | *volume fraction* |
| $V_i$ | *volume element* |
| $x_i$ | *relative density element* |
| $x_{min}$ | *minimum density element* |

## 4.4 MAIN CODE

The main code can be considered as the actual topology optimization algorithm that uses definitions from other parts of the code. Algorithm 1 explains the steps and loops made by the main in the pseudo code language. The size of the arrays is displayed in gray at the end of each line. The pseudo code is slightly more in depth than the flowchart in Figure 4.1.

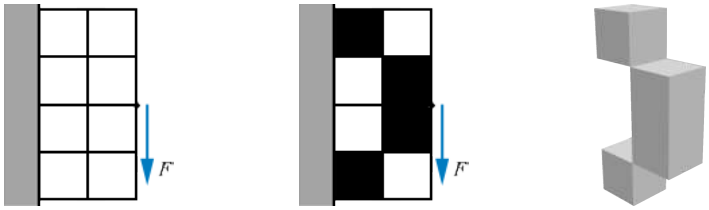
**Algorithm 1** Main code

**procedure** MAIN($nelx, nely, nelz, volfrac, penal, rmin$) ▷ *array size*
PRINT input variables
COMPUTE nodal location of load `F` ▷ *ndof × 1*
COMPUTE total degrees of freedom `ndof` ▷ *integer*
COMPUTE free degrees of freedom `freedofs` ▷ *1 × (ndof-bcdof)*
CALL elemental stiffness matrix `KE` ▷ *24 × 24*
ASSEMBLE global connectivity matrix `edofMat` ▷ *nele × 24*
COMPUTE row numbering global stiffness matrix `iK` ▷ *1 × nele·24·24*
COMPUTE column numbering global stiffness matrix `jK` ▷ *1 × nele·24·24*
CALL density filter preparation
SET initial densities per element `x` ▷ *1 × nele*
**while** change > allowed change **and** iteration < allowed iterations **do**
  COMBINE all elemental stiffness matrices in `sK` ▷ *1 × nele·24·24*
  COMPUTE global stiffness matrix `K` from `iK`, `jK`, and `sK` ▷ *ndof × ndof*
  REMOVE constrained DOF's from `K` ▷ *freedofs × freedofs*
  SOLVE nodal displacement vector for the free DOF's ▷ *ndof × 1*
  COMPUTE objective `c` ▷ *float*
  COMPUTE sensitivity objective function `dc` ▷ *1 × nele*
  COMPUTE sensitivity volume fraction `dv` ▷ *1 × nele*
  CALL optimality criterion
  COMPUTE new densities `xPhys` with density filter update ▷ *1 × nele*
  COMPUTE change of this iteration `change` ▷ *float*
  PRINT output of this iteration
PLOT result

The main code will now be clarified more on a line-by-line basis. A simple problem as proposed in Figure 4.8a will be addressed. It's two-dimensional with a very coarse mesh in order to simplify the topology optimization process. The optimal solution is stated by Bendsøe and Sigmund (2003) in Figure 4.8. The simplified problem with a very coarse mesh is displayed in Figure 4.9. The main code starts with the preparation of the Finite Element Analysis (FEA). In each iteration, there is a FEA to see how the updated model behaves, and, consequently, to see which elements need to have an increase or reduction in stiffness with the next iteration. In the Python code, the FEA consists of three parts: the definition of the elemental stiffness matrix, the preparation of the FEA in the main code, and the FEA itself in the loop in the main code.

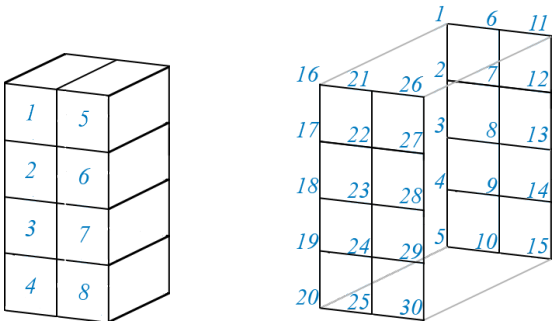

*(a) Problem* *(b) Solution (Bendsøe and Sigmund, 2003)* *(c) Found solution*

***Figure 4.8:*** *Visualization of the addressed problem with a dense mesh*




*(a) Problem* *(b) Solution* *(c) Found solution*

***Figure 4.9:*** *Visualization of the addressed problem with a coarse mesh*




*(a) Elements* *(b) Global nodes*

***Figure 4.10:*** *Numbering in the addressed problem*

Prepare FEA

The preparation includes several definitions that are used later in the algorithm. Starting with the number of elements `nele` and the accompanying degrees of freedom `ndof`, which are three per node. The applied force vector `F` has the size of `ndof`. Every three components represent the x-, y-, and z-degree of freedom of one node. In this case, a force of $-1$ in the z-direction is applied to node 13, which translates to component 38 in the force vector. The displacement vector works the same way and has the same size. The vector containing all the free degrees of freedom consists of the difference between all the degrees of freedom and the fixed ones. The vector is called `freedofs`. From Figure 4.10b, we know which nodes are fixed: nodes 1 to 5 and nodes 16 to 20, which translates to degrees of freedom 1 to 15 and 46 to 60 in `bcdof` since they are fixed in all three directions.

$$\texttt{nele} = \texttt{nelx} \cdot \texttt{nely} \cdot \texttt{nelz} = 8 \quad (1\times1)$$

$$\texttt{ndof} = 3(\texttt{nelx}+1) \cdot (\texttt{nely}+1) \cdot (\texttt{nelz}+1) = 90 \quad (1\times1)$$

$$\texttt{F} = [\overbrace{\underbrace{0}_{\text{x}},\underbrace{0}_{\text{y}},\underbrace{0}_{\text{z}}}^{\text{node 1}}\ldots\overbrace{\underbrace{0}_{\text{x}},\underbrace{-1}_{\text{y: pos. 38}},\underbrace{0}_{\text{z}}}^{\text{node 13}}\ldots 0,0,0]^T \quad (ndof\times1)$$

$$\texttt{U} = [0,0,0\ldots0,0,0]^T \quad (ndof\times1)$$

$$\texttt{bcdof} = [\overbrace{1,2,3\ldots13,14,15}^{\text{nodes 1 to 5}},\overbrace{46,47,48\ldots58,59,60}^{\text{nodes 16 to 20}}]$$

$$\texttt{freedofs} = [16,17,18\ldots43,44,45,61,62,63\ldots88,89,90]$$

The `24 x 24` elemental stiffness matrix is found in the research by Liu and Tovar (2014). It is determined in the definition `ElStiffnMat` in the algorithm and called in the main code during the preparation of the Finite Element Analysis. The only variable called in the definition is the Poisson's ratio and it returns the elemental stiffness matrix.

In the next line, `edofMat` is the connectivity matrix with size `nele x 24` that stores every element's local node IDs. Note: the global node ID differs from the local node ID! The next step is the density filter, which is prepared as explained in Section 4.5.1.

$$\texttt{edofMat} = \begin{bmatrix} 4 & 5 & 6\ldots46 & 47 & 48 \\ 7 & 8 & 9\ldots49 & 50 & 51 \\ 10 & 11 & 12\ldots52 & 53 & 54 \\ 13 & 14 & 15\ldots55 & 56 & 57 \\ 19 & 20 & 21\ldots61 & 62 & 63 \\ 22 & 23 & 24\ldots64 & 65 & 66 \\ 25 & 26 & 27\ldots67 & 68 & 69 \\ 28 & 29 & 30\ldots70 & 71 & 72 \end{bmatrix} \begin{matrix} \text{Element No. 1} \\ \text{Element No. 2} \\ \text{Element No. 3} \\ \text{Element No. 4} \\ \text{Element No. 5} \\ \text{Element No. 6} \\ \text{Element No. 7} \\ \text{Element No. 8} \end{matrix}$$

Initialize iteration

Before starting the iteration, several variables need to be assigned a starting point. The physical element densities `xPhys` are all set to the desired volume fraction in the first iteration. The elemental compliance `ce`, the derivative of the compliance `dc`, and derivative of the elemental volumes `dv` are all set to 1 for each element. The optimization loop runs until either the maximum allowed number of iterations have been completed or the change per iteration becomes lower than the set limit.

$$\texttt{volfrac} = 0.5 \quad (1\times1)$$
$$\texttt{xPhys} = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5] \quad (1\times nele)$$
$$\texttt{ce} = [1, 1, 1, 1, 1, 1, 1, 1] \quad (1\times nele)$$
$$\texttt{dc} = [1, 1, 1, 1, 1, 1, 1, 1] \quad (1\times nele)$$
$$\texttt{dv} = [1, 1, 1, 1, 1, 1, 1, 1] \quad (1\times nele)$$

FEA

The function of the Finite Element Analysis is to determine the displacements `U` of the structure under the applied load. After this, the internal force distribution can be determined. It achieves this goal in several steps. The global stiffness matrix `K` is put together as a sparse matrix. The sparse matrix employs three arguments: row vector `iK`, column vector `jK`, and entry vector `sK`. The row and column vectors indicate in which position the entries should be placed in the global stiffness matrix, and they are a rewritten, flattened connectivity matrix `edofMat`. The entry vector `sK` contains all the elemental stiffness matrices `KE`, multiplied with their stiffness. Since this involves the structural densities of the elements, rather than the physical densities `xPhys`, Huang and Xie (2010) tell us that the penalty factor is now also included for calculating the stiffness, resulting in Equation (4.2), where $x_i$ is the density of an element, $p$ is the penalty, and $K_i^0$ is the elemental stiffness matrix of the corresponding element.

$$K = \sum_{i=1}^{nele} x_i^p K_i^0 \tag{4.2}$$

$$\texttt{sK} = \left[x_0^p K_0^0, x_1^p K_1^0, \ldots, x_i^p K_i^0\right] = \left(\texttt{Emin} + \texttt{xPhys}^{\texttt{penal}}(\texttt{Emax} - \texttt{Emin})\right)\texttt{KE} \quad (1\times 24^2 nele)$$
$$\texttt{K} = K = \sum_{i=1}^{nele} x_i^p K_i^0 = \sum \texttt{sK} \quad (ndof\times ndof)$$

Finally, the displacements `U` are solved according to Equation (4.3) for the free degrees of freedom only, ensuring the boundary conditions are complied with. The nodal forces $f$ follow linearly from the displacement vector `U`. The sparse stiffness matrix is inverted in a direct manner, which is considered very quick for small design spaces and quick enough for larger ones.

$$Ku = f \tag{4.3}$$

$$\begin{aligned} &u = K^{-1}F, \quad \text{therefore: } \texttt{U} = \texttt{K}^{-1}\texttt{F} \quad {\scriptstyle (ndof \times 1)} \\ &f_i = u_i K_i^0, \quad \text{therefore: } \texttt{f} = \texttt{KE} \times \texttt{U} \quad {\scriptstyle (ndof \times 1)} \end{aligned} \tag{4.4}$$

Objective and sensitivities for a compliance problem

As mentioned in Section 4.3, the objective of the topology optimization is to maximize the overall stiffness of the structure. The overall stiffness is measured by the compliance and defined as `c`. The elemental compliance `ce` is computed by multiplying the nodal forces $f$ by the nodal displacements $u$, and is therefore measured in force · length. The sum of the elemental compliances results in the final compliance `c` that the quality of the iteration is measured with. Similar to the stiffness matrix, the penalty factor is also included in the final compliance value.

The sensitivity of the objective function is called `dc` and signifies which elements are the most sensitive to changing their stiffness. Elements which have the lowest sensitivity, and thus increase the objective the least, are the most attractive for removal. Similarly, elements which have the highest sensitivity, and thus increase the objective the most, are the most valuable for stiffening. The sensitivity of the objective function is derived in Equation (4.5).

$$\begin{aligned} &\text{If: } c = f^T u \\ &\text{Then: } \frac{\delta c}{\delta x_i} = \frac{\delta f^T}{\delta x_i} u + f^T \frac{\delta u}{\delta x_i} \end{aligned} \tag{4.5}$$

At this point, Huang and Xie (2010) propose introducing a Lagrange multiplier $\lambda$, which changes the function in Equation (4.5) to Equation (4.6). From Equation (4.3), it is known this does not change the outcome of the compliance function since the second term is equal to zero for every $\lambda$. The Lagrange multiplier can now be chosen in order to remove the unknown factor $\frac{\delta u}{\delta x_i}$ from Equation (4.8). From Equation (4.3), it follows that this is the case for $\lambda = u$, resulting in Equation (4.9).

$$\text{If: } c = f^T u + \lambda^T (f - Ku) \tag{4.6}$$

$$\frac{\delta c}{\delta x_i} = \frac{\delta f^T}{\delta x_i} u + f^T \frac{\delta u}{\delta x_i} + \frac{\delta \lambda^T}{\delta x_i}(f - Ku) + \lambda^T \left( \frac{\delta f}{\delta x_i} - \frac{\delta K}{\delta x_i} u - K \frac{\delta u}{\delta x_i} \right) \tag{4.7}$$

If $f - Ku = 0$ and $\frac{\delta f}{\delta x_i} = 0$, then:

$$\begin{aligned} \frac{\delta c}{\delta x_i} &= f^T \frac{\delta u}{\delta x_i} + \lambda^T \left( -\frac{\delta K}{\delta x_i} u - K \frac{\delta u}{\delta x_i} \right) \\ \frac{\delta c}{\delta x_i} &= \left( f^T - \lambda^T K \right) \frac{\delta u}{\delta x_i} - \lambda^T \frac{\delta K}{\delta x_i} u \end{aligned} \tag{4.8}$$

$$\frac{\delta c}{\delta x_i} = -u^T \frac{\delta K}{\delta x_i} u \tag{4.9}$$

By combining Equation (4.2) and Equation (4.9), the final sensitivity function displayed in Equation (4.10) is derived.

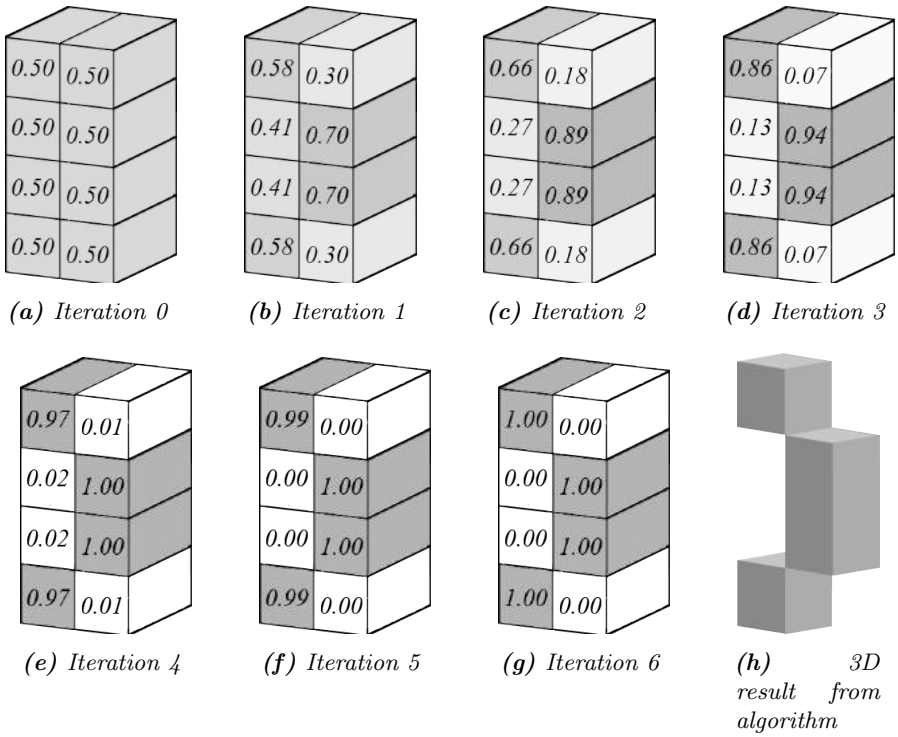
$$\frac{\delta c}{\delta x_i} = -p x_i^{p-1} u_i^T K_i^0 u_i \tag{4.10}$$

This is implemented in the algorithm as displayed below. During the filtering, the sensitivities are updated according to the results from Section 4.5.1. In this particular example `rmin` $= 1.0$, therefore the neighboring elements are not taken into account and the density filter does not change the sensitivity values for both the volume and the objective.

$$\begin{aligned} &\texttt{ce} = f_i^T u_i = u_i^T K_i^0 u_i = (\texttt{U}^T \times \texttt{KE}) \cdot \texttt{U} \quad {\scriptstyle (1 \times nele)} \\ &\texttt{c} = \sum_{i=1}^{nele} x_i^p f^T U = \sum \left( \texttt{Emin} + \texttt{xPhys}^{\texttt{penal}} (\texttt{Emax} - \texttt{Emin}) \right) \texttt{ce} \quad {\scriptstyle (1 \times 1)} \\ &\texttt{dc} = -p x_i^{p-1} u_i^T K_i^0 u_i = -\texttt{penal} \cdot \texttt{xPhys}^{\texttt{penal}-1} \texttt{ce} \quad {\scriptstyle (1 \times nele)} \\ &\texttt{dv} = [1, 1, 1, 1, 1, 1, 1, 1] \quad {\scriptstyle (1 \times nele)} \end{aligned}$$

### Optimality criteria update

Now that first the displacements and internal forces are determined, and secondly the sensitivities, the updated structure can be determined during the Optimality Criteria (OC) update. Section 4.5.2 contains the definition that is called in the loop. The method was proposed by Bendsøe (1995) and utilizes the bisection method. After the OC update, the updated elemental densities `xPhys` per iteration are known and this results in the following values. When `tolx=0.01`, the algorithm reaches its optimum after six iterations. The iterations are displayed in Figure 4.11.




***(a)*** *Iteration 0* ***(b)*** *Iteration 1* ***(c)*** *Iteration 2* ***(d)*** *Iteration 3*

***(e)*** *Iteration 4* ***(f)*** *Iteration 5* ***(g)*** *Iteration 6* ***(h)*** *3D result from algorithm*

***Figure 4.11:*** *Physical densities* `xPhys` *per iteration*

Compute change and plot

The factor `change` is based on the change in elemental densities per iterations, and not on the change in the objective function per iteration. The main loop finishes when `change` becomes smaller than `tolx`, or when the maximum number of iterations `maxloop` has been reached. In order to arrive at an optimum, however, the optimization should be terminated by `tolx` and not by `maxloop`. The visualization of the final result is achieved as described in Section 4.5.3.

## 4.5 DEFINITIONS

### 4.5.1 Density filter

Using a density filter kills two birds with one stone. First, a minimum member radius `rmin` is useful in concrete modeling. Secondly, the density filter prevents checkerboarding, as mentioned in Section 4.2. In this code, the density filter consists of two parts: a preparation of the filter as a definition in the main code, and later, the actual filtering during the optimization loop. Algorithm 2 elaborates on the definition of the preparation, called `df` in the algorithm, after which the actual filtering during each iteration occurs in the loop. The density filter is often used in topology optimization and was proposed by Bruns and Tortorelli (2001) and Bourdin (2001). The final updated physical densities of the iteration, are determined according to Equation (4.11), which is based on research by Bruns and Tortorelli (2001), in which $R$ is the minimum radius and $i, j, k$ denote the coordinates of the

elements.

$$
\begin{aligned}
x_i^{filtered} &= \sum_i \frac{\omega_i}{\omega} x_i \\
\omega_j &= \max\left(0, R - \sqrt{(i_i - i_j)^2 + (j_i - j_j)^2 + (k_i - k_j)^2}\right) \\
\omega &= \sum_i \omega_i
\end{aligned}
\tag{4.11}
$$

In the algorithm, it is applied as displayed below. The matrix `H` contains all the relative distances between the elements that are closer together than `rmin`. However, since the minimum radius is a lot smaller than the design space, the matrix mostly consists of zeros and can, therefore, be made sparse.

$$
\begin{aligned}
\texttt{xPhys} = x_i^{filtered} &= \frac{\texttt{H}}{\texttt{Hs}}\texttt{x} \quad {\scriptstyle (1 \times nele)} \\
\texttt{H} = \omega_j &= \max\left(0, \texttt{rmin} - \sqrt{(\texttt{i} - \texttt{i2})^2 + (\texttt{j} - \texttt{j2})^2 + (\texttt{k} - \texttt{k2})^2}\right) \quad {\scriptstyle (nele \times nele)} \\
\texttt{Hs} = \omega &= \sum \texttt{H} \quad {\scriptstyle (nele \times 1)}
\end{aligned}
$$

**Algorithm 2** Preparation density filter

```
procedure DEF DF(nelx, nely, nelz, rmin)                    ▷ array size
    SET size of filter nfilter                              ▷ integer
    for each element in z-direction do
        for each element in x-direction do
            for each element in y-direction do
                COMPUTE row numbering row                   ▷ integer
                for each element within range of rmin in z-direction do
                    for each element within range of rmin in x-direction do
                        for each element within range of rmin in y-direction do
                            COMPUTE column numbering col    ▷ integer
                            DEFINE row vector iH
                            DEFINE column vector jH
                            COMPUTE filter vector sH        ▷ 1 × freedofs
    RETURN sH, iH, and jH for assembly of H and Hs
```

### 4.5.2 Optimality criterion

Finding the solution to the compliance problem requires a method to find the minimum. In this algorithm, the Optimality Criterion (OC) method is applied to update the physical densities `xPhys` each iteration. The method was proposed by Bendsøe (1995) and utilizes the bisection method. He states the update as displayed in Equation (4.12). In this algorithm, it is defined as `oc` and called near the end of the

optimization loop. Algorithm 3 includes the pseudo code for the OC definition.

$$B_i = -\frac{\delta c}{\delta x_i}\left(\lambda\frac{\delta v}{\delta x_i}\right)^{-1} = \lambda^{-1} p x_i^{p-1} u_i^T K_i^0 u_i$$
$$x_i^{k+1} = \begin{cases} \max(x_{min}, x_i^k - m) & \text{if } x_i^k B_i^\eta \leq \max(x_{min}, x_i^k - m) \\ \min(1, x_i^k + m) & \text{if } \min(1, x_i^k + m) \leq x_i^k B_i^\eta \\ x_i^k B_i^\eta & \text{otherwise} \end{cases} \tag{4.12}$$

In Equation (4.12), $x_i^k$ is the elemental density at iteration $k$, $m$ is the positive move limit set at 0.2, and $\eta$ is the numerical damping coefficient set at 0.5 (Bendsøe, 1995). Now, the only unknown is the Lagrange multiplier $\lambda$, which can be found with the bisection method. The multiplier can be seen as a threshold at which material is either removed or added to an element. The Lagrange multiplier is guessed to be between a lower limit `l1` and an upper limit `l2`. Each iteration these bounds are cut in half until the solution is found within a range of 0.001.

**Algorithm 3** Optimality Criterion

```
procedure DEF OC(nelx, nely, nelz, x, volfrac, dc, dv, g)
    DEFINE boundaries l1 and l2
    DEFINE positive move limit move      ▷ 0.2 recommended (Bendsøe, 1995)
    while change > tolerance of 0.001 do
        UPDATE physical densities xnew
        COMPUTE new boundaries
    RETURN new physical densities xnew
```

### 4.5.3 Visualization optimum

The output of the algorithm is presented per iteration as a line of text, where `It.` gives the iteration number, `Obj.` the objective/compliance, `Vol.` the volume fraction, and `Ch.` the change per iteration.

After the last iteration has finished, a three-dimensional model is constructed, showcasing the final optimal result. This is accomplished with the help of the MayaVi package in the definition called `displayfigure` (Algorithm 4). In order to achieve a true black-and-white visualization, all the element densities larger than 0.5 are assumed to be 'filled', and the elements with a smaller density are left out of the figure. This interpretation of the actual 'gray' result is always a point of discussion inherent to the SIMP method. In practice, if the penalty factor is high enough and the tolerance for ending the optimization is small enough, the physical densities of the elements generally range between 0.8-1.0 for 'full' and 0.0-0.2 for 'empty' elements. This difference is considered acceptable for making a practical design.

**Algorithm 4** Visualization optimum

**procedure** DEF DISPLAYFIGURE($xPhys, nelx, nely, nelz$)
DEFINE empty array `xPlot`, which will contain the discretized `xPhys`
**for** each element in `xPhys` **do**
  **if** element density is more than 0.5 **then**
    UPDATE element density to be 1.0 in `xPlot`
  **if** element density is less than 0.5 **then**
    UPDATE element density to be 0.0 in `xPlot`
DEFINE element coordinates `px`, `py`, `pz`
**for** each element in `xPlot` **do**
  **if** density is more than 0.5 **then**
    DRAW filled element at its corresponding coordinate
DISPLAY final figure

## 4.6 VALIDATION

The topology optimization algorithm needs to be validated in order to confirm its accuracy, and to see if it meets its intended purpose. Validation has been performed by comparing the results to well-studied examples. These case studies include two cantilever beams, a wheel, and an MBB-beam which was first proposed by the Messerschmitt-Bölkow-Blohm aviation company. Repeatability has been checked by performing each problem multiple times. The examples also demonstrate the capabilities of the algorithm and can help the end user compose their own problems.

Only the standard algorithm has been evaluated in this section. That is to say, the algorithm without any additional constraints or self-weight. The variables are displayed in the `(nelx,nely,nelz,volfrac,penal,rmin)` fashion. Figure 4.12 to Figure 4.15 show that the results from the algorithm are almost identical to the solutions that are described in literature. The small differences can be explained by the manner in which the solution is found. Differences in the solution-finding algorithm cause small differences in the found optimum. Additionally, the three-dimensional visualization discretizes the solution, while the solution actually consists of floats. Lastly, the density of the mesh also plays a role. A denser mesh will result in figures that are closer to the optima defined by literature.

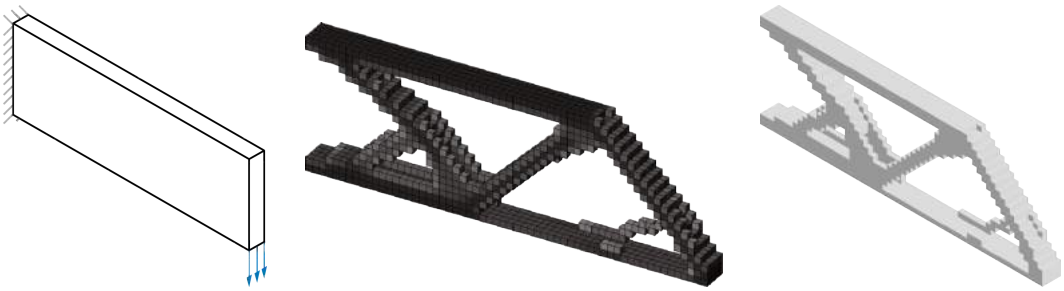
### 4.6.1 Cantilever beam A

The cantilever beam is displayed in Figure 4.12 and optimized with the following input: `(60,20,4,0.3,3,1.5)`. The load is defined as:

```
il = np.array([nelx])
jl = np.array([0])
kl = np.array([np.arange(0,nelz+1)])
loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl)
loaddof = 3*loadnid.T - 1
```

The boundary conditions are defined as:

```
[jbc,kbc] = np.mgrid[1:nely+2,1:nelz+2]
bcnid = (kbc.T-1)*(nely+1)*(nelx+1)+jbc.T
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid-2)).flatten()
```



*(a) Design space*

*(b) Result from literature (Liu and Tovar, 2014)*

*(c) Result from the algorithm*

***Figure 4.12:*** *Cantilever beam A comparison*

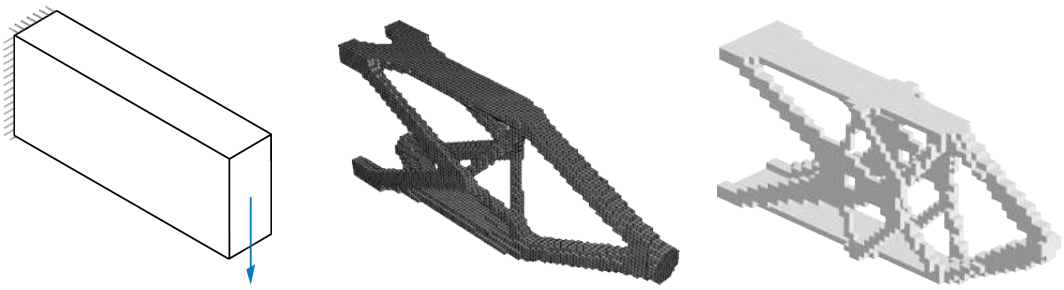
### 4.6.2 Cantilever beam B

An alternative cantilever beam is displayed in Figure 4.13. Instead of a line load at the bottom end of the cantilever, there is a point load in the center of the profile. It is noteworthy that the topology from TopOpPy differs from the BESO output. This is an example of the SIMP method not always finding the global optimum. A denser mesh might be able to achieve a more accurate result. The problem is optimized with the following input: `(60,24,12,0.1,3,1.5)`. The load is defined as:

```
il = np.array([nelx])
jl = np.array([nely/2])
kl = np.array([nelz/2])
loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl)
loaddof = 3*loadnid.T - 1
```

The boundary conditions are, again, defined as:

```
[jbc,kbc] = np.mgrid[1:nely+2,1:nelz+2]
bcnid = (kbc.T-1)*(nely+1)*(nelx+1)+jbc.T
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid-2)).flatten()
```



*(a) Design space*

*(b) Result from literature (Huang and Xie, 2010)*

*(c) Result from the algorithm*

***Figure 4.13:*** *Cantilever beam B comparison*

### 4.6.3 Wheel

The wheel is displayed in Figure 4.14 and optimized with the following input: `(20,10,20,0.2,3,1.5)`. The load is defined as:

```
il = np.array([nelx/2])
jl = np.array([0])
kl = np.array([nelz/2])
loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl)
loaddof = 3*loadnid.T - 1
```

The boundary conditions are defined as:

```
ibc = np.array([0,0,nelx,nelx])
jbc = np.array([0,0,0,0])
kbc = np.array([0,nelz,0,nelz])
bcnid = kbc*(nelx+1)*(nely+1)+ibc*(nely+1)+(nely+1-jbc)
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid-2)).flatten()
```

***(a)*** *Design space*

***(b)*** *Result from literature (Liu and Tovar, 2014)*

***(c)*** *Result from the algorithm*

***Figure 4.14:*** *Wheel comparison*

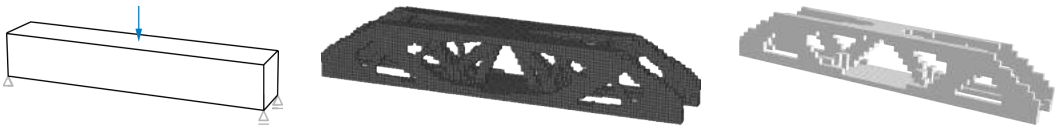
### 4.6.4 MBB beam

The MBB beam is displayed in Figure 4.15 and optimized with the following input: `(60,10,10,0.2,3,1.5)`. The load is defined as:

```
il = np.array([nelx/2])
jl = np.array([nely])
kl = np.array([nelz/2])
loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl)
loaddof = 3*loadnid.T - 1
```

The boundary conditions are defined as:

```
ibc = np.array([0,0,nelx,nelx])
jbc = np.array([0,0,0,0])
kbc = np.array([0,nelz,0,nelz])
bcnid = kbc*(nelx+1)*(nely+1)+ibc*(nely+1)+(nely+1-jbc)
bcdof=np.hstack((3*bcnid[0:2],3*bcnid[0:4]-1,3*bcnid[0:2]-2)).flatten()
```



*(a) Design space* *(b) Result from literature (Liu and Tovar, 2014)* *(c) Result from the algorithm*

***Figure 4.15:*** *MBB beam comparison*

## 4.7 SPEED

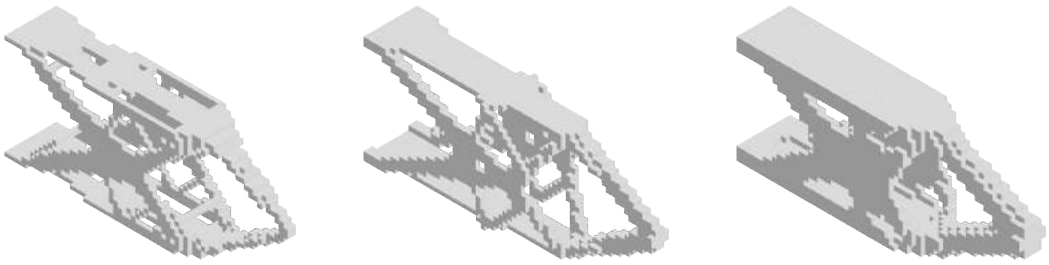
In this section, the standard topology optimization algorithm is compared to the similar Matlab algorithm by Liu and Tovar (2014). The algorithms by Sigmund (2001) and Andreassen et al. (2011) have not been compared since they do not function for three-dimensional problems. Table 4.2 displays the calculation times for Python and Matlab when solving the aforementioned topology optimization problems, for the same amount of iterations. They are measured using a laptop with an Intel Quad-core i7-4720HQ 2.6 GHz CPU and 16.0 GB RAM. The total number of elements and the number of neighboring elements influence the calculation times the most. When expanding the design space, the number of elements increases and, subsequently, so does the size of the global stiffness matrix. In each iteration, solving the displacements by directly inverting the sparse global stiffness matrix takes the longest time to accomplish. Additionally, bigger design spaces generally need more iterations to reach an optimum. The number of neighboring elements is also a factor that influences the calculation times, illustrated by the longer times from the cube-like 'wheel' problem versus the relatively flat 'cantilever beam' problems. From Table 4.2, it can be concluded that the Python algorithm is slower than the one in Matlab and that the calculation durations increase exponentially for larger design spaces.

***Table 4.2:*** *Calculation times*

| Problem | Design space | Number of iterations | Time in Python (s) | Time in Matlab (s) |
|---|---|---|---|---|
| Cantilever beam A | $15 \times 5 \times 1$ | 49 | 2.5 | 0.5 |
| | $30 \times 10 \times 2$ | 71 | 7.9 | 4.3 |
| | $60 \times 20 \times 4$ | 156 | 522.5 | 121.5 |
| Cantilever beam B | $15 \times 6 \times 3$ | 55 | 3.3 | 1.6 |
| | $30 \times 12 \times 6$ | 90 | 64.4 | 25.5 |
| Wheel | $20 \times 10 \times 20$ | 53 | 314.7 | 44.9 |
| MBB beam | $60 \times 10 \times 10$ | 157 | 563.7 | 154.4 |

## 4.8 SENSITIVITY

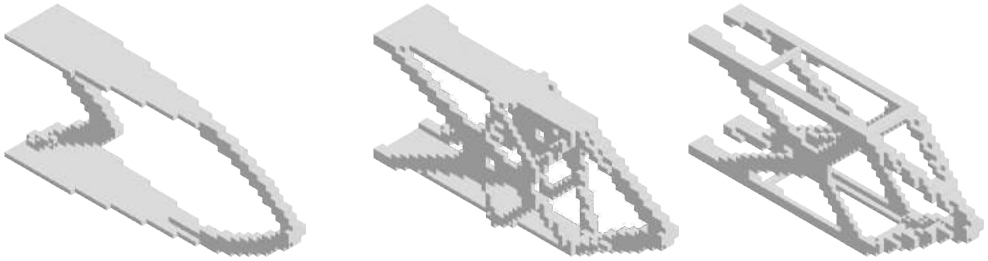
In this section, the algorithm's sensitivity has been researched. The goal is to see how different ratio's in input variables influence the final design. The standard problem of the second cantilever beam of Section 4.6.2 has been solved multiple times, with various input parameters, but without self-weight. The design space size has been left the same as `60 x 24 x 12`, but the minimum radius, the penalty factor, and volume constraint have been played with. The result is displayed in the following figures. Starting with the minimum radius in Figure 4.16, the result is directly visible in the thickness of the truss members. A smaller radius results in a greater quantity of thinner members. In this case, a value of 1.5 is enough to prevent checkerboarding, while still allowing very slender beams. The third image shows a minimum member thickness of 3.0, which requires more volume to find a black-and-white solution. In conclusion, the minimum radius seems to be rather sensitive as small changes result in different topologies, while still conforming to the volume constraint.



*(a)* `rmin=1.0` *(b)* `rmin=1.5` *(c)* `rmin=3.0`

***Figure 4.16:*** *Sensitivity minimum radius*

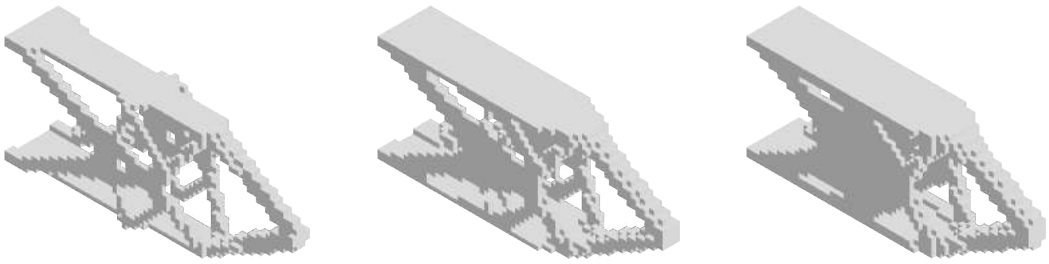
The sensitivity of the penalty factor has been researched, next. The higher this factor is, the more intermediate densities are penalized. This results in more black-and-white solutions, but can also result in not finding the global optimum. The images in Figure 4.17 depict the effect of several factors on the final design. A low penalty factor (Figure 4.17a) gives a topology that is nowhere near a viable structure. While it is hidden in the visualization, there is simply too much intermediate material in between the outer flanges. In order to arrive at a compromise between black-and-white solutions and convergence, a value of 3.0 has been adopted for all the designs in the rest of the report, as is customary with many other SIMP packages by, e.g., Sigmund (2001) and Andreassen et al. (2011). A correct method of application is running the first entire optimization for `penal=1.0`, then the second one for `penal=2.0`, and one last time for `penal=3.0`, each time continuing from the optimal solution of the previous optimization. This way the found optimum reaches closer to the global optimum.



*(a)* `penal=1.5` *(b)* `penal=3.0` *(c)* `penal=5.0`

***Figure 4.17:*** *Sensitivity penalty factor*

Similar to the minimum radius, adjusting the volume fraction also results in different topologies, as displayed in Figure 4.18. As expected, a higher allowed volume fraction gives a more filled design space, and a more stiff structure overall. The value of the volume fraction often follows from the design problem, so in practice it can be difficult to use the fraction in order to adapt the final design.



*(a)* `volfrac=0.10` *(b)* `volfrac=0.15` *(c)* `volfrac=0.20`

***Figure 4.18:*** *Sensitivity volume constraint*

## 4.9 BASIC TOPOLOGY-OPTIMIZED FLOOR SLAB DESIGN

### 4.9.1 Design space and boundary conditions

After validation of the algorithm in Section 4.6, the algorithm can be used to design a topology optimized floor slab. The design space represents the space in which the optimal solution must be found. For the floor design, it is shaped as a rectangular cuboid, because buildings and floors usually consist of rectilinear polygons. For topology optimization in general, a design space can be divided into either pixels, beams, or points. In this thesis, it has been divided in the three-dimensional interpretation of pixels: cubic elements.

The potential for development of a topology-optimized floor is the highest when its advantages are adequately exploited. A floor that can be produced with a complex shape, can withstand high loads, and has additional value from its aesthetics, benefits the most from a location where many people come together. Meeting areas in non-residential building construction appear to be suitable for fulfilling this function. These areas usually benefit from wide open spaces in order to keep the building function flexible. To that end, a span of seven meters should be able to provide

sufficient adaptability. This span is slightly bigger than the iconic Gatti wool factory floor from Nervi, discussed in Section 3.3.1, which has a span of five meters. The width of 2.5 m follows from the width of a truck, so the floor slabs can easily be moved from the factory to the building site.

The boundary conditions follow from the context and supports of the floor: is it supported by columns or walls, is the floor continuous or statically determined, or a combination of the aforementioned? For the boundary conditions, a slab supported by a beam or wall will be assumed. This translates to a hinged line support over two opposing edges (Figure 3.5). One line support is not allowed to translate in any direction, the other is allowed a degree of freedom in the x-direction to simulate a roll. This results in a statically determinate structure and, as a consequence, the forces in the structure are not influenced by their relative stiffness. The following code has been applied to select all the nodes on the lower edges of two sides, and secure their degrees of freedom:

```
ibc = np.array([0,nelx])
jbc = np.array([0,0])
kbc = np.array([np.arange(0,nelz+1),np.arange(0,nelz+1)])
bcnid = np.hstack((kbc[0]*(nelx+1)*(nely+1)+ibc[0]*(nely+1)+(nely+1-jbc[0])
, kbc[1]*(nelx+1)*(nely+1)+ibc[1]*(nely+1)+(nely+1-jbc[1])))
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid[0:nelz+1]-2)).flatten()
```

### 4.9.2 Loads

For meeting areas in non-residential building construction, the load class is based on class C3, according to Table 6.1 of NEN-EN 1991-1-1. This corresponds to an equally distributed load over the top surface, with a value of 5.0 kN/m$^2$ (Table 4.3) in the Serviceability Limit State. The live load is implemented with the following code. It can be interpreted as a vertical point load on all the top nodes, which resembles the equally distributed load.

```
jl = np.array([nely])
[il,kl] = np.mgrid[1:nelx+2,1:nelz+2]
loadnid = (kl.T-1)*(nely+1)*(nelx+1)+il.T+jl*(il.T-1)
loaddof = 3*loadnid.T - 1
load = -5.*(7.*2.5)/(nelx*nelz)
```

***Table 4.3:*** *NEN-EN 1991-1-1, Table 6.2*

| *Class of loaded area* | $q_k$ kN/m$^2$ | $Q_k$ kN |
|---|---|---|
| Class C3 | 5.0 | 4.0 |

### 4.9.3 Result

Without any manufacturability constraints, the optimization reduces to a basic topology optimization of a design space with a volume constraint. In theory, this results in the most optimal structure according to the objective function. However, the considered material is not appropriate for concrete, there are gaps in the top of the floor,

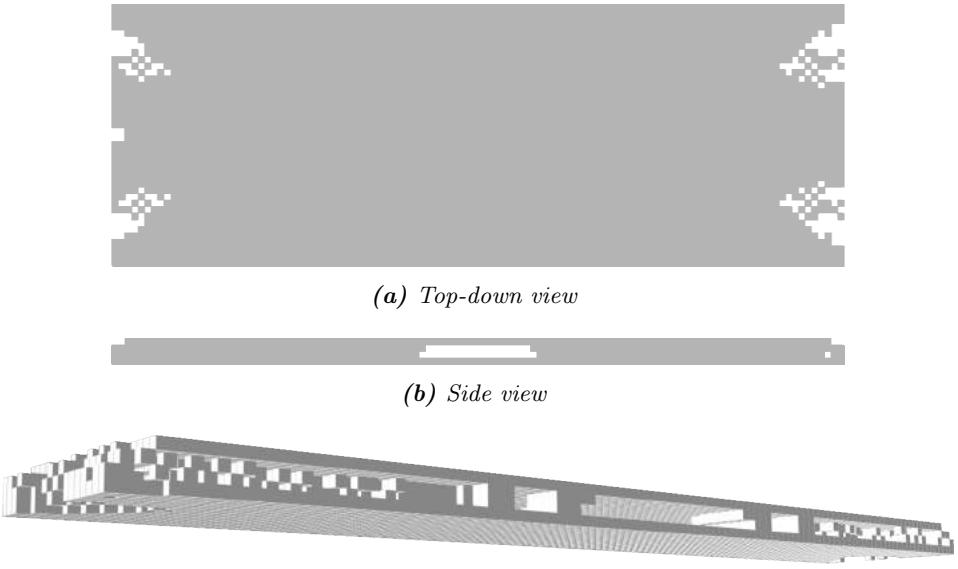
and the topology is still too complex to be manufactured. The design in Figure 4.20 has a maximum deflection of 3.7 mm and an objective of 0.0124 kN m.

***Table 4.4:*** *Input variables concrete topology optimization*

| Input variable | Value |
|---|---|
| `nelx` | 112 |
| `nely` | 4 |
| `nelz` | 40 |
| `volfrac` | 0.575 |
| `penal` | 3.0 |
| `rmin` | 0.5 |

What follows from the optimization process, is a hollow structure with the biggest possible internal arm at midspan, and very slender webs near the supports. Both the internal openings and the very thin members are not desirable. The checkerboard pattern is also unwanted. These three problems are illustrated in Figure 4.19.

The next chapter introduces a manufacturing method called vacuumatic formwork that is capable of producing complex shapes with a low-tech and sustainable procedure. While even vacuumatic formwork would have trouble producing the floor in Figure 4.20, utilizing such a versatile method can help make a topology-optimized floor more practical. After introducing the manufacturing method in Chapter 5, the optimization will be extended with all the additional manufacturability/material/-functionality features in Chapter 6.

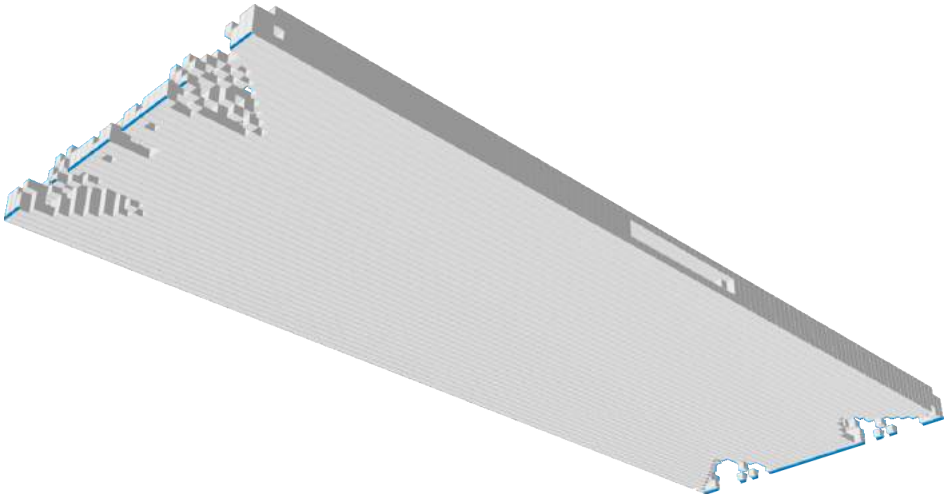


***(a)*** *Top-down view*

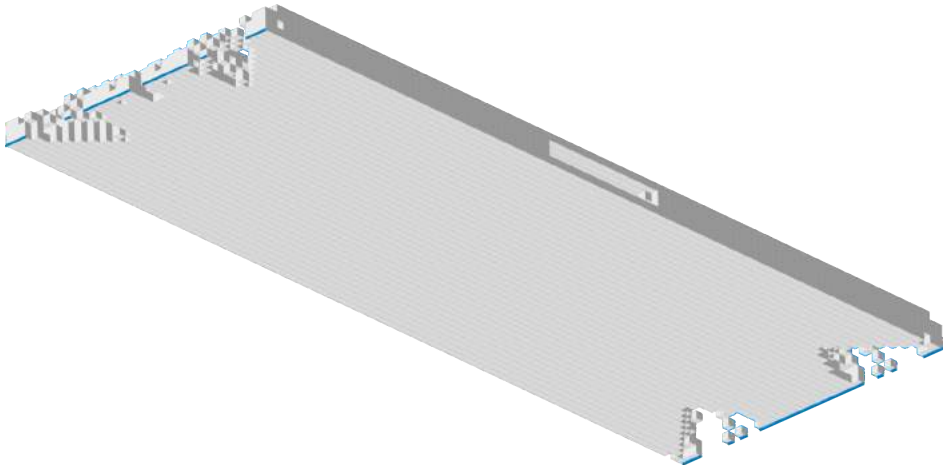
***(b)*** *Side view*

***(c)*** *Cross-section*

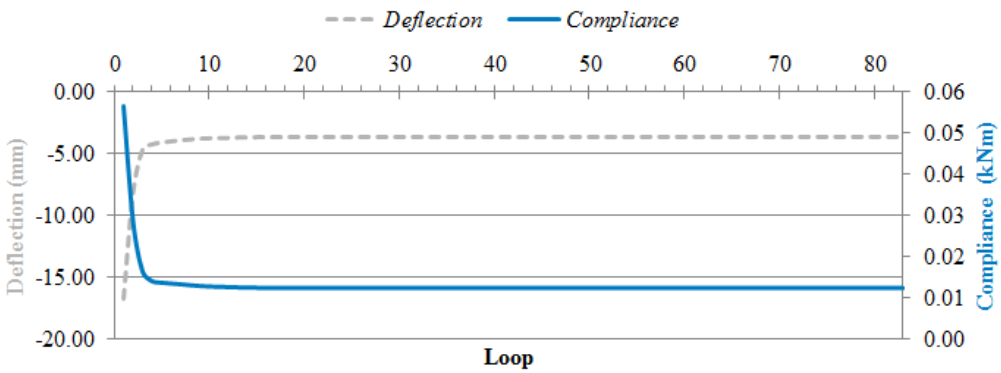
***Figure 4.19:*** *Top: the checkerboard pattern near the supports; middle: the gap at midspan where inefficient material is removed; bottom: cross-section to display the cavity inside*



*(a) Perspective projection*



*(b) Parallel projection*




*(c) Optimization diagram*

***Figure 4.20:*** *Underside of the optimal design according to traditional topology optimization. The supports are displayed in blue*

# 5

# Vacuumatic formwork

From Chapter 2, it can be concluded that the field of digital design and structural optimization is progressing quickly. Making the design is only one part of the building process, however, since manufacturability of that design is just as influential. Usually, the final design in the construction stage differs from the topology-optimized structure because of manufacturability. In theory, a structural topology-optimized floor is very efficient, but in practice, it can be too expensive to produce, despite new digital manufacturing methods, such as CNC-milling and additive manufacturing (Appendix B). Complex connections and irregular members cause the price to increase dramatically. Therefore, what usually happens is that, after the structural topology optimization of an element, the design is manually modified in order to be suitable for manufacturing. This implies that the element is designed in terms of manufacturability and not entirely optimized in terms of its structural performance anymore.

One possible solution is to apply a manufacturing method called 'sand casting', which is applied in the metal industry, mainly. It is characterized by using a mold made of sand and stabilized with either clay, another bonding agent, or a vacuum. The latter is used as the chosen manufacturing method of the topology-optimized floor, and will be called 'vacuumatic formwork' (Huijben, 2014) from now on. Stabilizing the sand by means of a vacuum allows the sand to assume more complex shapes than its own angle of repose will support. Major advantages of this technique are reusability, demoldability, and adaptability. These advantages mean that the formwork cost may be reduced severely when producing floors, especially since floors have a high rate of repetition in most multi-story buildings. The reduction of waste of mold material is also a sustainable consequence.

- *Shape complexity*: The constraints on the shape of the element are small. Very complex shapes have already been realized in the metal casting industry.

- *Reusability*: The mold can be used multiple times. Several topology-optimized structures have been produced using CNC milled EPS blocks, which can also produce highly complex shapes. However, these molds can only be used once.
- *Demoldability*: Even complex shapes can be demolded easily. After the vacuum is removed, the sand falls away and the final building component remains.
- *Adaptability*: After casting an element the mold can easily adapt to other shapes. This can be executed simply by removing the vacuum, placing another pattern, and reapplying the vacuum. In this fashion, the same sand can be used for different molds.

In following sections, several methods of sand casting are discussed. There can be a clear distinction in the field it is applied in: sand casting in the metal industry and sand casting in the building industry. The first has been more extensively researched, but the second is more applicable to the subject of this thesis. Subsequently, an evaluation of the strengths, weaknesses, opportunities, and threats of the vacuumatic formwork follows (otherwise known as a SWOT analysis). The chapter concludes with several comments on the combination of topology optimization and vacuumatics.

## 5.1 SAND CASTING IN THE METAL INDUSTRY

Sand casting is the most accepted production method for both ferrous and non-ferrous metals, accounting for roughly ninety percent of all castings (Ravi, 2005). The method is used to manufacture complex, small-scale elements, with a high degree of precision. Generally, the steps involved in producing a casting using sand include pre-casting, where the sand is prepared and compacted with the use of moist sand (green-sand molding), a moist sand mold which is then baked (dry-sand molding), organic binders with high strength after baking for elements with cores (core-sand molding), or with the use of sand bound with a phenolic resin and alcohol (shell molding). The production of the master pattern can be done by any other manufacturing technique, and is often made of wood, metal, plastic, or expanded polystyrene foam. The pre-casting is followed by casting, where the metal is melted and poured into the mold. After cooling, the process finishes with post-casting. The stabilized sand is removed and the cast element is ready for inspection (Wang et al., 2010). The production procedure is displayed below.

- Pre-casting
  - Sand preparation
  - Core making
  - Molding
- Casting
  - Melting
  - Pouring
  - Cooling
- Post-casting
  - Shake-out
  - Cleaning
  - Inspection

### 5.1.1 V-process

A sand casting method with which the sand is enclosed by a flexible membrane and stabilized by means of an internal under-pressure is referred to as the V-process (Nakata and Kubo, 1974). Developed in Japan in 1969, the method is explained step-by-step in Figure 5.1. In 1995, it was reported there were roughly 180 foundries in Japan, 85 in Europe, and 10 in the USA. While the number of foundries has dropped, it still is an outstanding production method in the metal casting industry (Shengping et al., 2008). Compared to the other sand casting techniques for metal, there are many advantages, according to Clark (nd). They include a zero degree draft, which reduces weight and machining, while still allowing the cast to be removed very easily from the mold. Thin walls, a high surface finish, and tight tolerances reduce weight even further and create a product twice as accurate as typical sand castings. The unlimited pattern life and a very high return of sand are sustainable benefits.

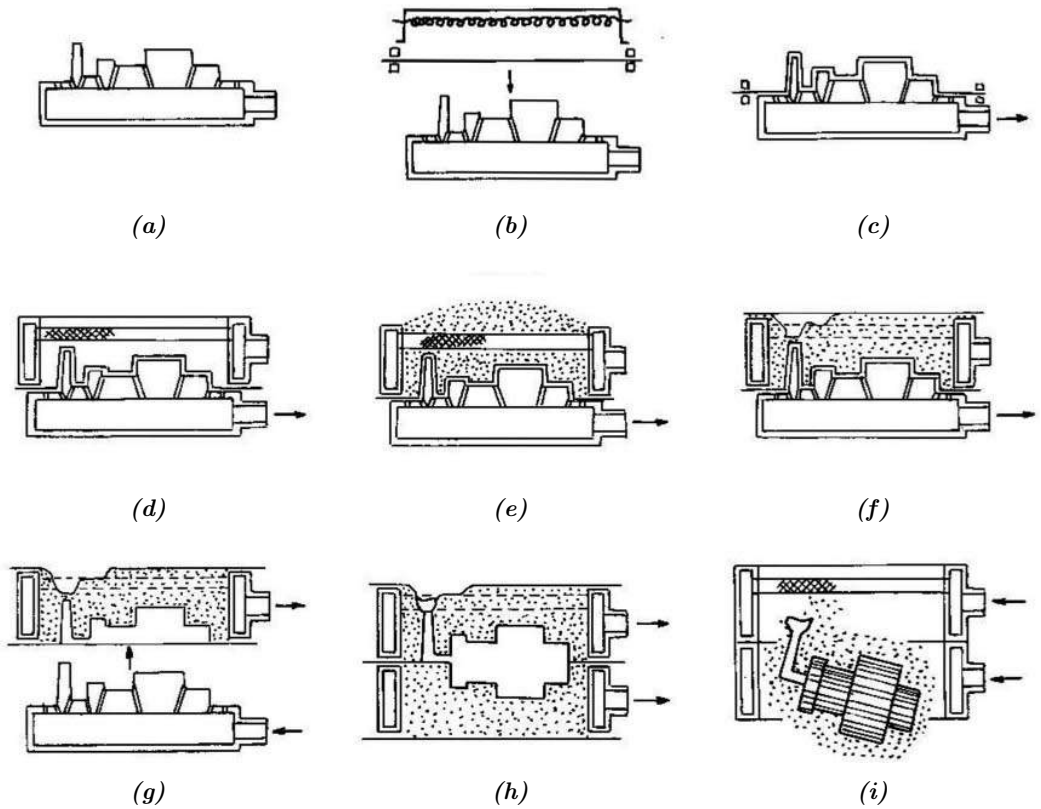



***Figure 5.1:*** *(a) The pattern is placed on a hollow plate. Vent holes allow the vacuum to reach the foil; (b) The plastic film is heated; (c) The film is placed over the pattern, vacuum shapes the foil; (d) A flask is set over the arrangement; (e) The flask is filled with dry sand and compacted; (f) A sprue cup is formed and an additional plastic film over the flask; (g) Vacuum is applied to the flask and atmospheric pressure helps preserve the shape; (h) During pouring the mold is kept under vacuum; (i) After cooling the vacuum is released, the sand falls away and a cast is left over (Kumar and Gaindhar, 1995)*

## 5.2 SAND CASTING IN THE BUILDING INDUSTRY

Where Section 5.1 focused on sand as a mold in the metal industry, this section mentions several examples where sand was used as formwork in the building industry. These examples, displayed in Figure 5.2 and Figure 5.3, are based on the natural angle of repose of sand, while vacuumatic formwork allows for more complex shapes and angles. In architecture, there are several examples of structures that utilized a sand formwork during their construction. The sand formwork was convenient for their complex double curved shapes because sand can be shaped and later reused very easily. In 1959, Heinz Isler referred to the 'freely shaped hill' as a design tool that offers a lot of freedom for design. Shells can be poured over the sand, and after hardening of the concrete, the shell can either be lifted or the sand can be excavated. Figure 5.2a displays how the hill can be used as a type of formwork. As reported by Chilton (2000), this particular example shows the beginnings of a test for an atomic shelter of unreinforced concrete.

Around the same time, Le Corbusier and Iannis Xenakis worked on the Philips electronics company pavilion for the World Expo of 1958 in Brussels (Figure 5.3a). Duyster (1958) reveals the construction method. The pavilion consisted of prefab reinforced concrete elements shaped like hyperbolic paraboloids, which were post-tensioned by steel cables between rigid edge beams. The prefab elements were manufactured with the use of sand formwork (Figure 5.3b). After hardening of the concrete, the elements were numbered and assembled on site. The reusable characteristic of the sand formwork was maximized in this structure: by changing the slope of the sand hill, it was possible to easily produce the panels with a wide range of curvatures.

The Teshima Art Museum (Figure 5.2b) is a contemporary example of sand being used as formwork. It was designed by Ryue Nishizawa and Kazuyo Sejima of SANAA architects and houses one single work of art by Rei Naito. Buntrock (2011) reports that the 250 mm thick white concrete layer was poured in-situ over the sand formwork over 22 hours. Similar to the freely shaped hill, after hardening, the sand was removed from under the concrete. The result is a wide open space of almost 2000 $m^2$, where seven-meter large holes allow the wind and rain to enter the minimalistic structure and exhibition.

***(a)*** *Heinz Isler's freely shaped hill (Chilton, 2000)*

***(b)*** *Teshima Art Museum (Buntrock, 2011)*

***Figure 5.2:*** *Sand formwork examples*

***(a)*** *Finished Philips pavilion (Arch Daily, 2011)*

***(b)*** *Sand formwork of the Philips pavilion's panels (Sijpkes, nd)*

***Figure 5.3:*** *Sand formwork examples*

## 5.3 VACUUMATIC FORMWORK

The technique of applying vacuum stabilized sand for producing free-form structures in concrete is referred to as vacuumatic formwork by Huijben (2014). It is the proposed construction method for the topology-optimized floor system. It combines the basic principles of the V-process method, with an application in the building industry. There are no examples yet of the V-process being used to cast complex reinforced concrete elements. It is assumed, however, that the advantages of the V-process translate well to a vacuumatic formwork. The benefits for concrete structures would then also include a zero degree draft, thin walls, a high surface finish, tight tolerances, an unlimited pattern life, and a reusable formwork.

Generally, the steps involved in making a reinforced concrete structure consist of the following. First, a pattern is made using any other production method. Digital manufacturing methods, such as CNC milling or additive manufacturing, are interesting options for making the often complex pattern, but traditional methods such as carpentry are also a possibility. Afterward, sand is stabilized in the shape of the pattern using a vacuum and an elastic film. The sand is now ready to be used as a formwork for the concrete, and the pattern can be used in another sand form. The same steps from the V-process are can be recognized. One major difference is that for a floor, a one-sided mold is sufficient when the concrete is cast from the top.

### 5.3.1 ISOFF-IASS WORKSHOP

At the International Association for Shell and Spatial Structures (IASS) symposium on August 16, 2015, the International Society of Flexible Forming (ISOFF) organized a two-day conference/workshop. Here, the feasibility of using vacuumatic formwork for producing (potentially topology-optimized) ribbed structures in concrete with low-tech equipment has been illustrated by Huijben (2015).

As feasibility was one of the goals of the workshop, accessible and simple tools were used to test the method. Timber planks for the outer mold, an EPS pattern, and a vacuum cleaner are all easy to come by. Highly elastic film, in this case, made from polyolefin copolymer, and a fine-grained sand complemented the applied equipment.

As mentioned before, the first step involves making the pattern. Figure 5.4a displays the pattern, which was cut from EPS. Next, the film was heated to increase the elasticity and placed over the pattern in Figure 5.4b. A vacuum was applied and the atmospheric pressure pushes the film further over the pattern. Once it all nicely fits, the timber outer mold containing the film and pattern was filled with sand in Figure 5.4c. Note that the vacuum is still applied.

Figure 5.4d shows the sand filled mold before a timber plate was placed on top and the entire unit was flipped upside down. After removing the pattern in Figure 5.4e, the mold was exposed and ready for casting the concrete in Figure 5.4f. After hardening, the concrete element was easily removed from the mold, due to the film not sticking to the structure, as displayed in Figure 5.4g. In the event of a more troublesome removal, it is possible to apply an overpressure so the element 'pops' out of its formwork.

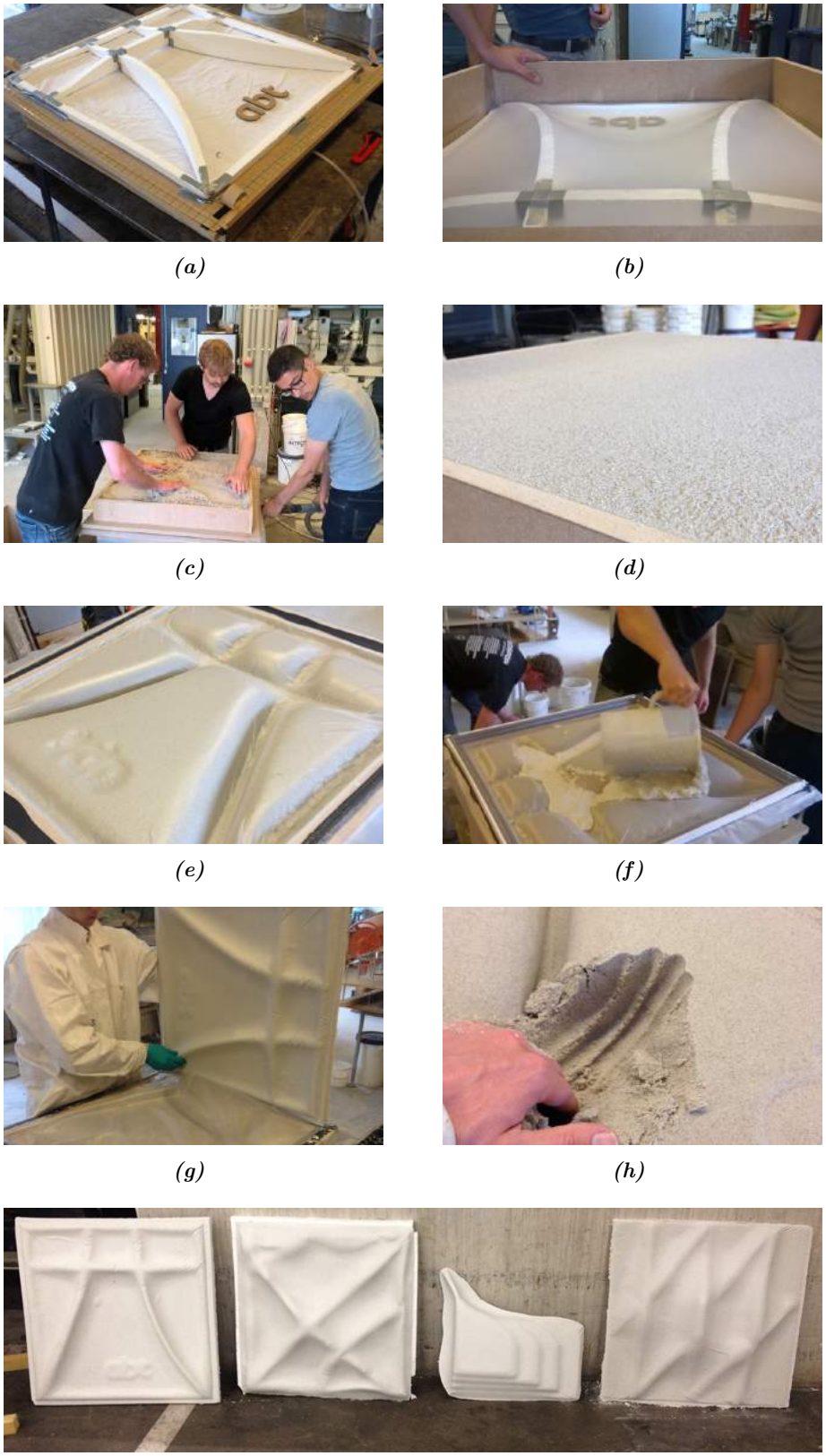
Figure 5.4h shows how the sand was fully reusable after removing the vacuum. The final result from the workshop is displayed in Figure 5.4i. The panels have a size of $700 \times 700$ mm$^2$ and a height of 100 mm.

### 5.3.2 Conclusion

Several conclusions can be drawn from the workshop. The most valuable outcome is the practical feasibility of the vacuumatic formwork, even with very low-tech equipment. While using identical tools, all four panels have a very different topology, showcasing the reusability and potential of the method.

There are many aspects that can influence the design of the mold. In the test panels, it can be seen that the ribs are shaped in a more organic form than the pattern was because the film did not completely adhere to the pattern. If a product more similar to the V-process's result is desirable, then a more elastic film and a more powerful vacuum pump are required.

Other than that, the grain size of the sand was sufficient for a smooth surface. The viscosity of the concrete mixture during casting also needs to be low enough for the mortar to reach everywhere. In this case, the fiber-reinforced UHPC mortar with white pigment was satisfactory.



*(a)* *(b)* *(c)* *(d)* *(e)* *(f)* *(g)* *(h)* *(i)*

***Figure 5.4:*** *'Vacuumatic Formwork' workshop at the ISOFF-IASS 2015 symposium*

## 5.4 SWOT ANALYSIS

One step in feasibility of the vacuumatic formwork has been performed in the form of a practical, hands-on workshop. The scale and application of the production method would be different in an actual construction process, however. In order to evaluate the method, a SWOT analysis will be discussed in this section. The goal is to find and take full advantage of the positive factors while managing or eliminating the negative ones. Strengths and weaknesses are internal factors, opportunities and threats are external factors. The analysis has been summarized in Figure 5.6.

### 5.4.1 Strengths

The strengths of vacuumatic formwork can be compared to other production methods for reinforced concrete. As mentioned in earlier in this chapter, the advantages of a vacuumatic formwork include the design freedom, a zero degree draft, thin walls, a high surface finish, tight tolerances, an unlimited pattern life, and a reusable sand core. Barring the additional research that is still required, the method has the potential to be relatively cheap for very complex geometry, as it can be applied in a low-tech fashion. The combination of these strengths is what makes vacuumatic formwork unique.

### 5.4.2 Weaknesses

One of the obvious weaknesses of the method is that fact that it's never been used before in a structure. While the workshop was successful, the actual method is on another scale and requires reinforcement bars. From the V-process it is already established that upgrading the size does not necessarily pose a problem, because the metal industry already manufactures elements of several meters long. Bending, placing, and binding of the reinforcement can be a problem, however. The bending is challenging, because of the complex shape of the ribs. Placing and binding of the reinforcement might also be complicated, as the formwork is relatively vulnerable (Huijben, 2014). Puncturing of the film, or making an unwanted imprint by workers are reasonable risks. After hardening, transport to the building site and placement of the prefab floor segments should be similar to other prefab floor systems.

### 5.4.3 Opportunities

The last few years, there has been an increase of interest in topology optimization, as evidenced by Figure 5.5. Designs inspired by, or directly taken from a topology optimization require a manufacturing method that is capable of producing them. Secondly, there are opportunities in the reduction of material usage. The more demanding requirements on sustainability, as explained in Chapter 1, make complex, slender structures more appealing.

### 5.4.4 Threats

Other manufacturing methods that allow for great design freedom include CNC milled EPS formwork and additive manufacturing. These relatively new digital production methods benefit from more popularity and, as a result, more funding. It is to be expected that their development will be quicker than the vacuumatic formwork's. Additionally, CNC milling has already been used in several structures and has found its niche in the market. An example of a socio-cultural threat would

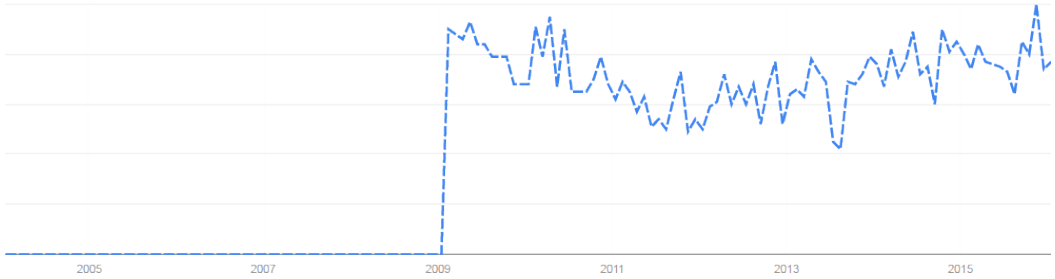



***Figure 5.5:*** *Popularity of the subject of topology optimization according to Google Trends*

be man-hour costs that remain high, meaning that a more complex structure would never be appealing, anyway, from a monetary point of view.

### 5.4.5 Conclusion

From the strengths, weaknesses, opportunities, and threats, strategies can be formed by identifying relations between these four aspects. The required research that needs to be done might benefit from the increase in popularity of topology optimization. This would then also help close the development gap with comparable digital production methods. The current high costs of labor might be intimidating, but the trend of the automation of the industry can be considered an opportunity. For example, in the future, robots might be able to cut, bend, and binding the reinforcement for a competitive price.

Fiber-reinforced concrete could also be considered a solution for the weaknesses with traditional reinforcement. Research by Hermans (2011) shows ambivalent opportunities for the topology-optimized floor slab, however. Conclusions in favor of the application are that the test-loaded floors failed at a distributed load of $12\,\mathrm{kN/m^2}$ on average, with big deflections and a well distributed cracking pattern. On the other hand, there are no regulations available yet and the test results are scattered. It would probably be safer to use traditional reinforcement bars combined with extra measures to counteract the weaknesses of the production method.

The floor would be made in prefabricated parts off-site, after which they are shipped to the building site to be combined. From restrictions on the size of a truck, it makes sense to divide a floor into parts of less than $12.0 \times 2.5$ m$^2$. At this size, the vacuumatic formwork system should still be able to operate while allowing the workers in the factory to still place and bind traditional reinforcement bars. If the risk of damaging the film or formwork is still too great, then it would be possible to pour a thin layer of concrete, wait until it is hardened, then place the traditional reinforcement, and finally pour the rest of the concrete layer. This would be similar to the technique Nervi applied in the construction of his floors.

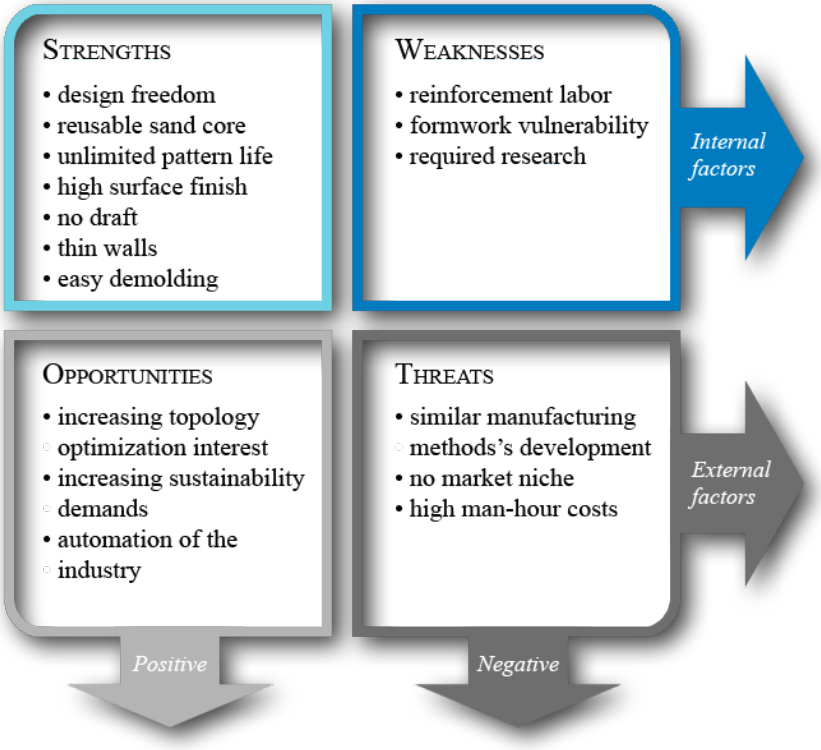



***Figure 5.6:*** *Visualization of the SWOT analysis*

## 5.5 COMBINING VACUUMATICS AND TOPOLOGY OPTIMIZATION

In this chapter, the advantages of a vacuumatic formwork were listed, and in Chapter 2, the advantages of topology optimization can be found. In summary, the strength of topology optimization lies in placing the material where it is the most efficient, resulting in relatively lightweight structures with organic shapes. For traditional manufacturing methods, this means that many man-hours are required to produce the formwork. In Western countries, man-hours are the major component of production costs, meaning that topology-optimized designs are extremely expensive.

These highly organic shapes that follow from topology optimization can be scaled down considerably by including manufacturability constraints inside the design process. The second solution to potentially decreasing production costs is the application of vacuumatic formwork. This manufacturing method is capable of producing complex shapes with a very low-tech procedure. Advantages in reusability and adaptability mean that it is also a sustainable technique. The costs of very expensive structures that follow from traditional topology optimization can be reduced by utilizing vacuumatics. The extent to which that is possible is not yet clear for this new production method, however. An introducing analysis is made in Section 7.1.2 but further research is needed to reveal how vacuumatic formwork truly influences the production costs.

As displayed in Table 5.1, vacuumatics strengthens the advantages of topology optimization, while counteracting the disadvantages.

***Table 5.1:*** *Advantages and disadvantages of topology optimization and vacuumatics*

| Topology optimization | Vacuumatic formwork |
|---|---|
| + Sustainability increase | + Sustainable manufacturing technique |
| + Weight reduction | |
| + Aesthetics increase | |
| − Complex shapes are difficult to manufacture | + Capable of producing complex shapes that are easily demolded |
| − Complex shapes are expensive to manufacture | + Low-tech procedure that has the potential to be inexpensive |

# 6

# Topology-optimized floor slab design

In this chapter, topology optimization is combined with vacuumatic formwork and applied to the challenge of designing a concrete floor slab. The input values for the algorithm are given, so the problem can easily be recreated by anyone who wishes to do so. The changes that have been made compared to the standard topology optimization algorithm from Chapter 4 are described from Section 6.1 to Section 6.4. Afterward, the final topology-optimized design is displayed in Section 6.5. It is developed with all the manufacturability constraints and can be produced with a vacuumatic formwork. After the final design has been obtained, Chapter 7 researches whether the advantages of the topology-optimized floor, predicted in Chapter 1, were correctly assumed.

## 6.1 ADDITIONAL INPUT BY THE USER

The input of the algorithm consists of all the factors that may affect the end result. In order to arrive at a better floor slab design, several additional input variables have been added to the standard algorithm. They are used to activate the manufacturability and functionality constraints and to define extra material properties. Table 6.1 elaborates on the definitions of the additional input variables.

***Table 6.1:*** *Input variables*

| Input variable | Description |
|---|---|
| `Etmax` | Maximum Young's modulus, applied in a fully solid element that is under tension. |
| `ft` | Elemental force at which an element is considered to be cracked concrete. |

***Table 6.1:*** *Input variables*

| Input variable | Description |
|---|---|
| `Emin` | Minimum Young's modulus, applied in an 'empty' element. The value is required to be greater than zero to prevent singular matrices. For the modeling of concrete, it is advised to apply a value of 100. |
| `specweight` | Specific weight of the considered material. |
| `activetop` | Implementation of a mandatory top layer when set to `1` |
| `activetopextra` | Implementation of a second mandatory top layer when set to `1` |
| `selfweight` | Implementation of self-weight loading when set to `1` |
| `castingconstraint` | Implementation of the casting constraint when set to `1` |

## 6.2 SELF-WEIGHT

### 6.2.1 Input value

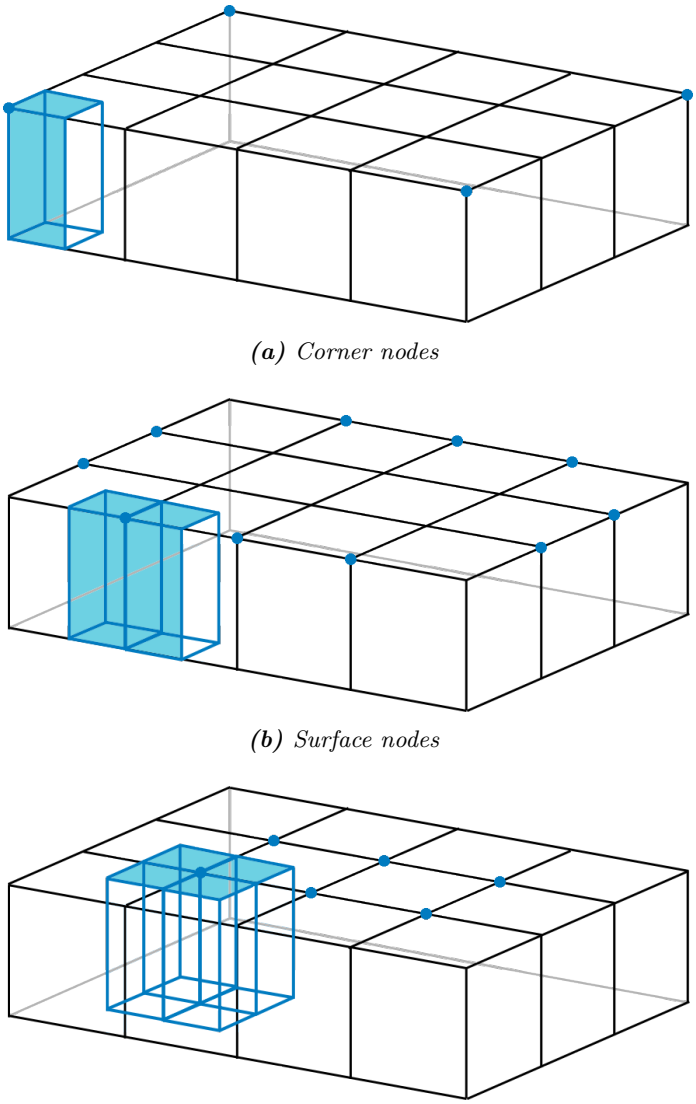
In addition to the live load from Section 4.9.2, self-weight is also modeled, since it forms a significant part of the total floor slab loading. The self-weight can be activated by setting the option `selfweight` to `1`. A concrete specific weight of $24.0\,\mathrm{kN/m^3}$ is assumed as displayed on the next line and in Table 6.2.

```
specweight = -24.*(7.*2.5*0.375)/(nelx*nely*nelz)
```

### 6.2.2 Implementation in the algorithm

The self-weight of a single element in the model is directly related to the relative stiffness of that corresponding unit. As Equation (6.1) implies, this means it differs slightly from the definition of the elemental stiffness in which a penalization is applied. Because of the penalty factor in the stiffness function, the solution of the topology optimization already approaches a black-and-white structure. It is, therefore, unnecessary to also penalize the self-weight. Additionally, it prevents the algorithm from not achieving convergence.

$$\begin{aligned} F_{e;sw} &= x_e \cdot \rho_0 \\ E_e &= x_e^p \cdot E_0 \end{aligned} \tag{6.1}$$



*(a) Corner nodes*

*(b) Surface nodes*

*(c) Inside nodes*

***Figure 6.1:*** *Self-weight distribution over different node locations of one layer of a 4x*`nely`*x3 design space*

In the algorithm, forces are applied on nodes, while the self-weight actually belongs to an element. This is modeled by averaging the weight of its neighboring elements to a node. Since the number of neighboring elements differs for corner nodes, surface nodes, and inside nodes, the definition also differs for each of these types of elements. Figure 6.1 and Algorithm 5 display how this is accomplished. Corner nodes transfer the weight of a quarter of its element. Surface nodes average the weight of two elements, and inside nodes average the weight of four neighboring elements.

**Algorithm 5** Self-weight

**procedure** DEF SW($xPhys, specweight, ndof, nelx, nely, nelz$)
DEFINE empty self weight matrix
**for** each corner of the design space **do**
    DEFINE node IDs
    COMPUTE degrees of freedom over the y-axis per node
    COMPUTE element corresponding to the node
    UPDATE the self-weight matrix
**for** each surface on the side of the design space **do**
    DEFINE node IDs
    COMPUTE degrees of freedom over the y-axis per node
    COMPUTE the two elements corresponding to the node
    UPDATE the self-weight matrix
**for** each layer of inside nodes in the z-direction **do**
    DEFINE node IDs
    COMPUTE degrees of freedom over the y-axis per node
    COMPUTE the four elements corresponding to the node
    UPDATE the self-weight matrix
RETURN self-weight matrix `Fsw`

### 6.2.3 Validation

For the validation of the self-weight, an arch has been researched. For the self-weight arch, only the self-weight of the structure is applied. The boundary conditions are the same as for the wheel in Section 4.6. The model is displayed in Figure 6.2 and optimized with the following input: `(24,15,24,0.06,3,1.5)`. It is clear that result from literature is the stiffer structure. Differences between the topologies can be explained by the mesh density and optimization method. At the same time, the modeling of the self-weight is considered to be accurate enough for next chapter. While the topologies are not completely the same, the result from the algorithm is still definitely an arch.

*(a) Design space*

*(b) Result from literature (Huang and Xie, 2010)*

*(c) Result from the algorithm*

***Figure 6.2:*** *Self-weight arch comparison*

## 6.3 CONCRETE PROPERTIES

Reference projects of built topology-optimized designs do not offer a solution for the choice of concrete strength and type of reinforcement. While regular strength concrete is often used, there are multiple projects where a higher strength class, self-compacting, and/or sprayed concrete is applied. In this thesis, it is assumed that the floor slabs are prefabricated in a factory off-site. Prefab elements usually utilize a higher strength grade concrete than cast in-situ elements. The higher strength concrete hardens quicker, which means that they can be demolded earlier and that more elements can be produced per day. Therefore, the relatively high concrete strength class C50/60 is considered for the floor slab design in this thesis. It is advised to use a concrete mixture with self-compacting abilities, in order to prevent technicalities during casting with the vulnerable vacuumatic formwork. The values that would normally be used in the analysis of a concrete structure are displayed in the third column of Table 6.2. The properties as applied in the algorithm for the floor are displayed in the fourth column.

***Table 6.2:*** *Properties for C50/60 as assumed in regular concrete models compared to the ones in the topology optimization algorithm*

| Material property | | Regular concrete modeling | Topology optimization algorithm | Unit |
|---|---|---|---|---|
| Uncracked Young's modulus | $E_{cm}$ | 37,000.00 | 37,000.00 | MPa |
| Cracked Young's modulus | $E_{ctm}$ | 12,000.00 | 12,000.00 | MPa |
| Compressive strength | $f_{ck}$ | 50.00 | N/A | MPa |
| Tensile strength | $f_{ctk,0.05}$ | 2.90 | N/A | MPa |
| Specific weight | $\gamma_c$ | 24.00 | 24.00 | kN/m$^3$ |
| Poisson's ratio | $\nu$ | 0.20 | 0.20 | – |

### 6.3.1 Implementation in the algorithm

In the previous chapter, it was determined that in topology optimization, the models generally consist of linear, isotropic elements, with equal compressive and tensile properties. As displayed in Table 6.2, in reality, concrete performs very well under compression but fails relatively quick under tension. The addition of reinforcement bars results in a composite, inhomogeneous material that should be described differently from the elements traditionally used in topology optimization. The reinforced concrete acts nonlinear when taking cracking into account and the addition of reinforcement bars means that it acts anisotropic. Therefore, for this thesis, the material definition has been changed to model concrete more accurately than the basic topology optimization algorithm.

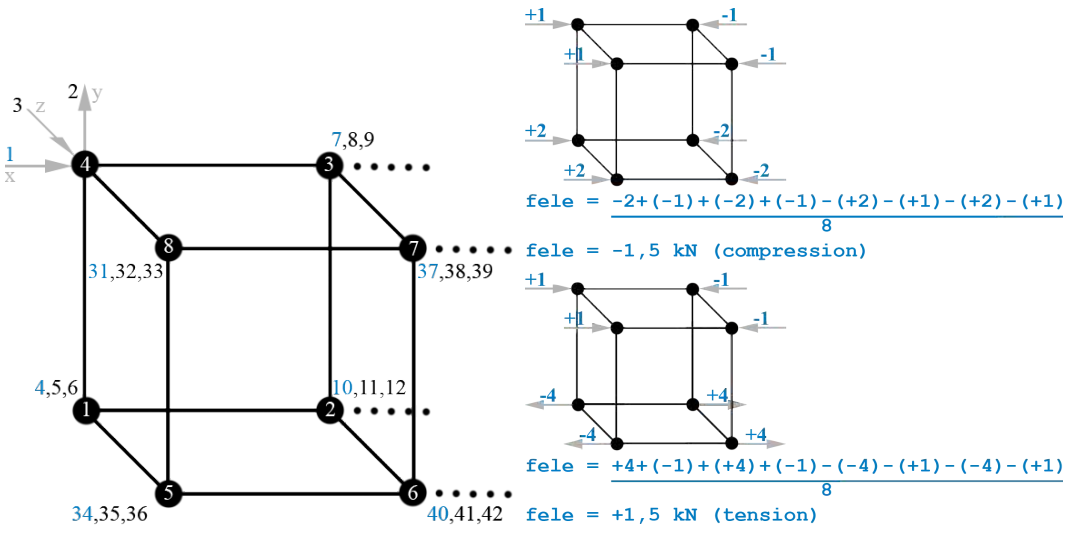
In order to allow the algorithm to also take the cracked Young's modulus into account, several lines in the algorithm need to be adjusted. First of all, both the cracked Young's modulus and the stress at which concrete cracks need to be defined in the input values.

```
Etmax = float(12000000.0)
ft = float(0.0)
```

Then, in the FEA part of each iteration, the cracked stiffness is also included in the global stiffness matrix. This involves altering the computation of the entry vector `sK` so that it uses `Ecmax` when an element is under compression and it uses `Etmax` when an element is under tension. Determining when an element is under compression or tension is achieved by applying Equation (4.4), which gives the force in each node. If the average of the nodal forces in the x-direction in an element is negative, then the element is considered to be compressed. In Figure 6.3, the blue numbers indicate the nodal forces that are taken into account (which follow the numbering of the degrees of freedom), while the black circles indicate the local node numbering (which differ from the global node numbering). The alteration is simplified in Algorithm 6. The exact changes in the algorithm can be found in Appendix G.

**Algorithm 6** Global stiffness entry vector assembly

```
procedure DETERMINE sK
    for every element do
        if it is the first iteration then
            DEFINE forces as zero because they are still unknown
        COMPUTE nodal forces corresponding to the element
        COMPUTE average elemental force
        if element is under compression then
            COMPUTE entry vector with compressive Young's modulus
        if element is under tension then
            COMPUTE entry vector with tensile Young's modulus
    RETURN global stiffness entry matrix sK
```




*(a) Force numbering* *(b) Examples*

***Figure 6.3:*** *Utilized forces (in blue) in determining the stress state of the first element in a 4x1x1 design space*

The objective function `c` and the sensitivities `dc` are also calculated with the Young's modulus. Therefore, they also require an adjustment. Similar to the new definition of the global stiffness matrix, `Ecmax` is applied when an element is under compression and `Etmax` when an element is under tension. The alteration is simplified in Algorithm 7. The exact changes in the algorithm can be found in Appendix G.

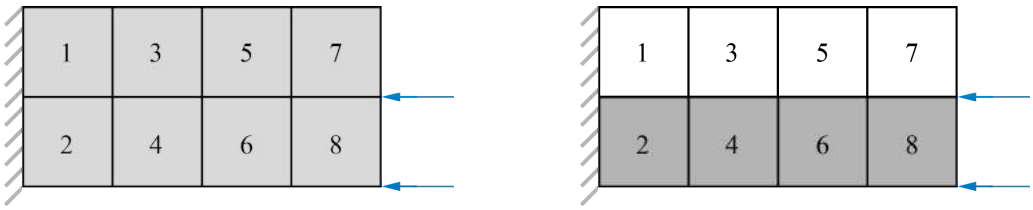
**Algorithm 7** Objective and sensitivity

```
    procedure DETERMINE C AND DC
2:      for every element do
            COMPUTE nodal forces corresponding to the element
4:          COMPUTE average elemental force
            if element is under compression then
6:              COMPUTE objective with compressive Young's modulus
                COMPUTE sensitivity with compressive Young's modulus
8:          if element is under tension then
                COMPUTE objective with tensile Young's modulus
10:             COMPUTE sensitivity with tensile Young's modulus
        RETURN objective c
12:     RETURN sensitivity dc
```

### 6.3.2 Validation

Validation of the method of modeling the cracked stiffness of concrete in the tension zone has been performed by two simple structural examples. The first is a compression/tension beam, displayed in Figure 6.4, and the second is the MBB beam, displayed in Figure 6.5. The displacements from the model have been compared to the ones that are expected according to structural mechanics in Table 6.3 and Table 6.4.

For a compression/tension beam with length $L$, cross section area $A$ and Young's modulus $E$ that is loaded with four forces $F$ in the corners, the expected horizontal displacement of the end of the beam can be defined as: $4F/EA \cdot L$. Assuming that $L = 4$ m, $A = 1$ m$^2$, $E_c = 1$ MPa, $E_t = 0.3$ MPa, and $F = 1000$ N, then the horizontal displacement is equal to $-16$ mm for the compression beam and 53 mm for the tension beam, in theory. For the algorithm, the problem as displayed in Figure 6.4 is used as input, in which the allowed volume fraction is 50%, the penalty is 3.0, the minimum thickness is 1.0 element. When using kilo Newtons and meters as the units, then the load, compressive and tensile Young's modulus are 1 kN, 1000 kN/m$^2$ and 300 kN/m$^2$, respectively. The model converges in 5 iterations and the result is as displayed in Figure 6.4b and Table 6.3. It can be concluded that the displacements are correct and that the appropriate Young's modulus is applied in the matching situation.




*(a) Problem* *(b) Solution*

***Figure 6.4:*** *Compression beam model for the validation of the cracked stiffness*

***Table 6.3:*** *Horizontal displacement of the end of the compression/tension beam*

| | Algorithm (mm) | Theory (mm) | Accuracy (%) |
|---|---|---|---|
| Compression | -15.86 | -16.00 | 99.1 |
| Tension | 52.91 | 53.33 | 99.2 |

The MBB beam has been used to validate the stresses that occur in a beam when subjected to a point load at the center top. For the design space, displayed in Figure 6.5, it is expected that the upper row of elements is under compression, while the lower row is under tension. When taking the reduced stiffness of cracked concrete into account, the lower row should have a Young's modulus of roughly one-third of its maximum elastic modulus. Multiple combinations of mesh densities and stiffness ratio's have been checked in Table 6.4.

For a beam with length $L$, Young's modulus $E$, and second moment of area $I$, that is loaded with a single force $F$, the expected vertical displacement at $0.5L$ is given by $\frac{1}{48}\frac{FL^3}{EI}$. Assuming that $L = 8$ m, $E_c = 1.0$ N/mm$^2$, $E_t = 0.3$ N/mm$^2$, width $b = 1$ m, height $h = 2$ m, and $F = 1000$ N, then the following values are returned. The maximum deflection is $-16$ mm for a beam with only the compressive Young's modulus, $-53$ mm for a beam with only the compressive Young's modulus, and $-31$ mm for a beam with both the compressive and tensile Young's modulus.

For the algorithm, the problem as displayed in Figure 6.5 is used as input, in which the allowed volume fraction is 99%, the penalty is 1.0, the minimum thickness is 1.0 element. When using kilo Newtons and meters as the units, then the load, compressive and tensile Young's modulus are 1 kN, 1000 kN/m$^2$ and 300 kN/m$^2$, respectively. The model converges in 2 iterations and results in the deflections as displayed in Table 6.4. It can be concluded that the maximum vertical displacements are not completely accurate. They are in the same order of magnitude, however, so they are considered precise enough for the analysis of the concrete floor slab. Additionally, the error has no significance when comparing different floor slab designs that have been found with the same topology optimization algorithm.

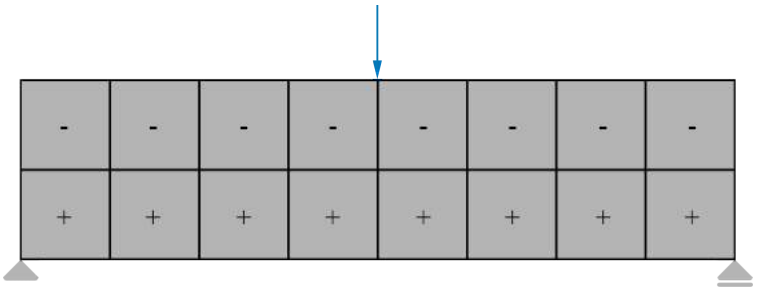



***Figure 6.5:*** *MBB beam model for the validation of the cracked stiffness*

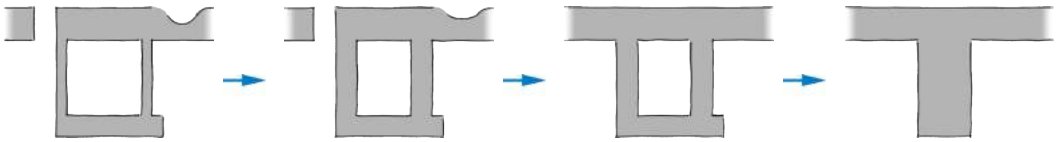
***Table 6.4:*** *Maximum vertical displacement of the MBB beam*

| | Mesh density | Algorithm (mm) | Theory (mm) | Accuracy (%) |
|---|---|---|---|---|
| `Ecmax=1000, Etmax=1000` | $8 \times 2 \times 1$ | -19.26 | -16.00 | 120 |
| | $16 \times 4 \times 2$ | -18.83 | | 118 |
| | $32 \times 8 \times 4$ | -20.07 | | 125 |
| `Ecmax=1000, Etmax=300` | $8 \times 2 \times 1$ | -39.98 | -31.46 | 127 |
| | $16 \times 4 \times 2$ | -39.04 | | 124 |
| | $32 \times 8 \times 4$ | -39.23 | | 125 |
| `Ecmax=300, Etmax=300` | $8 \times 2 \times 1$ | -64.20 | -53.33 | 120 |
| | $16 \times 4 \times 2$ | -62.78 | | 118 |
| | $32 \times 8 \times 4$ | -66.89 | | 125 |

It is important to note, that due to the method that determines whether an element is considered under tension or not, this method of modeling concrete only works for cracking due to forces in the x-direction. That is acceptable for the floor slab that is modeled in this thesis which spans in the same direction, but for many other problems this definition is not valid. Cracking due to shear force is also not considered. Furthermore, the loops that are used to model the cracked concrete slow the optimization process down considerably. As a result, the standard topology optimization algorithm has a speed advantage for regular topology optimization problems.

## 6.4 ADDITIONAL CONSTRAINTS

In order to arrive at a more manufacturable result, several constraints have been added to the algorithm. They listed in this section. The constraints that are derived from manufacturability with a vacuum stabilized sand mold are based on the findings in Appendix B. The flat top surface constraint follows from functional requirements in the utilization stage. Figure 6.6 visualizes how these additional constraints influence the result.



*(a) Topology optimization result* *(b) Inclusion of minimum thickness* *(c) Inclusion of flat top* *(d) Inclusion of casting constraint*

***Figure 6.6:*** *Visualizations of the functioning of the additional constraints for a hypothetical cross-section*

### 6.4.1 Maximum curvature surface

The maximum curvature of the surface seemed to be a manufacturing constraint during the test casts at the ISOFF-IASS 2015 symposium from Section 5.3. It occurs when either the film that is placed over the pattern is not elastic enough or when the applied partial vacuum pressure is not sufficient. This then results in a mold that does not accurately follow the pattern. However, from the metal casting industry, it is known that in practice there are films that are able to follow complex shapes. For this thesis, it is therefore assumed that the maximum curvature of the surface constraint is not of importance to the optimization model.

### 6.4.2 Casting constraint and void removal

The casting constraint places limits on how material is distributed in the design space. During construction, it makes sense for a floor to have the vacuumatic formwork on the bottom and to pour the concrete from the top. This results in the casting constraint, visualized in Figure 6.8. While the vacuumatic formwork does allow for holes in the structure when two separating formwork parts are utilized, as displayed in Figure 6.7a and Figure 6.7b with a polystyrene mold, for the floor design in this thesis, it is assumed only a one-sided formwork is allowed, similar to the method applied in the ISOFF-IASS 2015 symposium. When looking at Figure 6.7c, the displayed undercut can be manufactured with vacuumatics. However, this involves pushing the film back by hand, which in practice is not desired in the production process since it is needlessly complex. The casting constraint can be activated by setting the option `castingconstraint` to `1`. With the option enabled, the final design is able to be cast with a one-sided mold.


*(a) Separating formwork halves (Van Velden, 2015)* *(b) Result by Movares from the halves* *(c) Undesired undercut*

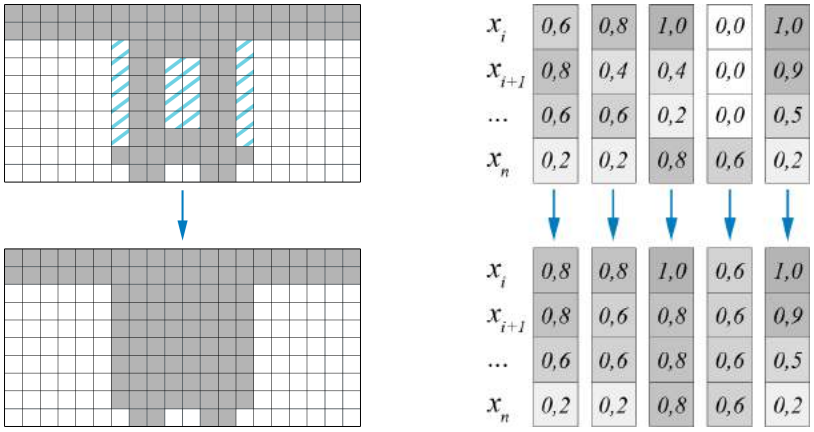
***Figure 6.7:*** *Relevance casting constraint*

### Implementation in the algorithm

In the algorithm, the casting constraint is activated with the use of the definition called `cc`, detailed in Figure 6.8b and Algorithm 8. In this definition, an element has its density updated whenever there is an element below it with a larger density:

$$x_i \geq x_{i+1} \geq x_{i+2} \geq ... \geq x_n \tag{6.2}$$

The casting constraint is called during the Optimality Criterion of Section 4.5.2, which means that the density of the elements is updated during each iteration, rather than at the end of an iteration. Because of the penalty factor, the final result can still be interpreted as a black-and-white solution.




***(a)*** *Problem and solution, showing one void and two undercuts*

***(b)*** *Examples of the applied logic*

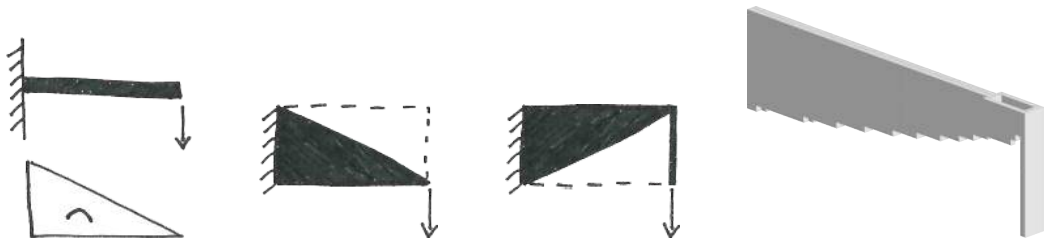
***Figure 6.8:*** *Casting constraint*

**Algorithm 8** Casting constraint

**procedure** DEF CC($xnew, nelx, nely, nelz$)
  **for** each column in the design space **do**
    **for** each element in the column, starting from the bottom **do**
      **if** current element density is greater than above element density **then**
        UPDATE above element density to match current element density
  RETURN updated density array `xnew`

### Validation

There are no examples available from literature to check whether the casting constraint has been correctly implemented. However, when considering Cantilever beam A from Figure 4.12, a prediction can be made for the result when the casting constraint is activated. By simplifying the problem to a one-dimensional beam, the bending moment can be found. Figure 6.9 displays a prediction of the solution and the one found by the algorithm. For a volume fraction of 30%, the algorithm needs 171 iterations to reach the optimum. It can be concluded that the casting constraint functions as required.



*(a) One dimensional simplification*

*(b) Size optimization*

*(c) Predicted topology optimization with casting constraint*

*(d) Found solution with the algorithm*

***Figure 6.9:*** *Validation casting constraint*

### 6.4.3 Minimum thickness

Having a minimum thickness of the concrete is essential because the result of a topology optimization can have very thin members. These thin members may perform very well in creating a stiff structure, which the algorithm optimizes for, but they can also be prone to breakage in practice. A minimum thickness constraint prevents this problem. In the Python algorithm, the `rmin` variable represents the minimal member radius. Assuming that the concrete cover for the reinforcement is 30 mm, the reinforcement itself consists of bars of 20 mm and some room for internal spacing, then the minimum thickness can be presumed to be roughly 170 mm, which corresponds to `rmin = 2.5` elements in the design space.

### 6.4.4 Flat top surface

A floor without a flat top surface would not fulfill its function very well. As a consequence, the top of the design space needs to be flat. This can be accomplished by prohibiting the top layer(s) of elements from taking an 'empty' value. These are then called 'active elements'. That way, they are always present in the final design. The flat top surface can be triggered by setting `activetop` and `activetopextra` to `1`. In each iteration, it then returns the topmost elements, defined as `activeele`, to their 'full' state in the physical identities `xPhys`.

## 6.5 TOPOLOGY-OPTIMIZED CONCRETE FLOOR SLAB DESIGN

In this section, several topology-optimized floors are displayed. In theory, the most optimal design is the one without any constraints in Section 4.9, but for manufacturing a concrete floor, it is far from practical. Step by step, the aforementioned features are included, which makes the designs less theoretically optimal, but more functional and suitable for manufacturing. The topology optimization process has been repeated for each additional feature, continuing from the previous optimum, so that the final result is close to the global optimum. Otherwise, the optima that would be found would be almost guaranteed to be local and far removed from the global minima.

The final floor slab design is demonstrated in Section 6.5.4. It is able to be produced with a vacuumatic formwork and has advantages in weight reduction, sustainability, and aesthetics over traditional floors systems. These claims are verified in Chapter 7.

### 6.5.1 Concrete topology optimization

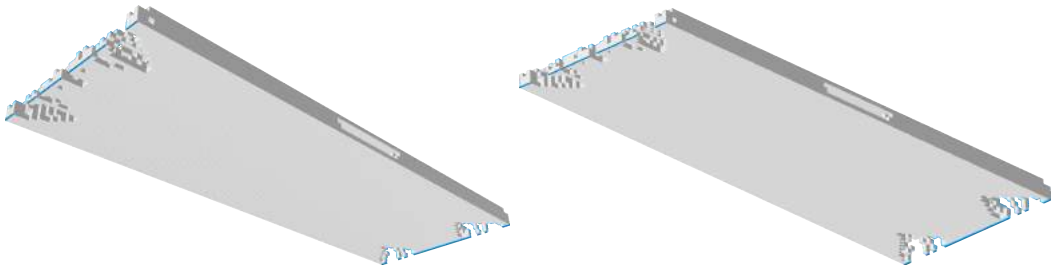
The problem for a concrete floor is similar to the basic topology optimization of Section 4.9 without any manufacturability constraints. The difference can be found in the way the material is modeled. Compared to the previous design, several factors have been changed. The Poisson's ratio is adjusted from 0.3 to 0.2, the self-weight of the floor is included in the load, and the reduced Young's modulus for elements under tension is also activated. In theory, this results in the most optimal concrete structure according to the objective function. Since the self-weight changes the load conditions and the reduced stiffness changes the global stiffness matrix and objective in each iteration, compared to the design in Section 4.9, the optimization takes much more iterations to converge and might show erratic behavior.

The optimization process with the objective and maximum deflection is displayed in Figure 6.10c. Each rise indicates another change to the topology optimization process being activated. It can be seen that each addition to design process of the floor results in a less optimal design compared to the theoretical optimum.

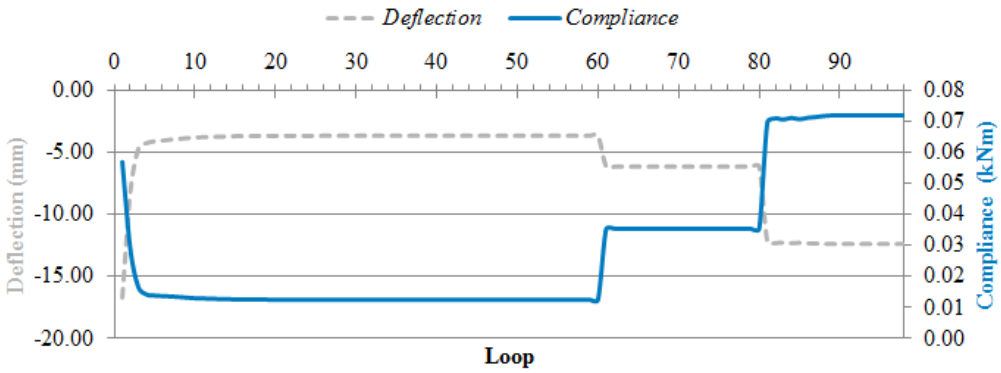
The concrete floor slab design in Figure 6.10 has a maximum deflection of 12.4 mm and an objective of 0.0720 kN m. The design shows similarities with the basic topology optimization. The topology itself does not differ very much because the influence of the Poisson's ratio is not very big, the self-weight for a thin plate is very similar to an equally distributed load that is already present, and because even though tensile elements have a lower Young's modulus, they cannot be avoided. The maximum deflection is more realistic, however. The design still shows the same manufacturing complications as before.

***Table 6.5:*** *Input variables concrete topology optimization*

| Input variable | Value |
|---|---|
| `nelx` | 112 |
| `nely` | 4 |
| `nelz` | 40 |
| `volfrac` | 0.575 |
| `penal` | 3.0 |
| `rmin` | 0.5 |
| `activetop` | off |
| `selfweight` | on |
| `castingconstraint` | off |



*(a) Perspective projection* *(b) Parallel projection*




*(c) Optimization diagram*

***Figure 6.10:*** *Optimal design according to concrete topology optimization*

### 6.5.2 Inclusion of a minimum thickness

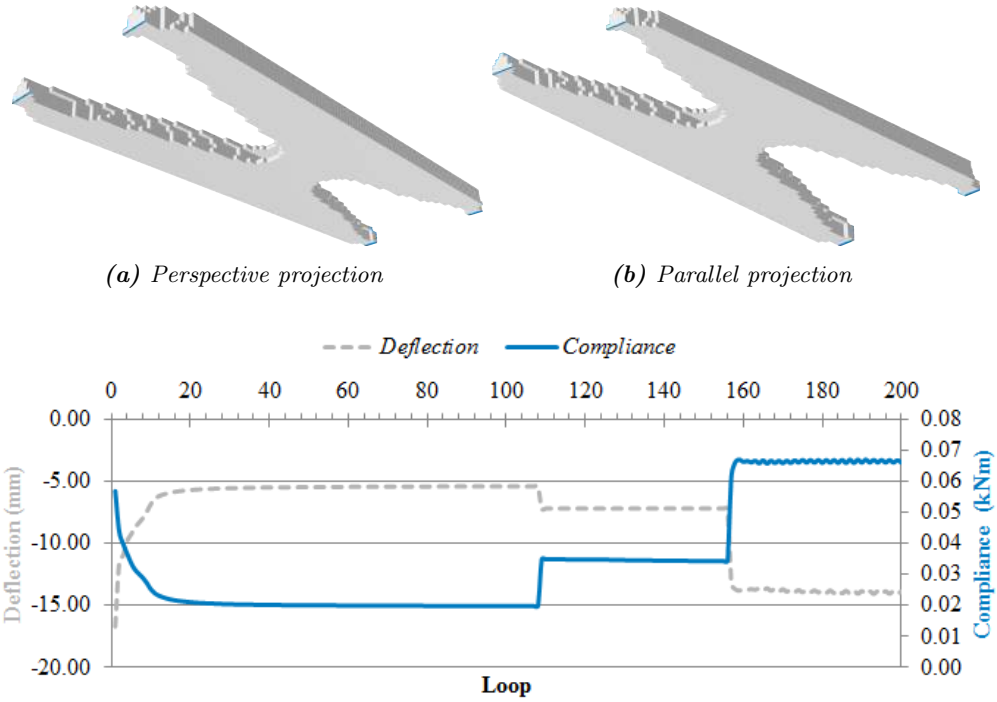
The design in Figure 6.10 shows several very thin members. These can be removed or made thicker with the inclusion of a minimum member radius. As determined in Section 6.4, the minimum radius is set to 170 mm. The floor slab in Figure 6.11 demonstrates how the minimum thickness influences the design. The very thin members are removed and instead the material tends to stick together, creating two beams on the sides. They are still connected in the center so that they work together and a gap is also still visible at midspan. At the same time, the required number of iterations increases considerably, partly due to the oscillating effect of the reduced Young's modulus for tensile elements.

***Table 6.6:*** *Input variables concrete topology optimization with a minimum thickness*

| Input variable | Value |
|---|---|
| `nelx` | 112 |
| `nely` | 4 |
| `nelz` | 40 |
| `volfrac` | 0.575 |
| `penal` | 3.0 |
| `rmin` | 2.5 |
| `activetop` | off |
| `selfweight` | on |

***Table 6.6:*** *Input variables concrete topology optimization with a minimum thickness*

| Input variable | Value |
|---|---|
| `castingconstraint` | off |




***(a)*** *Perspective projection* ***(b)*** *Parallel projection*

***(c)*** *Optimization diagram*

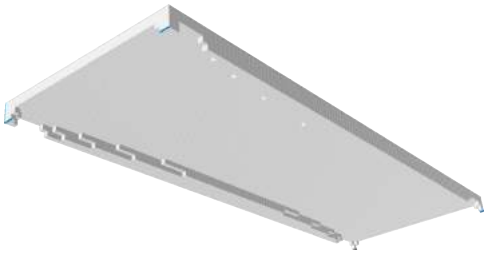
***Figure 6.11:*** *Optimal design according to concrete topology optimization with a minimum thickness*

### 6.5.3 Inclusion of a flat top

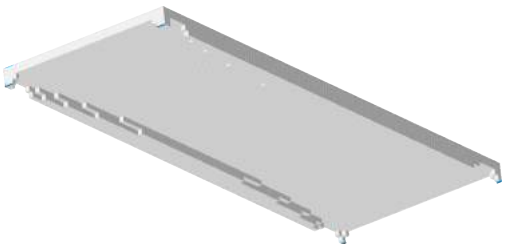
The previous floor slab designs show two big gaps in its surface. This can be resolved with the inclusion of the flat top. For this particular mesh density, the upper two elemental layers are active, rather than only the first layer. Now that elements are required to be present in the top layers, that means that less material has the freedom to be placed in an effective location. The top consists of 50% of the volume fraction, leaving only 7.5% to be distributed over the rest of the design space for stiffening. Compared to the previous section, this difference is immediately noticeable in Figure 6.12. While this particular outcome already seems suitable for manufacturing, the result can contain undercuts for taller design spaces or for denser meshes, as displayed in Figure 6.13.

***Table 6.7:*** *Input variables concrete topology optimization with a minimum thickness and a flat top*

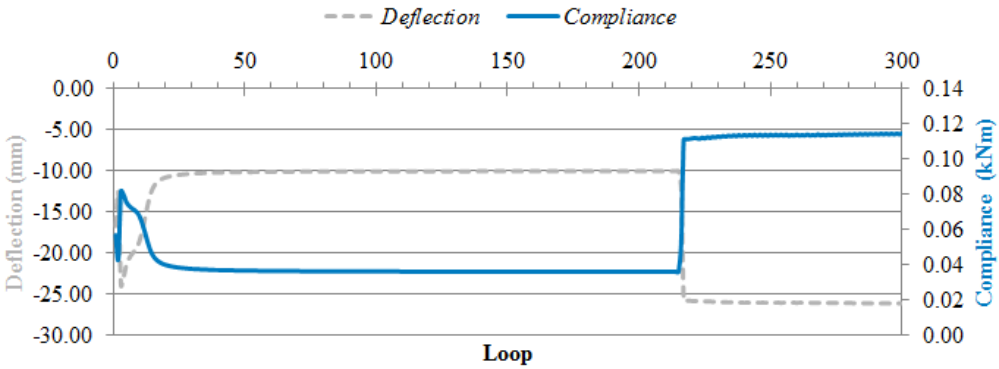
| Input variable | Value |
|---|---|
| `nelx` | 112 |
| `nely` | 4 |
| `nelz` | 40 |
| `volfrac` | 0.575 |
| `penal` | 3.0 |
| `rmin` | 2.5 |
| `activetop` | on |
| `selfweight` | on |
| `castingconstraint` | off |



***(a)*** *Perspective projection*

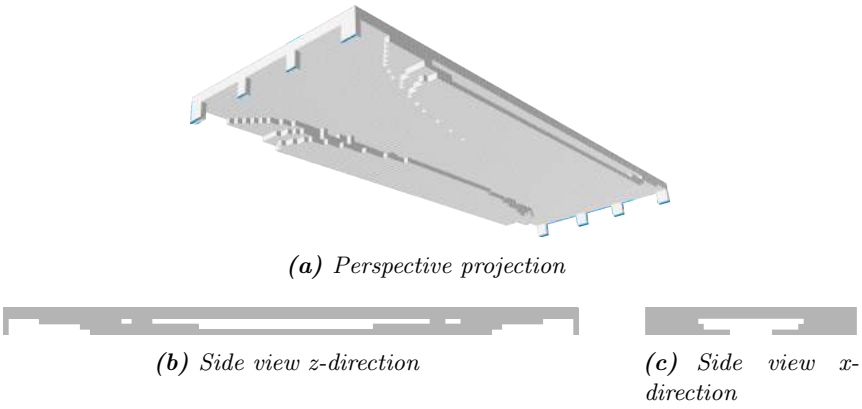


***(b)*** *Parallel projection*




***(c)*** *Optimization diagram*

***Figure 6.12:*** *Optimal design according to concrete topology optimization with a minimum thickness and a flat top*



*(a) Perspective projection*

*(b) Side view z-direction*

*(c) Side view x-direction*

***Figure 6.13:*** *Possible outcome with undercuts and gaps for taller design space or denser mesh*

### 6.5.4 Inclusion of the casting constraint and final result

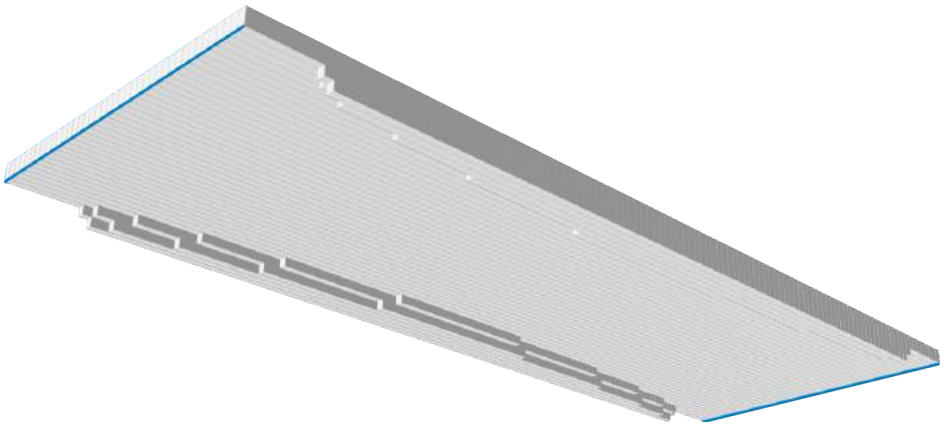
The undesirable undercuts of the previous section can be removed with the use of the casting constraint. The result of the topology optimization with the input from Table 6.8 is displayed in Figure 6.14, for the following variables. The design has a maximum deflection of 22.7 mm and an objective of 0.0884 kN m.

***Table 6.8:*** *Input variables concrete topology optimization with a minimum thickness, a flat top, and a casting constraint*

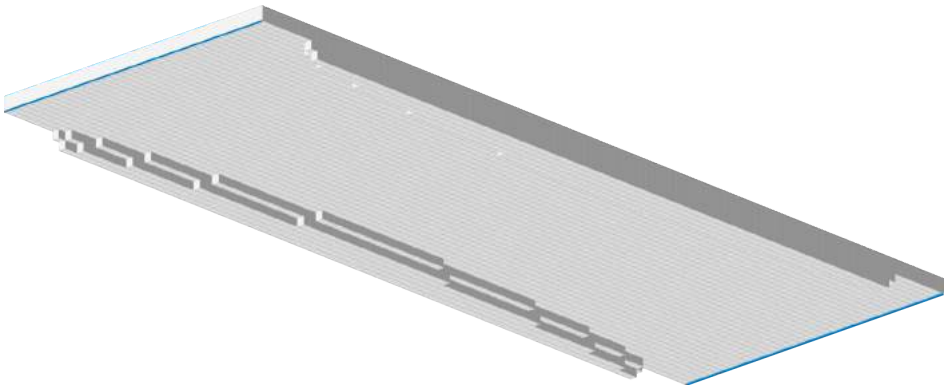
| Input variable | Value |
|---|---|
| `nelx` | 112 |
| `nely` | 4 |
| `nelz` | 40 |
| `volfrac` | 0.575 |
| `penal` | 3.0 |
| `rmin` | 2.5 |
| `activetop` | on |
| `selfweight` | on |
| `castingconstraint` | on |

In iteration 237, the casting constraint is added. It is interesting to note that this change in material distribution results in only a small increase, while at the same time making it much more suitable for manufacturing.

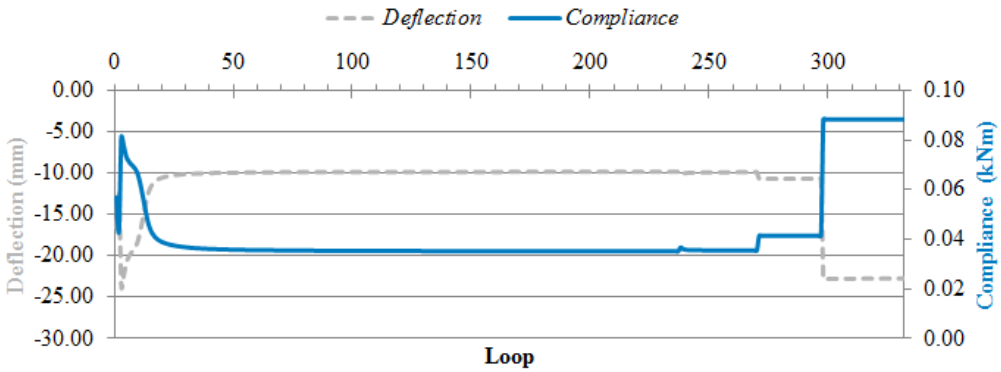
In iteration 271, the self-weight causes a jump in the objective function. Only 27 iterations are required to converge again. This is caused by the self-weight for a thin plate having a very similar distribution to the equally distributed load. Therefore, the design does not need to adapt very much.



*(a) Perspective projection*



*(b) Parallel projection*




*(c) Optimization diagram*

***Figure 6.14:*** *Optimal floor slab design according to concrete topology optimization with a minimum thickness, a flat top, and a casting constraint*

The last round in the optimization process starts in iteration 298 and includes the reduced Young's modulus for tensile elements. The design is already almost converged due to the previous stages and as such, the objective function only reduces 0.000 016 kN m until the last iteration.

In Chapter 3, several predictions for a ribbed floor slab system were made with the help of isostatics. While the process behind it differs from the topology optimization process, the result of that analysis can be used to compare the topology-optimized floor of this chapter. In summary, the main ribs were expected to flow between the supports, the secondary ribs were expected to be perpendicular to the main ones, and the main ribs were expected on the sides of the slab where the bending moment is the highest at midspan. The final floor design follows a very similar design with regard to these anticipations. It should be noted, however, that the reason that ribs are formed in the first place, is mostly because of the casting constraint. This forces the algorithm to find a solution similar to the floor slab system with isostatic ribs.

Differences with the floor designs by Nervi are also visible. First of all, he seems to have assumed that all the ribs have an equal height and width, while the topology optimization algorithm allows for variation therein. As anticipated, the algorithm then makes use of this freedom. A second difference is the absence of ribs in the minimum principal moment trajectories. Assuming they are present in Nervi's design because of stability reasons such as buckling, it makes sense they are not featured in the topology-optimized floor slab. The algorithm does not consider this failure mode and, even if it did, the bottom side of the floor is not under compression for this statically determinate design space, anyway.

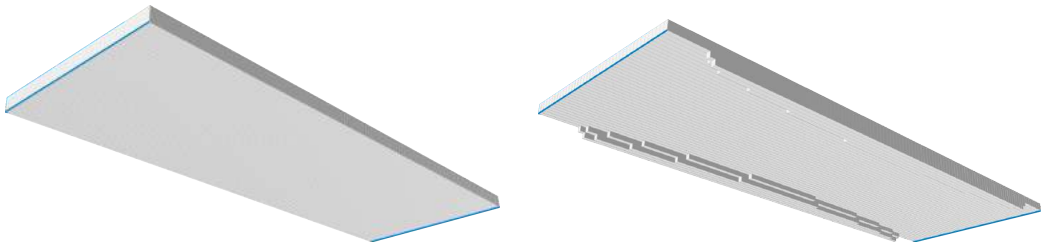
For this particular design, two ribs are formed. The theory of isostatics does not disclose how many there are expected to be. The topology optimization algorithm can help in that regard. While the number cannot be given explicitly, more ribs are anticipated by changing the following input parameters. When the design space becomes wider, the relatively thin upper slab requires more ribs to retain its overall stiffness. A smaller minimum member radius allows for more but thinner ribs while still complying with the set volume fraction. A higher volume fraction means that more material is allowed to fill the design space, so more volume means that either the existing ribs become larger, or that more ribs are formed. Which option of the two occurs, depends on the other input variables. Lastly, the thickness of the slab can be considered a variable that influences the number of ribs. A very thin slab will deform more than thicker slab. The resulting greater deformation will be corrected by the formation of more ribs.

# 7 Discussion

## 7.1 VERIFICATION OF THE HYPOTHESIS

In Chapter 1, several potential advantages of a topology-optimized floor slab were mentioned in the hypothesis. In this section, it is checked whether they were correct. The check is performed by comparing the topology-optimized floor to traditional floor slab systems. The floor that was designed in the previous chapter is assumed to be unprestressed and prefabricated. From the floor slabs discussed in Appendix A, the composite plank floor is the most similar system. Both the topology-optimized floor slab and the traditional, monolithic floor slab are displayed in Figure 7.1. The suggested advantages over a composite plank floor can be summarized as follows:

- Reduced weight;
- Reduced costs;
- Increased sustainability;
- Increased architectural value.



***(a)*** *Traditional, monolithic* ***(b)*** *Topology-optimized*

***Figure 7.1:*** *The two floor slabs that have been compared*

### 7.1.1 Weight

The Eurocode prescribes a maximum deflection of $0.004L$ in the Serviceability Limit State. For the given design space, that means that the deflection of the floor under the design load is allowed to be anything up to 28 mm. The composite plank floor is modeled in the topology optimization algorithm the same way the topology-optimized slab is modeled. The only difference is in the height and volume fraction quantities. The `volfrac` is set to 0.9999 so the solution is to fill the entire space, while the floor thickness `nely` (or $d_1$ in Figure 7.2) can be considered the variable, that needs to match the maximum allowed deflection $w_{max}$.

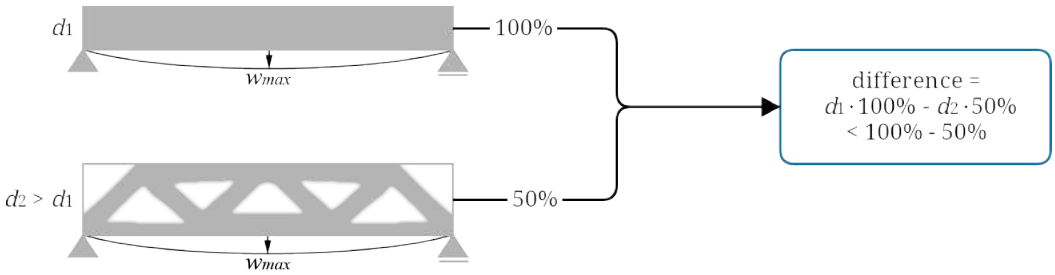



***Figure 7.2:*** *Comparing the weight of two hypothetical structures*

By applying this methodology to a composite plank floor, we arrive at a floor slab with a maximum deflection of 27 mm for a space that is 3 elements or $d_1 = 187.5$ mm high and has a volume fraction of 99.99%. The topology-optimized floor slab reaches a maximum deflection of 23 mm for a space that is $d_2 = 250$ mm high and has a volume fraction of 57.5%. These values are summarized in Table 7.1 .

The two floors can be compared fairly, now. The traditional floor slab has a total concrete volume of $0.999 \cdot (7.0 \cdot 2.5 \cdot 0.1875) = 3.28$ m$^3$, while the topology-optimized floor has a total volume of $0.575 \cdot (7.0 \cdot 2.5 \cdot 0.250) = 2.51$ m$^3$. It can be concluded that the topology-optimized floor slab is roughly 24% lighter than the traditional floor slab.

***Table 7.1:*** *Comparing the weight of a traditional floor slab and the topology-optimized floor slab*

| | Traditional | Topology-optimized | Unit |
|---|---|---|---|
| Allowed deflection | 28.0 | 28.0 | $mm$ |
| Measured deflection | 27.0 | 22.7 | $mm$ |
| Height | 187.5 | 250.0 | $mm$ |
| Span | 7000.0 | 7000.0 | $mm$ |
| Width | 2500.0 | 2500.0 | $mm$ |
| Optimization constraint | 100.0 | 57.5 | % |
| Volume | 3.3 | 2.5 | $m^3$ |

### 7.1.2 Costs

This section does not aim to give a quantitative price tag but instead, focuses on the aspects that can influence it because the vacuumatic formwork still mostly

remains in uncharted territory. The direct costs of reinforced concrete can be simplified as the sum of the concrete costs, the reinforcement costs, and the formwork costs (Appendix C). The concrete costs consist of three parts: material, labor, and transportation. In the previous section, it was found that the topology-optimized floor slab requires less concrete than a traditional slab so the material costs will be lower. Assuming that the distance between the factory and the building site are the same and that pouring the concrete requires the same amount of effort, then the topology-optimized floor slab offers an overall advantage in the concrete costs.

The reinforcement costs consist of material and labor costs. The topology-optimized floor slab, on average, might require more reinforcement steel. This means a higher material expense when compared to a traditional slab. The shape is also more complex, which results in more labor for bending and binding of the reinforcement. In the future, this difference might vanish, when workers get more familiar with the production method, or when robots take over reinforcement placement. Quist (2015) reports that one concrete factory in The Netherlands already operates with robots only. The complexity of the formwork will play a much smaller role when more factories are going to adopt this production method.

The formwork costs are composed of labor costs and the form efficiency. At the moment, the labor costs for the production of the formwork itself will be in favor of the traditional floor slab. The production of the vacuumatic formwork system requires a lot of work, especially when the method is still unknown to the workers. This was confirmed during the IASS-ISOFF 2015 workshop in Section 5.3. Similar to the reinforcement labor, however, when the production method becomes more familiar, the amount of labor reduces. Demoldability for the vacuumatic formwork is also easier and requires less labor. Lastly, the formwork efficiency plays a significant role in the costs. A vacuumatic formwork can be used over and over, whereas a timber or steel formwork has a limited number of repetitions.

Compared to traditional floor slabs, it is reasonable to assume that the topology-optimized floor slab is more expensive since the floor slab industry is greatly optimized for costs. However, in the near future, this difference might become smaller or it might even invert completely with the automation of the industry. Table 7.2 summarizes this section and displays how a vacuumatic formwork compares to a traditional timber or steel formwork, now and in the near future.

***Table 7.2:*** *Comparing reinforced concrete costs for a topology-optimized slab produced with vacuumatics and a traditional slab with timber/steel formwork*

| | Cost component | Presently | Prospective |
|---|---|---|---|
| Concrete | Material | + | + |
| | Transportation | 0 | 0 |
| | Labor | 0 | 0 |
| Reinforcement | Material | − | − |
| | Labor | − | 0 |
| Formwork | Labor | − | 0 |
| | Form efficiency | ++ | ++ |

### 7.1.3 Sustainability

The damage a product does to the environment can be expressed in the amount of carbon dioxide it produces in its creation and transport to the building site. There are several tools available to calculate the environmental impact of a reinforced concrete component. In this thesis, a calculator developed by Dijkhuis (2015) from ABT bv is utilized. Aspects such as sound insulation/reflection, waste, and reusability and adaptability of the vacuumatic formwork are not examined.

To calculate the carbon emission of a floor slab, the amount of concrete and reinforcement need to be determined. In Section 7.1.1, the volume of the floors was established to be 3.28 $m^3$ for a traditional floor and 2.51 $m^3$ for the topology optimized floor. An environmental class of XC1 is assumed according to Table 4.1 of NEN-EN 1992-1-1. A distance of 30 km is adopted for the transportation. The concrete mixture has a strength of roughly C50/60 for both floor slabs. The amount of reinforcement for the traditional slab is assumed to be 100 $kg/m^3$. An extra 20% of reinforcement is adopted for the topology-optimized floor slab.

The calculator sheet is included in Appendix D, which demonstrates the input and output of the comparison. It can be seen that the traditional slab produces1150 kg and the topology-optimized slab produces 950 kg of $CO_2$ per manufactured unit. The total reduction in emission per produced floor slab is equal to 17%. This is the equivalent of driving a car 1300 km or the $CO_2$ absorption of one hectare of trees over seven days. In conclusion, the topology-optimized floor slab design is more sustainable than its traditional counterpart. When taking the reusability and adaptability advantages of a vacuumatic formwork into account, this improvement will be even greater than 17%.

### 7.1.4 Aesthetics

The aesthetics of the topology-optimized floor slab cannot be measured quantitatively. It is therefore left up to the reader to conclude for him or herself whether the design as displayed in Figure 6.14 has more architectural value than a traditional floor slab.

## 7.2 LIMITATIONS, ADVANTAGES, AND DISADVANTAGES

### 7.2.1 Limitations

- Placing, bending, and binding reinforcement in the vacuumatic sand mold of the topology-optimized floor slab can be complicated, because of the complex shape and because of the fragility of the film containing the sand. The trend of the automation of the industry can be considered an opportunity, however. In the future, robots might be able to cut, bend, and binding the reinforcement for a competitive price. If the risk of damaging the film or formwork is still too great, then it would be possible to pour a thin layer of concrete, wait until it is hardened, then place the reinforcement, and finally pour the rest of the concrete layer. This would be in principle similar to the technique Nervi applied in his floors.
- If regular reinforcement bars prove to be too complicated to apply with the methods mentioned above, then fiber-reinforced concrete would not be able to offer a safe solution, at the moment. Research by Hermans (2011) shows ambivalent opportunities for the topology-optimized floor slab. Conclusions in favor of the

application are that the test-loaded floors failed at a distributed load more than twice as big as the design load, with big deflections and a well distributed cracking pattern. On the other hand, fibers are the most effective for big cross-sections and statically undetermined spans in order to make use of stress redistribution. The proposed floor slab design has neither. Additionally, there are no regulations available yet and the test results are scattered. This uncertainty makes it rather risky to only apply fibers as reinforcement to a slender topology-optimized floor slab.

### 7.2.2 Advantages of the topology-optimized floor slab

- It is roughly 24% lighter than a traditional, monolithic floor slab.
- It at least 17% more sustainable than a traditional, monolithic alternative.
- It has more architectural value than a traditional, monolithic alternative.
- Due to its low-tech nature and reusability, the costs of vacuumatic formwork are probably lower than other manufacturing methods for complex topologies such as CNC-milled polystyrene.
- The costs of a topology-optimized floor slab produced with vacuumatic formwork have the potential to be lower in the near future than that of a traditional, monolithic floor slab produced with timber formwork.

### 7.2.3 Disadvantages of the topology-optimized floor slab

- The reduction in weight also brings a reduction in sound insulation.
- Currently, the costs are probably still higher than that of a traditional floor slab.
- The higher structural efficiency of the present material can result in a lower fire resistance.

## 7.3 PROPOSED APPLICATIONS

### 7.3.1 Topology optimization algorithm

The topology optimization algorithm is open-source in the sense that it is open to anyone to use for free, but also, it is one hundred percent transparent. As the name already implies, the topology optimization algorithm is purely an algorithm to help design structures. The results should therefore not be seen as a final structural design. The algorithm may offer a solution to researchers or engineers that want to have an indication of a three-dimensional topology-optimized structure.

### 7.3.2 Topology-optimized floor slab

Regarding the topology-optimized floor slab, it is advised to consider the design when:

- The client is willing to pay extra for a floor that has more architectural value than traditional floor slab systems;
- The client is willing to pay extra for a floor that is more sustainable than the traditional alternatives;
- The rest of the structure benefits a lot from smaller detailing that can be achieved thanks to the reduction in weight from the floor slabs;
- The decreased sound insulation from the mass reduction can be overcome by insulation of the walls and/or the top side of the floors;

- The fire safety can be still guaranteed by, e.g., applying a sufficiently thick concrete/insulation cover (passive fire protection) and/or using a sprinkler system and fire alarms (active measures);
- A building has many different floor contours or supports and thus different design spaces, that maximize the advantages in adaptability of the vacuumatic formwork.

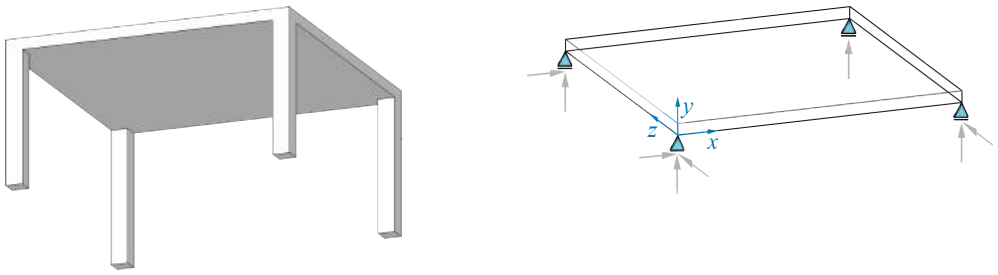
## 7.4 POTENTIAL OF THE METHOD

While the design from Section 6.5.4 performs very well compared to a traditional, monolithic floor slab, visually, it is not very different from a TT-slab. This section demonstrates the potential of the topology optimization methodology described in this thesis, by providing several examples that highlight its capabilities for different design problems.

### 7.4.1 Column-supported floor

By supporting the floor slab with four columns, rather than two beams, the optimization gives a more interesting result because the load is not transferred to the supports in straight, parallel lines, anymore. There are several examples that offer different solutions to roughly the same problem that is posed, here. Both the Gatti wool factory (Section 3.3.1) and the Palazzo del Lavoro (Section 3.3.2) show the solution according to isostatics. Key differences between isostatics and topology optimization are reported in Section 6.5.4.

Figure 7.3 displays how the design space is set up. The columns are modeled with rolling, hinged supports. The gray arrows show in which direction displacements are restricted. The floor is loaded by a distributed load and self-weight. The cracked Young's modulus is not introduced since previous optimizations show that it does not affect the found topology. With the input values from Table 7.3, the results in Figure 7.5a and Figure 7.6a are found. This design also inspired the front cover of this thesis.




***(a)** Context* ***(b)** Model*

***Figure 7.3:** Design space of the floor with four column supports*

***Table 7.3:** Input variables for the column-supported floor*

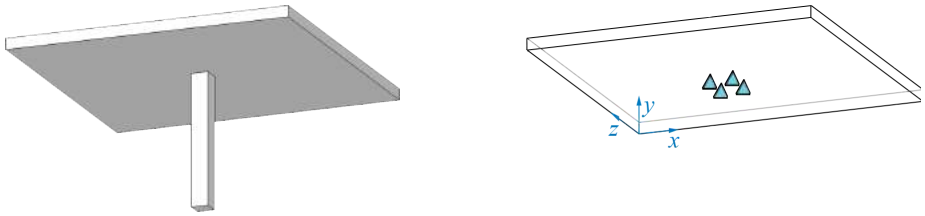
| Input variable | Value |
|---|---|
| `nelx` | 80 |
| `nely` | 3 |

***Table 7.3:*** *Input variables for the column-supported floor*

| Input variable | Value |
|---|---|
| `nelz` | 80 |
| `volfrac` | 0.57 |
| `penal` | 3.0 |
| `rmin` | 1.5 |
| `activetop` | on |
| `selfweight` | on |
| `castingconstraint` | on |

### 7.4.2 Toadstool

In this thesis, the design space that consists of a slab supported by one column in the center is called the toadstool. Similar to the column-supported floor slab, the load is not transferred to the support in straight, parallel lines, anymore. Figure 7.4 provides the context for this optimization problem. The four neighboring nodes that constitute the boundary condition from the head of the column have their degrees of freedom restricted in all three directions. The folly at sanatorium Zonnestraal by ABT bv & Hurks (2005) in Appendix B provides a similar but fundamentally different design. Bak (2011) has researched the same design space for a structure that is suitable for manufacturing with fabric formwork. With the input values from Table 7.4, the results in Figure 7.5b and Figure 7.6b are found.




***(a)*** *Context* ***(b)*** *Model*

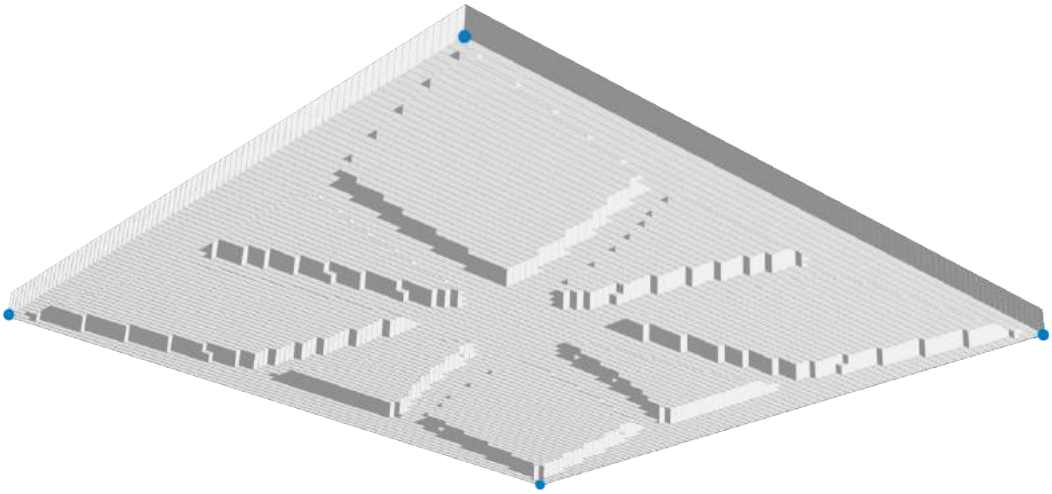
***Figure 7.4:*** *Design space of the toadstool*

***Table 7.4:*** *Input variables for the column-supported floor*

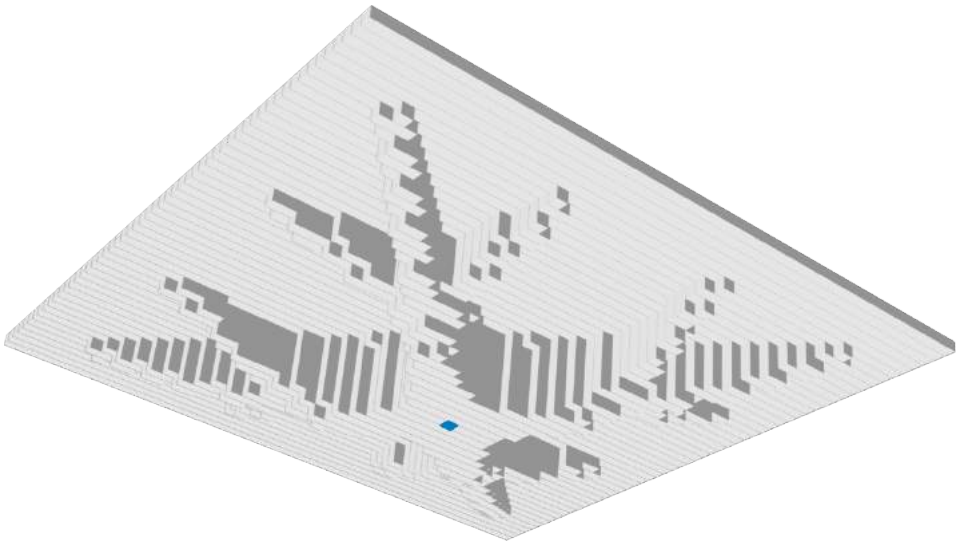
| Input variable | Value |
|---|---|
| `nelx` | 55 |
| `nely` | 8 |
| `nelz` | 55 |
| `volfrac` | 0.225 |
| `penal` | 3.0 |
| `rmin` | 1.5 |
| `activetop` | on |
| `selfweight` | on |

***Table 7.4:*** *Input variables for the column-supported floor*

| Input variable | Value |
|---|---|
| `castingconstraint` | on |



***(a)*** *Column-supported floor*



***(b)*** *Toadstool*

***Figure 7.5:*** *Results from the topology optimizations*

*(a) Column-supported floor*

*(b) Toadstool*

***Figure 7.6:*** *Interpretations of the topology optimizations*

# 8 Conclusions

## 8.1 ANSWER TO THE RESEARCH QUESTION

In Chapter 1, the research question was stated as follows:

*How can the topology of a concrete floor slab be optimized when taking manufacturability constraints from a vacuumatic formwork into account, and how would it compare to a traditional, monolithic floor slab in terms of weight, costs, sustainability, and appearance?*

The research question can now be answered. For the first part of the question, it was found that the traditional, fictive topology optimization material model does not translate too well to a reinforced concrete material model. In order to arrive at a material that more closely resembles concrete, several changes have been made to the topology optimization process. The Poisson's ratio, the Young's modulus for compressed elements, and the Young's modulus for cracked concrete have been altered or added. With regard to manufacturability, the production method of a vacuum stabilized sand mold brings constraints in the maximum curvature, the minimum concrete thickness, and the size of the formwork. The maximum curvature is not introduced in the design process because it is assumed the film is elastic enough to completely follow the pattern shape. A minimum concrete thickness is applied. Additionally, the size of the formwork is restricted to 2.5 m by 7.0 m. This size also allows the prefabricated floor panels to be transported to the building site by truck. It is assumed that only a one-sided mold is allowed, therefore voids are not possible and a casting constraint has also been applied. Lastly, the floor slab needs to have a flat top surface to allow people to walk on it. By constraining the solution space of the topology optimization, the iterative design process is influenced by both manufacturability and functionality.

This methodology has been applied to arrive at a preliminary design for a topology-optimized floor slab, answering the second part of the research question. It can be concluded that the research objective has been accomplished, because this design, which was made with the developed topology optimization algorithm in Python, consists of a floor slab system that can be produced with a vacuumatic formwork system. The advantages of the design compared to a traditional, monolithic slab include a weight reduction of 24%, the potential for a cost decrease, a $CO_2$ emission reduction of at least 17%, and increased architectural value.

## 8.2 GENERAL CONCLUSIONS

- By combining topology optimization with vacuumatics, the strengths of topology optimization are enhanced, while its weaknesses are scaled down. The costs of the very complex structures that follow from traditional topology optimization can be reduced by utilizing a vacuumatic formwork. Both vacuumatics and topology optimization can also be applied to other structural components such as walls, columns, or beams.
- In terms of being practical, topology optimization is already a step up from the highly theoretical Michell structures. Despite still not being completely functional for reinforced concrete structures, it is considered a valuable tool in designing structures with an efficient material distribution. For now, it is concluded that the bottleneck in the process is not caused by the manufacturing method, but by the engineering, which is the opposite of what usually occurs in the building industry.
- Nervi's ribbed floor slabs with isostatic ribs were and still are an effective method for designing floors with a reduced material usage. The similarities between the topology-optimized floor slab and his ribbed floor slab system are a testament to that.
- It is possible to arrive at an optimal design by applying additional manufacturability and functional constraints to the topology optimization process. However, with each additional feature, the optimum becomes harder to find. The process then requires more iterations, especially when the constraints are applied in a series rather than all at once. Additionally, the objective function shows erratic behavior for the features that change the conditions each iteration, such as the self-weight and concrete modeling.
- The addition of a reduced Young's modulus for tensile elements does not result in a significantly different topology but does give a more realistic deflection for each iteration. It is concluded that for concrete structures the quicker, standard topology optimization algorithm is sufficient, provided that the exact deformations are not of importance.
- In practice, topology optimizations of concrete floor slabs will have the most added value when applied to very complex problems. The final floor slab from Section 6.5.4 and both examples from Section 7.4, are not puzzling enough that they cannot be predicted by structural engineers. The design space of the former zoology lecture hall at Freiburg University from Section 3.3.4, on the other hand, poses a more challenging problem where topology optimization has significant advantages over traditional design methods.

# 9 Recommendations

## 9.1 TOPOLOGY OPTIMIZATION ALGORITHM

- In the Finite Element Analysis part of each iteration, the function $KU = F$ is solved for $U$. At the moment, the global stiffness matrix $K$ is sparse, which is already a step up from the slower, dense alternative. The function is solved in a direct manner with `spsolve`, which is considered very quick for small design spaces, and quick enough for larger ones. However, for even larger design spaces, or for denser meshes, the algorithm crashes when solving the function. Adding the following options to the algorithm might speed up the optimization process and also not crash the algorithm when a denser mesh is applied.
  - In the `scikit.umfpack` module, the same function `spsolve` can be found, that might be quicker than the one from the `scipy.sparse.linalg` module.
  - There are iterative solvers available in the `scipy.sparse.linalg` module that approximate the exact solution but might take less time to do so.
  - Utilizing the GPU rather than the CPU with, for example, CUDA might be quicker. However, since the speed is mostly dependent on one line in the algorithm, parallel computing might not offer a satisfying solution.
- In contrast with many other topology optimization algorithms, the one presented in this thesis is open-source. This provides opportunities in cloud computing to speed up the optimization process. This would require the line in which the global stiffness matrix is inverted to be distributed over multiple computers, however, which calls for further research.
- At the moment, MayaVi models all the 'full' elements as cubes, so elements on the inside of the structure (that are not visible) are also drawn. This can make the visualization slow for dense meshes. The speed of the visualization of the found optimum in the `displayfigure` definition can be improved by only displaying the outside surfaces or cubes of the result.

- The result that follows from the topology optimization algorithm can help with making a design. It is strictly a tool to come up with designs and the result cannot be viewed as definitive. There is still a step to be made in post-processing in order to arrive at a structural design. Among other things, part of this post-processing would include modeling of reinforcement and smoothing of the display.

## 9.2 TOPOLOGY OPTIMIZATION AND MANUFACTURABILITY

- The subject of optimization is very broad and as such, there are still many topics not yet researched. In the area of topology optimization guided by manufacturability, there are still many possibilities related to the material or production method that is chosen. For example, the field of additive manufacturing shows a lot of promise. The manufacturing method allows for placing material only where it is wanted, which is exactly what topology optimization can provide. Currently, however, more research is still required for both concrete and steel printing.
- Topology optimization can also be used to optimize steel truss structures. The focus for manufacturability then moves more towards the reduction of connection complexity and number of connections, among other things. The so-called growing ground structure method is a suitable method for truss optimization. It does not include manufacturability in the design process, however, so progress can be made there.
- The methodology that is applied in this thesis can also be applied to structures other than concrete floors. Concrete walls, bridges, columns, etc. are all possibilities for further research.
- Since costs are an important factor in choosing a structural system or design, adding a cost function to the optimization process would be a welcome addition. This would result in a different or an extra optimization objective. For regular manufacturing methods, there are key figures available to relate a design to their costs. For a vacuumatic formwork, these do not exist yet. Therefore, this recommendation for further research consists of two parts: research key figures for a vacuumatic formwork first, and add a cost objective to the topology optimization process second.

## 9.3 TOPOLOGY OPTIMIZATION AND MANUFACTURABILITY OF A CONCRETE FLOOR

- For topology-optimized floor designs, the addition of multiple materials for the modeling of reinforcement would be valuable. The mesh is still too rough to accurately model an extra material, however. One possible simplification of reality would be to model the multiple reinforcement bars as one larger equivalent reinforcement cylinder.
- If larger spans are required, then the inclusion of pre-stress would have its advantages. In the topology optimization algorithm, this could be modeled as two point loads where the cables are applied (as long as the tendons are laid out in a straight line). This addition would also cause new challenging manufacturability problems.

- For floors, more combinations of boundary conditions and design spaces can be researched. Both topology optimization and the vacuumatic formwork system can be applied to complex problems, so very specific flooring situations can also be analyzed should a project call for that.
- Lastly, during the ISOFF-IASS workshop, it was found that the plastic film was not elastic enough to accurately follow the shape of the pattern. A solution on the manufacturing side of the problem would be to either heat the film more or to use a more elastic film type. The solution can also be looked for on the design side of the problem: the application of a curvature constraint can limit the curvature of the surface that is allowed to be found.

# 10 Bibliography


3ders (2014). 3ders.org - 10 completely 3D printed houses appear in Shanghai, built under a day. http://www.3ders.org/articles/20140401-10-completely-3d-printed-houses-appears-in-shanghai-built-in-a-day.html [Online; accessed 24-June-2015].

Abspoel, R., Pasterkamp, S., de Vries, P., and Terwel, K. (2012). Dictaat constructief ontwerpen. Technical report, Delft University of Technology.

ABT bv (2014). Ontwerpoptimalisatie – effectiviteit van optimalisatie-typen. Internal document ABT bv.

ABT bv & Hurks (2005). Folly at sanatorium Zonnestraal, Hilversum. http://www.uhsb.nl/?p=150. [Online; accessed 1-November-2015].

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B., and Sigmund, O. (2011). Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16.

Anink, D., Van Ewijk, H., Schuurmans, A., Nieman, H., Van Luijk, P., and Levels-Vermeer, J. (2015). Bepaling van de milieuprestaties van gebouwen en gww-werken. Technical report, SBRCURnet in cooperation with Stichting Bouwkwaliteit. Commissioned by Ministerie van BZK/WB.

Antony, F., Griesshammer, R., Speck, T., and Speck, O. (2014). Sustainability assessment of a lightweight biomimetic ceiling structure. *Bioinspiration and biomimetics*, (9).

Arch Daily (2011). AD classics: Expo '58 + Philips pavilion / Le Corbusier and Iannis Xenakis. http://www.archdaily.com/157658/ad-classics-expo-58-philips-pavilion-le-corbusier-and-iannis-xenakis. [Online; accessed 27-January-2016].

Bak, A. (2011). Interactive formfinding for optimised fabric-cast concrete. Master's thesis, University of Bath.

Bendsøe, M. (1989). Optimal shape design as a material distribution problem. *Structural Optimization*, 1(4):193–202.

Bendsøe, M. (1995). *Optimization of structural topology shape and material*. Springer.

Bendsøe, M. and Sigmund, O. (2003). *Topology optimization: theory, methods, and applications*. Springer.

Blaauwendraad, J. (2006). Plate analysis, theory and application: Volume 1 theory. Technical report, Delft University of Technology.

Blaauwendraad, J. (2010). *Plates and FEM*. Springer.

Boon, P., Van der Klauw, M., and van Zeijl, E. (2014). Milieuprestatieberekening gebouwen – evaluatie 2013. Technical report, Movares.

Bourdin, B. (2001). Filters in topology optimization. *International journal for numerical methods in engineering*, (50):2143–2158.

BroekBakema (2014). Communicatiezuil a2 broekbakema, architectuur, interieur en masterplanning. `http://www.broekbakema.nl/alleprojecten/dsm-reclamemast/`. [Online; accessed 19-May-2015].

Bruns, T. E. and Tortorelli, D. A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, (190).

Buntrock, D. (2011). Teshima Art Museum by Ryue Nishizawa, Teshima island, Japan. *The Architectural Review*.

Burns, S. (2002). *Recent Advances in Optimal Structural Design*. ASCE.

Burry, J. and Burry, M. (2010). *The New Mathematics of Architecture*. Thames and Hudson.

Chen, Y. and Li, Y. (2010). Beam structure optimization for additive manufacturing based on principal stress lines. *Solid Freeform Fabrication Proceedings*.

Chilton, J. (2000). *An Engineers Contribution to Contemporary Architecture – Heinz Isler*. Telford.

Chiorino, M. (2012). Art and science of building in concrete: The work of pier luigi nervi. *Concrete International*, (34):32–44.

Clark, T. (n.d.). Mccann's casting process and information guide. `http://www.mccannsales.com/book/vprocess.pdf`. [Online; accessed 25-January-2016].

Culmann, K. (1875). *Die graphische Statik*. Meyer and Zeller.

Del Pico, W. (2013). *Project Control : Integrating Cost and Schedule in Construction*. Wiley.

Dijkhuis, L. (2015). ABT ontwerptool groen beton. Internal document ABT bv.

Dombernowsky, P. and Sondergaard, A. (2010). Danish team uses HyperWorks to prove the value of topology optimization for concrete architectural structures. Technical report, Altair Engineering.

Dubbeldam, J. W. (1991). *Handboek Bekistingen*. Studievereniging uitvoering betonconstructies, STUBECO.

Duyster, H. (1958). *Het Philipspaviljoen op de Wereldtentoonstelling te Brussel 1958*. Philips.

EDC (2015). Engineering design centre: Structural systems optimization for the building industry. `https://www-edc.eng.cam.ac.uk/projects/buildingindustry/`. [Online; accessed 13-May-2015].

Fanella, D. and Alsamsam, I. (2007). Design of concrete floor systems. *Cement*.

Flint, M. (2008). Ant system based structural design of a roof in ultra-high performance concrete. Master's thesis, Delft University of Technology.

Gargiani, R. (2012). *L'architrave, le plancher, la plate-forme – Nouvelle histoire de la construction*. Presses polytechniques et universitaires romandes.

Georgopoulos, C. and Minson, A. (2014). *Sustainable Concrete Solutions*. Wiley.

Halpern, A., Billington, D., and Adriaenssens, S. (2013). The ribbed floor slab systems of Pier Luigi Nervi. In *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2013*.

Henket, M. (2013). Construeren met minder materiaal. *Cobouw*.

Hermans, M. H. (2011). Staalvezelbeton toegepast in woningbouwcasco. Master's thesis, Eindhoven University of Technology.

Huang, X. and Xie, Y. M. (2010). *Evolutionary Topology Optimization of Continuum Structures: Methods and Applications*. Wiley.

Huijben, F. (2014). *Vacuumatics: 3D formwork systems*. PhD thesis, Eindhoven University of Technology.

Huijben, F. (2015). Vacuum-gestabiliseerde zandmallen. *Tektoniek*.

Hunter, W. (2009). Predominantly solid-void three-dimensional topology optimisation using open source software. Master's thesis, Stellenbosch University.

Iori, T. and Poretti, S. (2005). Pier Luigi Nervis Works for the 1960 Rome Olympics. In *Actas del Cuarto Congreso Nacional de Historia de la Construccin*.

ISOFF (n.d.). Types of fabric framework. `http://www.fabricforming.org/TECHNICAL/TYPES.html`. [Online; accessed 2-November-2015].

Iyengar, N. (2004). Optimization in structural design. Technical report, IIT Kanpur.

Kumar, P. and Gaindhar, J. L. (1995). Off-line quality control for v-process castings. *Quality And Reliability Engineering International*, 11(3):175–181.

Levy, M. (1899). Sur l'equilibre elastique d'une plaque rectangulaire. *Comptes Rendus*, (129):535–539.

Liu, K. and Tovar, A. (2014). An efficient 3D topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, 50(6):1175–1196.

Living Projects (2014). Livingprojects Creative LED lighting — Nieuws — Van communicatiezuil tot lichtsculptuur - Creative LED Lighting. `http://livingprojects.nl/nl/nieuws/van-communicatiezuil-tot-lichtsculptuur`. [Online; accessed 19-May-2015].

mbX (2015). OV Terminal – Arnhem – mbX. `http://www.mbx.nl/project/ovt-arnhem/`. [Online; accessed 4-November-2015].

Merriam-Webster (2015). Optimization — Definition of optimization by Merriam-Webster. `http://www.merriam-webster.com/dictionary/optimization`. [Online; accessed 15-October-2015].

Meyer, H. (1867). Die Architectur der Spongiosa. *Wissenschaften im Medicin*.

Michell, A. G. M. (1904). The limits of economy of material in frame-structures. *Philosophical Magazine*, (6):589–597.

Nadai, A. (1915). Die Formanderungen und die Spannungen von rechteckigen elastischen Platten. *Forschungsarbeiten auf dem Gebiete des Ingenieurwesens*.

Nakata, K. and Kubo, Y. (1974). Molding method with shielding member drawn against particulate material.

Nationaal Militair Museum (2015). Homepage - Nationaal Militair Museum. `https://www.nmm.nl/`. [Online; accessed 19-May-2015].

Nedcam (2008). Spencer Dock Bridge Dublin - Nedcam. `http://www.nedcam.com/spencer-dock-bridge-dublin.htm`. [Online; accessed 2-November-2015].

Nervi, P. L. (1963). *New Structures*. Architectural Press.

Nervi, P. L. (1965). *Aesthetics and Technology in Building*. Harvard University Press.

Ohmori, H., Futai, H., Iijima, T., Mutoh, A., and Hasegwa, Y. (2005). Computational morphogenesis and its application to structural design. In *Proceedings of International Symposium on Shell and Spatial Structures*.

OIII architecten (2012). Faculteit CiTG TU Delft – OIII Architecten. `http://www.o-drie.nl/portfolio/citg/`. [Online; accessed 28-October-2015].

Orlando, A. and Luege, M. (2014). Topology optimization. Presentation.

PERI (2005). Mercedes-Benz Museum. `http://www.peri.com/en/projects/cultural-buildings/mercedes-benz-museum.html`. [Online; accessed 1-November-2015].

Peurifoy, R. and Oberlender, G. (1996). *Formwork for Concrete Structures*. McGraw-Hill.

Piano, R. (1969). Progettazione sperimentale per strutture a guscio. *Casabella*.

Powell, K. (2011). *The Great Builders*. Thames & Hudson.

Quist, J. (2015). Bim-bestand stuurt robot in betonfabriek aan. *Cobouw*.

Ravi, B. (2005). *Metal casting: Computer-Aided Design and Analysis*. Prentice-Hall of India.

Rijksoverheid (2015). Milieuprestaties van gebouwen meten — duurzaam bouwen en verbouwen — rijksoverheid.nl. `http://www.rijksoverheid.nl/onderwerpen/duurzaam-bouwen-en-verbouwen/duurzaam-bouwen/milieuprestaties-van-gebouwen-meten`. [Online; accessed 7-August-2015].

Sasaki, M. (2008). Morphogenesis of flux structure. In *From control to design: parametric/algorithmic architecture*. Actar.

Schipper, H. R. (2015). *Double-curved precast concrete elements – Research into technical viability of the flexible mould method*. PhD thesis, Delft University of Technology.

Sharp, D. (2002). *Twentieth-Century Architecture: a Visual History*. Images Publishing.

Shengping, Y., Lei, W., and Dehan, L. (2008). The EVA plastic film for V-process and steel castings by V-process in china. *China Foundry*, 5(2):77–81.

Sigmund, O. (2001). A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(2):120–127.

Sijpkes, P. (n.d.). McGill University – sand box. `http://www.arch.mcgill.ca/prof/sijpkes/Downloads/SandBox.jpg`. [Online; accessed 27-January-2016].

Skedros, J. and Baucom, S. (2007). Mathematical analysis of trabecular trajectories in apparent trajectoral structures: The unfortunate historical emphasis on the human proximal femur. *Journal of Theoretical Biology*, (244):15–45.

Spuybroek, L. (2004). *NOX: Machining Architecture*. Thames and Hudson.

Structurae (2005). Millau viaduct toll station. `http://structurae.net/structures/millau-viaduct-toll-gate/photos?min=0`. [Online; accessed 1-November-2015].

Studio RAP (2015). 3TU Prototype - Studio RAP. `http://studiorap.nl/projects/3tu-concrete-detail/`. [Online; accessed 26-October-2015].

Tam, K. M., Coleman, J. R., Fine, N. W., and Mueller, C. T. (2015). Stress line additive manufacturing (slam) for 2.5-d shells. In *Proceedings of International Symposium on Shell and Spatial Structures*.

Timoshenko, S. (1956). *Strength of materials: part 2, 3rd edition*. Van Nostrand.

Timoshenko, S. and Woinowsky-Krieger, S. (1959). *Theory of plates and shells*. McGraw-Hill.

Uponor (2015). Detailed reference view layer - system - uponor. http://www.uponor.nl/site-data/layer/system/detailed-reference-view-layer.aspx?id=B17B99C6-4B21-47B4-850E-C0E5BF91C4A7. [Online; accessed 19-May-2015].

Van de Brug, E., de Boer, S., and Grasmayer, H. (2000). Polyester bekisting voor bruggen leidschenveen. *Cement*, 3:42–45.

Van der Horst, A. (2013). Construction technology of civil engineering projects. Technical report, Delft University of Technology.

Van der Ploeg, C. (2013). A knowledge-based framework for structural optimisation. Master's thesis, Delft University of Technology.

Van Velden, F. (2015). Software bepaalt vorm van betonconstructie. *Cobouw*.

Vollers, K. J. and Rietbergen, D. (2009). Vloeiende vormen fabriceren. *Gevelbouw*, 6(3):30–33.

Wang, W., Stoll, H., and Conley, J. (2010). *Rapid tooling guidelines for sand casting*. Springer.

wikimapia (n.d.). Qatar national convention centre - education city. http://wikimapia.org/14080307/Qatar-National-Convention-Centre. [Online; accessed 24-June-2015].

Xie, Y. M. and Steven, G. P. (1993). A simple evolutionary procedure for structural optimization. *Computers and Structures*, 49(5):885–896.

Xie, Y. M. and Steven, G. P. (1997). *Evolutionary structural optimization*. Springer.

Yale Bone Lab (n.d.). Bone lab – systems cell biology. http://medcell.med.yale.edu/systems_cell_biology/bone_lab.php. [Online; accessed 17-July-2016].

# A

# Traditional concrete floor systems

Concrete flooring systems can be divided into cast in-situ floors and precast floors. A concrete floor, where the liquid concrete mixture is poured and hardens on the building site, is called in-situ concrete. Precast floors are manufactured in a factory and assembled on site. A combination of these methods is where part of the floor structure is precast to function as a permanent formwork onto which a concrete topping is poured. Figure A.1 displays these three different types and some corresponding floor slab systems. In a multi-story building, the concrete floor slabs can take up 85% of the total structural weight (Georgopoulos and Minson, 2014). Therefore, the choice in flooring system can have major consequences and reduction of material use offers interesting opportunities for both weight reduction and sustainability.

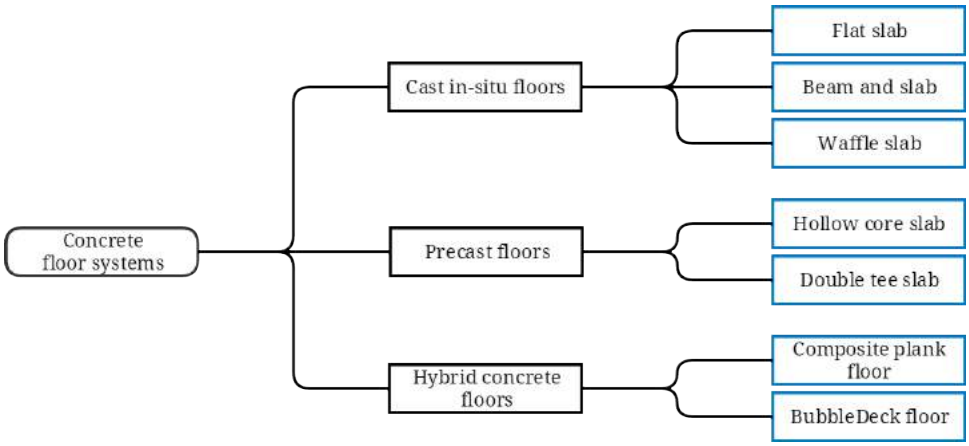



***Figure A.1:*** *Concrete floor systems divided by manufacturing method*

## A.1 CAST IN-SITU FLOORS

### A.1.1 Flat slab

The flat slab (Figure A.2) scores the lowest of all floor systems in terms of material efficiency since the dead load is rather high. Deflections and punching shear at the column locations are usually the factors limiting the small span or slab thickness. A thick, solid slab does have other advantages, however, such as a flexible column grid. Since it is also easy and quick to build it is very popular. Therefore, when considering current manufacturing technologies, it could be said that from a manufacturing point of view this is an optimal structure.

Lightweight table forms allow for large areas to be poured at the same time. Repetitive use of the forms results in the quickest floor erection, resulting in an average build speed of 500 $m^2$/week/crane, according to Georgopoulos and Minson (2014).

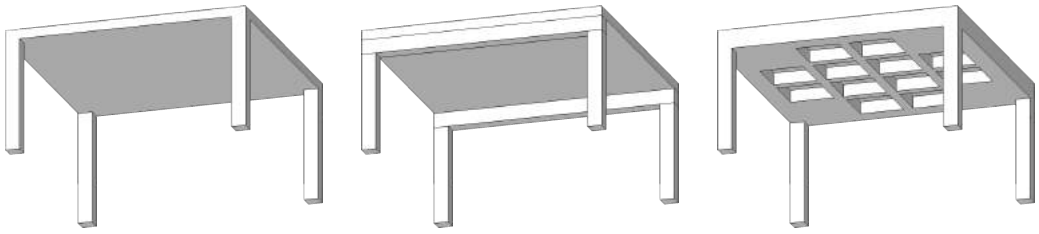
### A.1.2 Beam and slab

This type of floor construction consists of a slab, which is positioned on top of beams (Figure A.2). This addition to the flat slab system allows for medium spans, an irregular grid, and/or bigger loads. Construction time is longer, however, as it takes more effort to make the formwork and fix the reinforcement in the beams.

### A.1.3 Waffle slab

The waffle slab (Figure A.2) aims to reduce the material efficiency of the flat slab by removing concrete in the areas where it is used less. The result is a lighter and stiffer slab, but with a greater thickness. This means that it is able to carry heavier loads or span longer distances than the flat slab. Disadvantages include a stricter column grid as a square grid is easier to produce. Near the columns, the waffle shape changes to a solid slab to resist the shear stress, hogging moment, and punching shear.

Just like the flat slab, the waffle slab is constructed using table forms, except that there are additional molds added to the process on top of the forms. The construction time is longer than the flat slab as a result.



***(a)*** *Flat slab* ***(b)*** *Beam and slab* ***(c)*** *Waffle slab*

***Figure A.2:*** *Cast in-situ floor systems supported by columns*

## A.2 PRECAST FLOORS

As mentioned in the introduction of Appendix A, precast floors are manufactured in a factory and assembled on site. This brings several advantages. Scaffolding is not necessary anymore as the elements are already hardened during assembly. The construction time is shorter. The soffit has a smooth finish, so post-processing is not absolutely necessary. Longer spans are possible since these elements are of a higher concrete quality and also have the option of being prestressed. The precast floors mentioned below are visualized in Figure A.3.

### A.2.1 Hollow core slab

Hollow core slabs are prestressed units with round, square, or oval longitudinal cores in the center in order to save weight. The units are usually manufactured using the extrusion or slip forming process where either the mold or the extruding machine pushes itself forward, leaving a hollow core slab behind. Options for sawing off a slab are limited because of the prestressing tendons. Building speed is quick when applying a concrete topping for structural integrity because the topping is relatively thin. When not applying any topping at all the flooring system is even quicker.

### A.2.2 Double tee slab

Also known as a TT-slab, this floor system is prestressed but allows for larger spans than the hollow core slab. The price is higher, however. Each element consists of two T-shaped ribs on the bottom while the top is flat. The possibility of adding a concrete topping exists, but the building speed is even quicker without this layer.

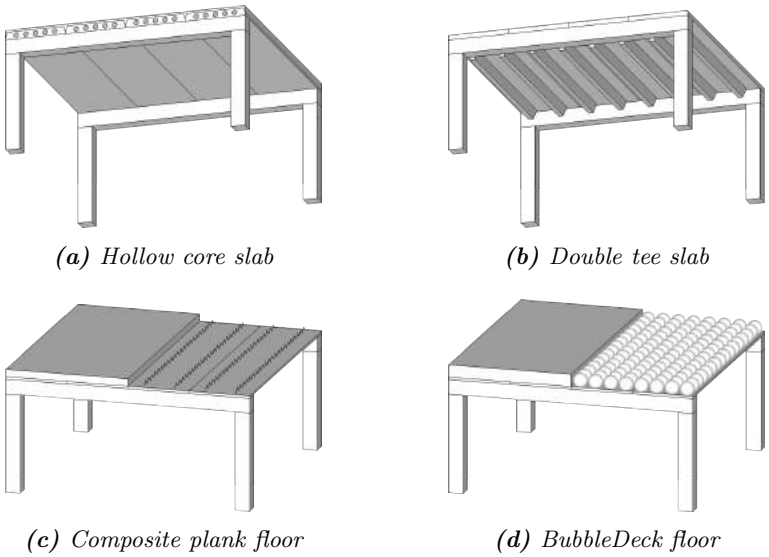
## A.3 HYBRID CONCRETE FLOORS

### A.3.1 Composite plank floor

This is a hybrid concrete construction, where the advantages of a precast and in-situ cast floor are combined. Since the quality is high and the construction time relatively quick it is also very popular. Precast concrete elements act as a permanent mold on the bottom side and concrete is cast on top. The precast mold can be either prestressed or reinforced in two directions. A lattice girder in the element ensures a good connection with the cast concrete and facilitates hoisting of the element on site.

### A.3.2 BubbleDeck floor

Precast concrete elements are used as a permanent mold. Plastic balls are then placed on top and fixed using reinforcement before casting concrete in-situ. This saves approximately 35% (Abspoel et al., 2012) of weight compared to a flat slab of the same thickness. Near the supports, the balls are not used in order to increase the shear stress and hogging moment capacity. Construction time is average, because even though the mold is permanent and quick to place, the thick in-situ concrete layer has a relatively long setting time.



**(a)** Hollow core slab

**(b)** Double tee slab

**(c)** Composite plank floor

**(d)** BubbleDeck floor

***Figure A.3:*** *Precast and hybrid floor systems supported by columns and beams*

# B

# Topology optimization and concrete production methods

There are many different techniques available for the production of concrete structural elements; some of these are impractical for the application of this research, however. The goal of this part of the research is organizing the manufacturing techniques viable for the production of concrete topology-optimized structures. From this list of methods, the manufacturability constraints can be researched. Supported by reference projects, a selection has been made based on which production methods are able to manufacture freely-curved concrete structures. Aspects such as surface finish and production speed are also of importance, but not governing. These methods include:

- Timber formwork
- Steel formwork
- Reconfigurable mold
- Ferrocement formwork
- Polyester formwork
- CNC milled EPS formwork
- Fabric forming
- Vacuum stabilized formwork
- Additive manufacturing

## B.1 PRODUCTION METHODS

### B.1.1 Timber formwork

The traditional timber formwork is commonly used as almost any shape can be constructed. Combined with a good surface protection one mold can be used several times. While timber molds are often used to manufacture elements with orthogonal angles and flat surfaces, more complex double curved shapes are also possible.

Adaptability for these complex molds is not optimal, however, so for structures without repetition a different type of formwork should be considered. Two methods are examined for the construction of topology-optimized structures. The first method is displayed in Figure B.1 and uses CNC (Computer Numerical Control) milled timber ribs for the supporting structure. On top of the ribs, a thin wooden layer forms the contact surface. A thinner contact layer allows for more curvature. Displayed in Figure B.2, the second method is based on the waffle-slab, but instead of using cubic forms to shape the space between the ribs more complex forms are used. The folly at sanatorium Zonnestraal is an example where the ribs are curved in a single direction. With the use of UHPC, a very slender canopy has been realized.

***(a)*** *Timber ribs* ***(b)*** *Timber ribs with plated contact surface* ***(c)*** *After demolding*

***Figure B.1:*** *Mercedes-Benz museum, Stuttgart (PERI, 2005)*

***(a)*** *Timber formwork* ***(b)*** *Close-up* ***(c)*** *Finished structure*

***Figure B.2:*** *Folly at sanatorium Zonnestraal, Hilversum (ABT bv & Hurks, 2005)*

### B.1.2 Steel formwork

Similar to the timber formwork, a steel mold consists of steel ribs for the supporting structure and a steel plate for the contact surface. Making a steel mold with single curvature is relatively simple with the use of rollers. Adding a second curvature can be troublesome, however. Even with a thin plate, only low curvatures are generally possible. The reference project in Figure B.3 demonstrates how the second curvature is rather low. An advantage over the timber mold is the high reusability: they can be reused more often which is useful for structures with a lot of repetition.

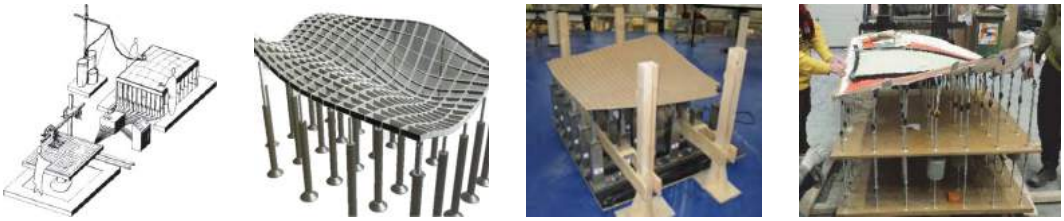
### B.1.3 Reconfigurable mold

The reconfigurable mold can be seen as a solution to the problem of low adaptability. The adjustable formwork consists of two elements: a flexible contact surface on top of a pin bed where each pin's height can be modified. Famous architect Renzo Piano was one of the first to build such a mold in 1966. Since then many researchers have

*(a) Steel formwork* *(b) Steel formwork and several finished elements* *(c) Finished structure*

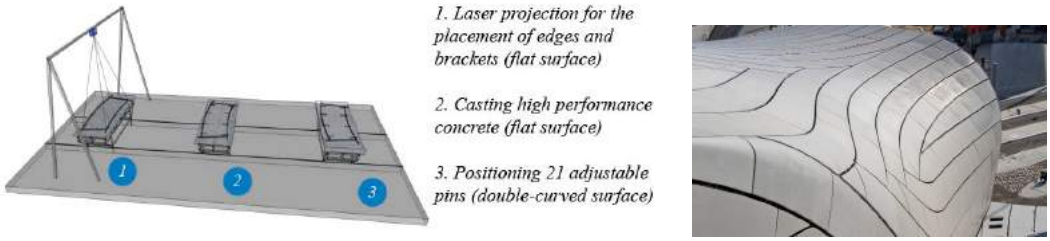
***Figure B.3:*** *Millau viaduct toll gate (Structurae, 2005)*

come up with improvements, several of which are displayed in Figure B.4. These concepts allow for molds to be used many times over and they are also able to produce double curved elements. There are limitations on the radius of the curves, however. Construction firm mbX has successfully applied this method in the fabrication of the façade of the Arnhem public transport terminal, as shown in Figure B.5.



*(a) Piano (1969)* *(b) Spuybroek (2004)* *(c) Vollers and Rietbergen (2009)* *(d) Schipper (2015)*

***Figure B.4:*** *Reconfigurable mold concepts*




*(a) Applied method* *(b) Close-up of the elements*

***Figure B.5:*** *Public transport terminal, Arnhem (mbX, 2015)*

### B.1.4 Ferrocement formwork

Ferrocement is the material used by Pier Luigi Nervi for the construction of the Gatti wool factory and Palazzetto dello Sport, among others. Producing the organic shapes is a challenge that Nervi solved by making use of ferrocement. It was used to make prefabricated concrete elements in plaster casts. Regular reinforcement was placed inside the ferrocement elements acting as a permanent mold and afterward they were covered in concrete creating one single slab.

*(a) Formwork production* *(b) Finished structure* *(c) Formwork production* *(d) Finished structure*

***Figure B.6:*** *Gatti wool factory and Palazzetto dello Sport (Nervi, 1965)*

### B.1.5 Polyester formwork

Fiber-reinforced polyester formwork refers to the material of the contact surface. The shape of the polyester formwork is produced with the help of a counter mold or pattern. The pattern is made with any of the other mentioned production methods, after which the polyester can be applied in layers by hand or by spraying. The amount of shapes possible with this type of mold is immense but generally restricted by the production method of the pattern. The supporting structure can be made with the any of the other production methods. The reusability is excellent with this type of mold: 75 beams in three variants have been realized in Leidschenveen, The Netherlands (Figure B.7).

*(a) Polyester formwork* *(b) One fabricated element* *(c) Finished structure*

***Figure B.7:*** *Ribcage bridges, Leidschenveen (Van de Brug et al., 2000)*

### B.1.6 CNC milled EPS formwork

Expanded polystyrene (EPS) blocks can be used as a mold after being milled with a computer controlled cutter head or a wire. Depending on the milling machine, the robot arm can translate in three directions and rotate over two axes, allowing for almost any shape to be cut and ensuring a very accurate result. Afterward, the blocks need to be coated for a good finish and to make demolding easier. A disadvantage is a low adaptability: for every different shape, a new mold needs to be cut. The molds are also not reusable. This was not an issue for the 3TU prototype (Figure B.8) & the Unikabeton prototype (Figure B.9), and for highly unique projects such as the

Spencer Dock Bridge (Figure B.10), however.

*(a) Formwork design* *(b) Milling the EPS blocks* *(c) Finished EPS molds* *(d) Finished structure*

***Figure B.8:*** *3TU prototype (Studio RAP, 2015)*

*(a) Milled elements and reinforcement* *(b) Finshing of the concrete* *(c) Finished structure*

***Figure B.9:*** *Unikabeton prototype, Aarhus (Dombernowsky and Sondergaard, 2010)*

*(a) Spraying the coating* *(b) The elements put together on site* *(c) Finished structure*

***Figure B.10:*** *Spencer Dock Bridge, Dublin (Nedcam, 2008)*

### B.1.7 Fabric forming

The production method of fabric forming replaces traditional formwork materials with a flexible textile membrane. When concrete is poured inside, the fabric of the mold tends to follow efficient structural curves. These curves are a result from its own self-weight and can be influenced by changing and/or tightening the shape of the textile. This characteristic of the method also brings severe constraints in the possible shapes. There is still a lot of research being conducted in order to discover the possibilities in shaping the formwork. Several examples are displayed in Figure B.11.

*(a) Precast wall panel (2002)*

*(b) Four meter span truss (2006)*

*(c) Column (2007)*

***Figure B.11:*** *Research by the CAST-laboratory led by prof. Mark West from the University of Manitoba (ISOFF, nd)*

### B.1.8 Vacuum stabilized formwork

Please refer to Chapter 5.

### B.1.9 Additive manufacturing

Additive manufacturing (or 3D printing) is a relatively new production method still in development. It shows a lot of promise for the production of topology-optimized structures: a 3D printer can distribute material where it is needed, just like topology optimization can. At the moment, there are two major additive manufacturing techniques in development which are to be used in the construction of concrete housing. The first is a method where a concrete house is 3D printed layer by layer directly on top of each other. Examples of this are Contour Crafting (CC) on behalf of the University of Southern California (Figure B.13a), and the houses by WinSun Decoration Design Engineering in China (Figure B.12). The second method is the 3D print canal house to be built in Amsterdam and is researched by DUS Architects, which prints a cast using a bioplastic that is later filled with a lightweight foaming concrete (Figure B.13b).

*(a) Printing*

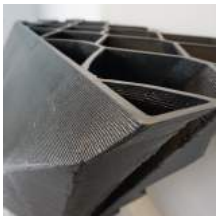
*(b) Wall element*

*(c) Finished house*

***Figure B.12:*** *Housing designed by WinSun (3ders, 2014)*

*(a) Contour crafting printhead*

*(b) Contour crafting beam*



*(c) 3D print canal house cast*

*(d) 3D print canal house filled cast*

***Figure B.13:*** *Additive manufacturing of concrete structures*

## B.2 LIST OF MANUFACTURABILITY CONSTRAINTS

### B.2.1 Minimum radius formwork curvature

Some of the production methods mentioned in this chapter are more suitable for manufacturing freely-curved structures than others. A small analysis of the minimum radius (Figure B.14) of the formwork curvature aims to demonstrate these differences. A smaller radius means that more curvature can be reached and the structure is less constrained by this manufacturability aspect.

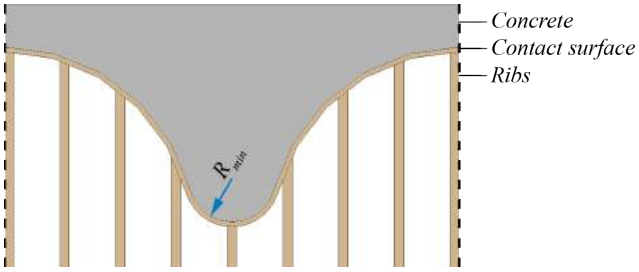



***Figure B.14:*** *Minimum radius of the formwork curvature for a cross section of a rib*

#### Timber formwork

Especially for the timber formwork that uses ribs, the minimum radius is important as there is a double-curvature. Table B.1 shows these radii for different grain directions and plate thicknesses. The timber formwork method that is more similar to the waffle slab has most of its curvature in one direction, so the more constraining curvature parallel to the grain is of less importance.

***Table B.1:*** *Minimum radii plywood (Dubbeldam, 1991)*

| Thickness (mm) | Curvature parallel to grain Minimum radius (m) | Curvature perpendicular to grain Minimum radius (m) |
|---|---|---|
| 4 | 0.40 - 0.75 | 0.15 |
| 6.5 | 1.30 - 1.50 | 0.40 - 0.70 |
| 9 | 2.00 - 2.40 | 0.90 - 1.60 |
| 12 | 2.70 - 3.60 | 1.80 - 2.50 |
| 15 | 3.80 - 4.80 | 2.40 - 3.60 |
| 18 | 4.80 - 7.00 | 3.60 - 6.00 |

Steel formwork
Making a steel mold with single curvature is relatively simple with the use of rollers. The second curvature is extremely complex to achieve.

Reconfigurable mold
The flexible mold method by (Schipper, 2015) has been applied successfully for the production of precast concrete elements with a radius down to 1.50 m.

Ferrocement formwork
No constraint.

Polyester formwork
The polyester formwork is only constrained by the production method of the counter-mold.

CNC milled EPS formwork
A wire cutter, the first CNC milling option, can only cut in straight lines. This does not mean double-curved structures are impossible to manufacture: a hyperbolic paraboloid consists of straight lines for example. The second option of a cutter head robot that can move in three directions will show milling paths near high curvatures. Therefore, for more complex structures a five-directional head is recommended. It moves perpendicular to the surface and leaves a smoother result. There are no curvature constraints from the production method.

Fabric forming
No constraint.

Vacuum stabilized formwork
Whether there is a radius constraint on the formwork depends on the air pressure difference and the elasticity of the plastic foil. The V-process from the metalworking industry proves that there are virtually no constraints as the foil follows the counter mold almost perfectly. On a larger scale, the ISOFF workshop does show some issues, as can be seen in the top right of Figure 5.4. More research needs to be conducted in order to quantify the minimum radius of the curvature.

Additive manufacturing
No constraint.

### B.2.2 Void removal & casting constraint

The result from a regular topology optimization of a structure often has the following problem when manufacturability is taken into account. Usually, there are voids in the optimized result. For small or practically two-dimensional objects this can be overcome, for the production of a concrete floor, this is less than ideal. Void removal, therefore, is an essential constraint (Figure B.15, left). The second problem, more related to the scope of this thesis, is the process of the casting of an element. For a floor, it makes sense to have the formwork on the bottom and to pour the concrete from the top. This results in a so-called casting constraint, visualized in Figure B.15 on the right.

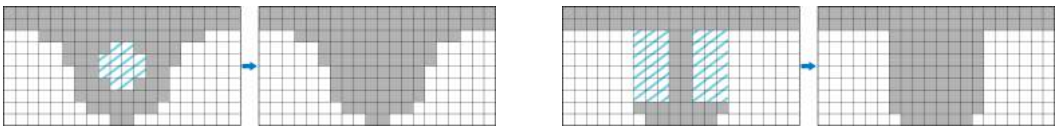



***Figure B.15:*** *Void removal and casting constraint for a cross section of a rib*

Timber formwork
Applicable.

Steel formwork
Applicable.

Reconfigurable mold
Applicable.

Ferrocement formwork
Applicable.

Polyester formwork
Applicable.

CNC milled EPS formwork
Applicable.

Fabric forming
Applicable.

Vacuum stabilized formwork
Applicable.

Additive manufacturing
Void removal should be applied, as falsework inside the structure cannot be removed. The casting constraint should only be utilized when falsework is not desired.

### B.2.3 Minimum thickness

In concrete construction, it is established that there needs to be a certain concrete cover over the reinforcement bars. The minimum thickness in the model should then be equal to twice the concrete cover plus the thickness of the reinforcement layer(s). For fiber-reinforced concrete there also is a minimum thickness, otherwise the risk of

breakage is too high. This constraint is applicable to all the production methods.

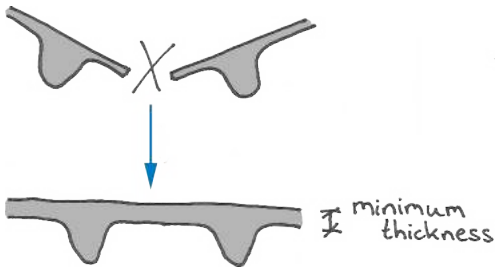



***Figure B.16:*** *Minimum thickness for a cross section of two hypothetical ribs*

### B.2.4 Angle constraint

Most production methods are not suitable for the production of structures with sharp angles. While concrete also does not benefit from very sharp angles, in this section only the feasibility of the production method is regarded. The constraint is not applicable to the reconfigurable mold, since it is restricted by its curvature constraint. For a vacuumatic formwork, the minimum radius of the formwork curvature should already solve the restrictions on angles.

### B.2.5 Formwork size

Depending on the size of the formwork, an entire floor can be cast in one go, or needs to be cast in elements that need to be connected on site.

#### Timber formwork

No constraint.

#### Steel formwork

No constraint.

#### Reconfigurable mold

The reconfigurable mold currently has a size limit of roughly $2 \times 1$ m$^2$, but larger elements are expected to be feasible, according to Schipper (2015).

#### Ferrocement formwork

No constraint.

#### Polyester formwork

No constraint.

#### CNC milled EPS formwork

The EPS blocks come in two standard sizes: a small $4.00 \times 1.25 \times 1.00$ m and a larger $8.00 \times 1.25 \times 1.00$ m. Usually, the smaller block size will be used because these can result in less material waste. Combining the blocks after milling is not an issue, however, so casting can still be achieved in one go.

#### Fabric forming

No constraint.

Vacuum stabilized formwork
The size of the flask influences the size of the element. Larger elements are desirable but a larger flask comes with several disadvantages. The larger the flask, the more powerful the vacuum pump needs to be in order to create a large enough pressure difference. Secondly, the weight of the unit is increased dramatically as the amount of sand increases. This can result in a massive, unmanageable mold. However, from the V-process it is already established that upgrading the size does not necessarily pose a problem, because the metal industry already manufactures elements of several meters long.

Additive manufacturing
No constraint.

### B.2.6 CNC milling robotic arm range

This constraint is only applicable to the CNC milling of EPS blocks. The robotic arm of the milling machine has a certain range in which it can operate. This means that very large mold parts are restricted by this range. The degree to which this constraint is leading is closely related to the EPS block size.

## B.3 CONCLUSION

The result from this chapter is visualized in Table B.2. It shows each analyzed production method, which constraints are applicable, and the extent to which they are decisive when designing for this method. Two conclusions can be made: there are three constraints which are shared across all production methods. Void removal, a casting constraint, and minimum member thickness can be viewed as the minimum requirements for adding manufacturability in the design process of a concrete floor. The second conclusion is that there is not a single method with which a design can be made with only these three constraints because there are always additional constraints necessary. From Table B.2, it seems the polyester formwork requires only three constraints. The counter mold needs to be made with any of the other methods, however.

***Table B.2:*** *Production methods and their accompanying constraints; a larger/darker circle means the constraint is more influential*

| | Maximum curvature | Void removal | Casting constraint | Minimum thickness | Angle constraint | Formwork size | Robotic arm range |
|---|---|---|---|---|---|---|---|
| Timber formwork – curved | ● | ● | ● | ● | ● | | |
| Timber formwork – waffle | · | ● | ● | ● | ● | | |
| Steel formwork | ● | ● | ● | ● | · | | |
| Reconfigurable mold | ● | ● | ● | ● | ● | ● | |
| Ferrocement formwork | · | ● | ● | ● | ● | | |
| Polyester formwork | | ● | ● | ● | | | |
| CNC milled EPS formwork | | ● | ● | ● | · | · | ● |
| Fabric forming | | ● | ● | ● | · | | |
| Vacuum stabilized formwork | · | ● | ● | ● | · | · | |
| Additive manufacturing | | ● | · | ● | ● | | |

# C

# Reinforced concrete costs

In the construction industry, estimating the costs of a project is of great importance to the contractor and owner. A good estimation of a construction project includes the total costs of all materials, labor, subcontractors, equipment, and overhead. These factors can be divided into direct costs, indirect costs, and overhead costs (Figure C.1). Direct costs include the expense of materials, labor, and equipment. These are all directly related to an activity or a service. The indirect costs, on the other hand, do not have this clear attribution and are instead related to the realization of a service in general. Overhead costs include expenses for risk, profit, interest, etc. (Van der Horst, 2013). Indirect costs and general overhead are often a percentage of the total costs, so they are irrelevant when comparing alternatives. Consequently, they are not incorporated in the cost function in this chapter.

The unit price estimating method is most commonly used in competitive bidding. It consists of dividing a building into its basic elements and applying a price per unit of material use. As long as it is properly applied it is an accurate tool to predict project costs. The final Estimate Summary accumulates the cost of each separate element. It is greatly dependent on the location and time of the project so the result may not be completely accurate everywhere in the world. However, it can still be utilized as a tool to compare different floor lab designs (Del Pico, 2013).

## C.1 INTEGRAL COST

The integral costs of reinforced concrete is the sum of the concrete costs (Appendix C.2), the reinforcement costs (Appendix C.3), and the formwork costs (Appendix C.4), resulting in Equation (C.1).

$$C = C_c + C_r + C_f \tag{C.1}$$

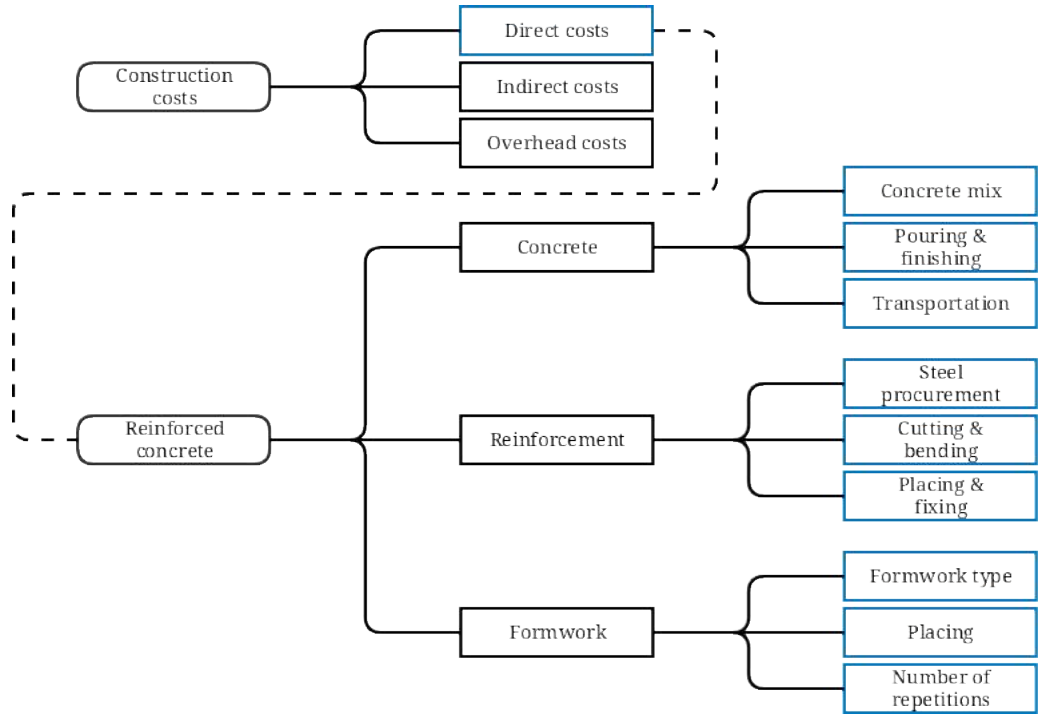
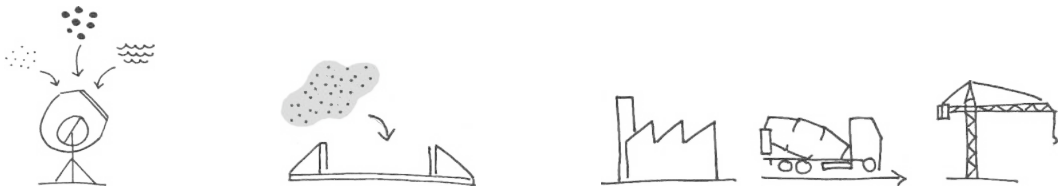



***Figure C.1:*** *Components of the reinforced concrete cost*

## C.2 CONCRETE COSTS

As displayed in Figure C.2 and Equation (C.2), the concrete costs consist of three components. Procurement of the materials includes cement, sand, gravel, admixtures, and water. The ratio depends on the type of concrete. The mixture is made at a batching plant where depreciation, maintenance, and interest drive the costs. The price of pouring and finishing depends on the shape, size, and complexity of the element. Lastly, the costs of transportation from the plant to the building site depend on the distance between these two.



***(a)*** *Material* ***(b)*** *Labor* ***(c)*** *Transportation*

***Figure C.2:*** *Concrete costs*

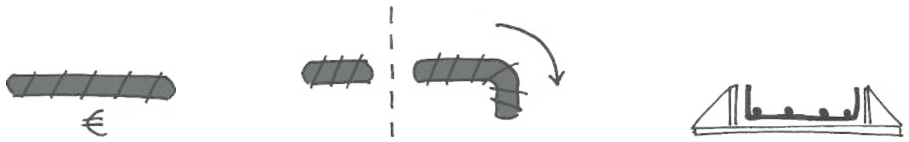
$$C_c = \underbrace{V_c \cdot I_c}_{\text{material}} + \underbrace{hr \cdot C_{mhr}}_{\text{labor}} + \underbrace{(T_{c:0} + T_{c:1} \cdot L)}_{\text{transportation}} \tag{C.2}$$

In which:

| | |
|---|---|
| $C_c$ | *costs of concrete* [€] |
| $V_c$ | *volume of concrete* $[m^3]$ |
| $I_c$ | *investment for the procurement of the concrete* $[€/m^3]$ |
| $T_{c:0}$ | *transportation cost initial* [€] |
| $T_{c:1}$ | *transportation cost per kilometer* $[€/km]$ |
| $L$ | *travel distance* $[km]$ |
| $hr$ | *workload in hours* $[h]$ |
| $C_{mhr}$ | *costs of one man-hour* $[€/h]$ |

## C.3 REINFORCEMENT COSTS

The costs of reinforcement depend first on the procurement of the steel as the market value has the greatest impact on the price. When the reinforcement steel is on location it needs to be bent and cut in order to fit the formwork. Besides the diameter of the steel, the shape and complexity of the element play a significant role again. This has been summarized in Figure C.3 and Equation (C.3). Any potential fiber reinforcement costs can be included in the concrete costs in Equation (C.2), because it is part of the concrete mixture and not cut, bent, and placed on site or the factory.



***(a)*** *Material* ***(b)*** *Labor* ***(c)*** *Labor*

***Figure C.3:*** *Reinforcement costs*

$$C_r = \underbrace{V_r \cdot I_r}_{\text{material}} + \underbrace{hr \cdot C_{mhr}}_{\text{labor}} \tag{C.3}$$

In which:

| | |
|---|---|
| $C_r$ | *costs of reinforcement* [€] |
| $V_r$ | *volume of reinforcement steel* $[m^3]$ |
| $I_r$ | *investment for the procurement of the steel* $[€/m^3]$ |
| $hr$ | *workload in hours* $[h]$ |
| $C_{mhr}$ | *costs of one man-hour* $[€/h]$ |

## C.4 FORMWORK COSTS

There are many types of formwork and the costs depend on which specific type is used. Important aspects of choosing a formwork system are the complexity of the shape and the number of times the formwork needs to be reused. A steel form has high initial costs compared to a timber mold, for example, but because it can be used a greater number of times it may still be economically viable. For traditional structures, the formwork costs exceed the concrete and reinforcement costs so there is a lot to be gained here (Peurifoy and Oberlender, 1996). A cost model for formwork is displayed in Figure C.4 and Equation (C.4).

***(a)*** *Labor*

***(b)*** *Formwork efficiency*

***Figure C.4:*** *Formwork costs*

$$C_f = \underbrace{hr \cdot C_{mhr}}_{\text{labor}} + \underbrace{\frac{I_f}{N}}_{\text{form efficiency}} \tag{C.4}$$

In which:

| | |
|---|---|
| $C_f$ | *costs of formwork* [€] |
| $I_f$ | *initial investment of the formwork system* [€] |
| $N$ | *number of repetitions, or times the formwork can be reused* [−] |
| $hr$ | *workload in hours* [$h$] |
| $C_{mhr}$ | *costs of one man-hour* [€/$h$] |

# D

# Sustainability comparison

The input values displayed on the next page have been determined as follows:

- The amount of concrete for both floor slabs is determined in Section 7.1.1.
- The sheet does not allow C50/60 to be used as the concrete strength class. Therefore, the similar C53/65 has been assumed for both floor slabs.
- The environmental class of XC1 is assumed according to Table 4.1 of NEN-EN 1992-1-1. This class can be applied concrete structures inside buildings with a low humidity. Both floor slabs have the same environmental class.
- The average concrete mixture in The Netherlands consists of roughly 65% CEM III (blast-furnace cement) and 35% CEM I (Portland cement). This mixture has been assumed for both floor slabs.
- The transport distance is assumed to be 30 km for both floor slabs.
- The amount of reinforcement has been estimated with a two-dimensional simplification of the floor slab. In the Ultimate Limit State, the total distributed load is roughly $q_d = 1.5 \cdot q_{live} + 1.35 \cdot q_{dead} = 1.5 \cdot 5.0 + 1.35 \cdot 2.5 = 10.9$ kN/m. This results in a bending moment at midspan of $M_d = \frac{1}{8} q_d L^2 = \frac{1}{8} 10.9 \cdot 7.0^2 = 67$ kN m. Assuming an internal arm of $z = 0.9d = 0.9(h - 0.03 - 0.05)$, then the required amount of reinforcement is $A_s = N_s/435 = \frac{M_d}{z \cdot 435} = 1582$ mm$^2$/m. Assuming that the density of steel is equal to 7859 kg/m$^3$, and that the total reinforcement in the slab is $\frac{3}{2}$ times the lower reinforcement layer, then the amount of reinforcement is determined to be $7859 \times 1.5 \times 15.81 \cdot 10^{-4}/0.1875 = 100$ kg/m$^3$ for the traditional floor slab.

# Ontwerptool groen beton

$CO_2$-emissie "cradle to site"

| | | | | | |
|---|---|---|---|---|---|
| project | : | Duurzaam construeren | werkcode | : | |
| opdrachtgever | : | TU Delft | referentie | : | mgn |
| onderwerp | : | $CO_2$ quick scan | printdatum | : | 16-jun-16 |

## origineel betonmengsel/ontwerp

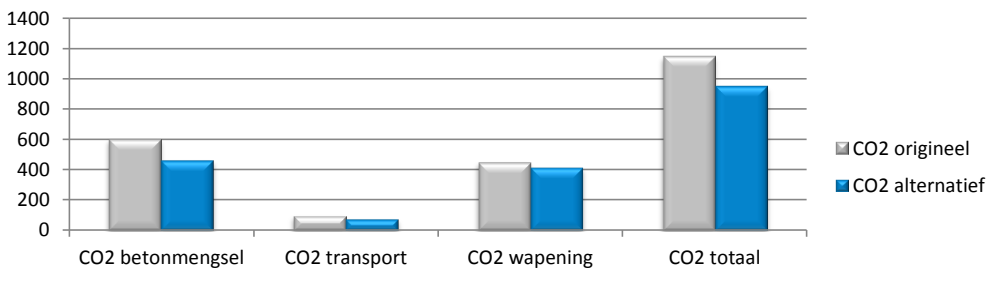
| | | | | emissie | | bouwkosten | | |
|---|---|---|---|---|---|---|---|---|
| hoeveelheid beton | | 3.28 | $m^3$ | | | | | |
| betonkwaliteit | | C53/65 | | | | | | |
| milieuklasse | | XC1 | | | | | | |
| typering mengsel | | prefab normaal | | | | | | |
| cementgehalte | | 370 | $kg/m^3$ | | | | | |
| | waarvan | 65% | CEM III | 72 | $kg\ CO_2/m^3$ | € | 18.04 | |
| | | 35% | CEM I | 106 | " | € | 11.66 | |
| vliegas toegevoegd | | 0 | $kg/m^3$ | 0.0 | " | € | - | |
| water | | 241 | $kg/m^3$ | 0.1 | " | € | 0.26 | |
| zand | | 667 | $kg/m^3$ | 2.1 | " | € | 3.34 | |
| grind | | 1,001 | $kg/m^3$ | 3.2 | " | € | 18.01 | |
| transport grondstoffen-centrale | | | | 10 | " | | | |
| transport centrale-bouwplaats | | 30 | km | 13.7 | " | € | 17.00 | |
| productie | | | | 5.9 | " | € | 8.25 | |
| | | | Totaal beton | 213 | $kg\ CO_2/m^3$ | € | 76.56 | $/m^3$ |
| wapeningshoeveelheid | | 100 | $kg/m^3$ | 137 | " | € | 125.00 | |
| transport wapening-bouwplaats | | 0 | km | 0.0 | " | € | 1.52 | |
| | | | Totaal staal | 137 | $kg\ CO_2/m^3$ | € | 126.52 | $/m^3$ |
| | | | ***Totaal*** | ***350*** | ***kg $CO_2/m^3$*** | ***€*** | ***203.07*** | ***$/m^3$*** |

## alternatief betonmengsel/ontwerp

| | | | | emissie | | bouwkosten | | |
|---|---|---|---|---|---|---|---|---|
| hoeveelheid beton | | 2.52 | $m^3$ | | | | | |
| betonkwaliteit | | C53/65 | | | | | | |
| milieuklasse | | XC1 | | | | | | |
| typering mengsel | | prefab normaal | | | | | | |
| cementgehalte | | 370 | $kg/m^3$ | | | | | |
| | waarvan | 65% | CEM III | 72 | $kg\ CO_2/m^3$ | € | 18.03 | |
| | | 35% | CEM I | 106 | " | € | 11.65 | |
| vliegas toegevoegd | | 0 | $kg/m^3$ | 0.0 | " | € | - | |
| water | | 241 | $kg/m^3$ | 0.1 | " | € | 0.26 | |
| zand | | 667 | $kg/m^3$ | 2.1 | " | € | 3.34 | |
| grind | | 1,001 | $kg/m^3$ | 3.2 | " | € | 18.01 | |
| transport grondstoffen-centrale | | | | 10 | " | | | |
| transport centrale-bouwplaats | | 30 | km | 13.7 | " | € | 17.00 | |
| productie | | | | 5.9 | " | € | 8.25 | |
| | | | Totaal beton | 213 | $kg\ CO_2/m^3$ | € | 76.54 | $/m^3$ |
| wapeningshoeveelheid | | 120 | $kg/m^3$ | 164 | " | € | 150.00 | |
| transport wapening-bouwplaats | | 0 | km | 0.0 | " | € | 1.52 | |
| | | | Totaal staal | 164 | $kg\ CO_2/m^3$ | € | 151.52 | $/m^3$ |
| | | | ***Totaal*** | ***378*** | ***kg $CO_2/m^3$*** | ***€*** | ***228.06*** | ***$/m^3$*** |

## vergelijking betonmengsels/ontwerp (alternatief - origineel)

| | | | | | |
|---|---|---|---|---|---|
| verschil hoeveelheid beton | -0.77 | $m^3$ | verschil emissie | -199 | $kg\ CO_2$ |
| verschil hoeveelheid wapening | -26.25 | kg | verschil kosten | € 92.62- | |



# E

# Getting started guide

1. Download and install a Python distribution: Enthought Canopy includes a user-friendly package manager and doesn't require a separate Python 2.7 install. It can be downloaded at:
   `https://store.enthought.com/downloads/#default`
   Other distributions, such as Anaconda, Pythonxy, etc. also include Python combined with the most popular packages. They can be found at:
   `https://www.python.org/download/alternatives/`

   Alternatively, download and install Python 2.7:
   `https://www.python.org/downloads/`
   And download and install the required packages separately: numpy, scipy, time, traits, and mayavi.

2. Download and install an editor. A list of editors can be found at:
   `https://wiki.python.org/moin/PythonEditors`

3. Run `TopOpPy.py` via one of two ways:
   - From the editor
   - From command prompt

4. Changes to the input can be made in the code itself. The included file `input.py` can help with reproducing the problems presented in this thesis.

5. Other included algorithms at `http://repository.tudelft.nl/`:
   - `TopOpPy_concrete.py`: the modified algorithm for designing the topology-optimized floor.
   - `TopOpPy_concrete_sequential.py`: the same modified algorithm for designing the topology-optimized floor but with a sequential addition of the constraints. This gives a solution closer to the global optimum.
   - `TopOpPy_visualizer.py`: an algorithm than can read `.csv` files that contain the physical densities `xPhys`. These are automatically generated at the end of the optimization with both concrete versions of the algorithm.

# F

# Topology optimization algorithm

```
"""
==============================================================================================
--- Topology optimization of a Concrete Floor Slab Guided by Vacuumatic Formwork Constraints ---
==============================================================================================
This code is discussed in the paper 'Topology optimization of a Concrete Floor Slab Guided by
Vacuumatic Formwork Constraints' by C.M. Magan (2016), which can be found on
repository.tudelft.nl

Author: Christopher Magan
Date:   July 2016
        Delft University of Technology
----------------------------------------------------------------------------------------------
"""
import numpy as np
from scipy.sparse import coo_matrix
from scipy.sparse.linalg import spsolve
import time
from traits.etsconfig.api import ETSConfig
ETSConfig.toolkit = 'qt4' # Force PyQt4 utilization for the GUI of MayaVi
from mayavi import mlab

# ==============================================================================================
# --- Input by the User ---
# ==============================================================================================
# ---------------------
# --- Design Space ---
nelx = int(60)
nely = int(20)
nelz = int(4)
volfrac = float(0.3)
penal = float(3.0) # default value: 3.0
rmin = float(1.5) # default value: 1.5

# ------------------------
# --- Loop Parameters ---
maxloop = int(200) # default value: 200
tolx = float(0.01) # default value: 0.01

# ---------------------------
# --- Material Properties ---
Ecmax = float(1.0) # default value: 1.0
Emin = float(1e-9) # default value: 1e-9
nu = float(0.3) # default value: 0.3

# ---------------------------------
# --- Loads & Degrees of Freedom---
il = np.array([nelx])
jl = np.array([0])
kl = np.array([np.arange(0,nelz+1)])
loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl)
loaddof = 3*loadnid.T - 1
load = float(-1.0) # default value: -1.0

# -----------------------------------------------
# --- Boundary Conditions & Degrees of Freedom---
```

```
# ibc = empty
[jbc,kbc] = np.mgrid[1:nely+2,1:nelz+2]
bcnid = (kbc.T-1)*(nely+1)*(nelx+1)+jbc.T
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid-2)).flatten()

# ========================================================================================
# --- Topology Optimization Definitions---
# ========================================================================================
# -----------------
# --- Main Code ---
def main(nelx,nely,nelz,volfrac,penal,rmin):
    starttimer = time.clock()
    print "Minimum compliance topology optimization guided by manufacturability constraints."
    print "Number of elements: " + str(nelx) + " x " + str(nely) + " x " + str(nelz)
    print "Volume fraction:    " + str(volfrac)
    print "Penalty factor:     " + str(penal)
    print "Minimum radius:     " + str(rmin)
    # Prepare FEA
    nele = nelx*nely*nelz
    ndof = 3*(nelx+1)*(nely+1)*(nelz+1)
    F = np.zeros((ndof,1))
    F[loaddof-1,0] = load
    U = np.zeros((ndof,1))
    freedofs = np.setdiff1d(np.arange(1,ndof+1),bcdof)
    g = 0 # for oc
    KE = ElStiffnMat(nu)
    edofMat = np.zeros((nelx*nely*nelz,24),dtype=int)
    for elz in range(nelz):
        for elx in range(nelx):
            for ely in range(nely):
                el =  ely    + elx   * nely    + elz  *  nelx* nely
                n1 = (ely+1) + elx   *(nely+1) + elz  *  nelx*(nely+1) + (nely+1)*elz
                n2 = (ely+1) +(elx+1)*(nely+1) + elz  *  nelx*(nely+1) + (nely+1)*elz
                n3 = (ely+1) +(elx+1)*(nely+1) + (elz+1)*nelx*(nely+1) + (nely+1)*elz
                edofMat[el,:]=np.array( [3*n1+1, 3*n1+2, 3*n1+3, 3*n2+1, 3*n2+2, 3*n2+3, 3*n2-2, 3*n2
 -1, 3*n2, 3*n1-2, 3*n1-1, 3*n1, 3*n3+1, 3*n3+2, 3*n3+3, 3*n3+13+(3*nely-9), 3*n3+14+(3*nely-9), 3*
 n3+15+(3*nely-9), 3*n3+10+(3*nely-9), 3*n3+11+(3*nely-9), 3*n3+12+(3*nely-9), 3*n3-2, 3*n3-1, 3*n3
 ] )
    iK = np.kron(edofMat,np.ones((24,1))).flatten()
    jK = np.kron(edofMat,np.ones((1,24))).flatten()
    # Prepare density filter
    sH,iH,jH = df(nelx,nely,nelz,rmin)
    H  = coo_matrix((sH,(iH,jH)),shape=(nele,nele)).tocsc()
    Hs = H.sum(1)
    # Initialize iteration
    x = volfrac*np.ones(nele)
    xold = x.copy()
    xPhys = x.copy()
    loop = 0
    change = 1.
    ce = np.ones(nele)
    dc = np.ones(nele)
    dv = np.ones(nele)
    # Start iteration
    change = 1.
    while change>tolx and loop<maxloop:
        loop = loop + 1
        # FEA
        sK = ((KE.flatten()[np.newaxis]).T*(Emin+(xPhys)**penal*(Ecmax-Emin)).flatten()).flatten(order=
 'F')
        K = coo_matrix((sK,(iK-1,jK-1)),shape=(ndof,ndof)).tocsc()
        K = (K+K.T)/2
        K = K[freedofs-1,:][:,freedofs-1]
        U[freedofs-1,0] = spsolve(K,F[freedofs-1,0])
        # Objective and sensitivities for a compliance problem
        ce[:] = ( np.dot(U[edofMat-1].reshape(nele,24),KE) * U[edofMat-1].reshape(nele,24) ).sum(1)
        c     = ((Emin+xPhys**penal*(Ecmax-Emin)) * ce).sum()
        dc[:] = -penal*xPhys**(penal-1)*(Ecmax-Emin)*ce
        dv[:] = np.ones(nele)
        # Filtering
        dc[:] = np.asarray(H*(dc[np.newaxis].T/Hs))[:,0]
        dv[:] = np.asarray(H*(dv[np.newaxis].T/Hs))[:,0]
        # Optimality criteria update
        xold[:] = x
        (x[:],g)= oc(nelx,nely,nelz,x,volfrac,dc,dv,g)
        xPhys[:]= np.asarray(H*(x[np.newaxis].T)/Hs)[:,0]
        # Compute change
        change = np.linalg.norm(x.reshape(nele,1)-xold.reshape(nele,1),np.inf)
        # Print iteration history
        print 'It.: {}     Obj.: {:.3f}     Vol.: {:.2f}     Ch.: {:.3f}'.format(loop,c,np.mean(xPhys
 [:]),change)
    # Plot final design
    endtimer = time.clock()
    print 'Optimization finished in {:.0f}s. Preparing visualization...'.format(np.ceil(endtimer-
 starttimer))
    displayfigure(xPhys,nelx,nely,nelz)
    print 'MayaVi finished.'

# ----------------------------------
# --- Elemental Stiffness Matrix ---
def ElStiffnMat(nu):
    A = np.array([[32, 6, -8, 6, -6, 4, 3, -6, -10, 3, -3, -3, -4, -8],[-48, 0, 0, -24, 24, 0, 0, 0,
```

```
     12, -12,  0,  12,  12,  12]])
    k = 1/144. * A.T.dot( np.array([[1],[nu]]) )

    k1 = np.empty((6,6))
    k1[0,0]=k[0]; k1[0,1]=k[1]; k1[0,2]=k[1]; k1[0,3]=k[2]; k1[0,4]=k[4]; k1[0,5]=k[4]
    k1[1,0]=k[1]; k1[1,1]=k[0]; k1[1,2]=k[1]; k1[1,3]=k[3]; k1[1,4]=k[5]; k1[1,5]=k[6]
    k1[2,0]=k[1]; k1[2,1]=k[1]; k1[2,2]=k[0]; k1[2,3]=k[3]; k1[2,4]=k[6]; k1[2,5]=k[5]
    k1[3,0]=k[2]; k1[3,1]=k[3]; k1[3,2]=k[3]; k1[3,3]=k[0]; k1[3,4]=k[7]; k1[3,5]=k[7]
    k1[4,0]=k[4]; k1[4,1]=k[5]; k1[4,2]=k[6]; k1[4,3]=k[7]; k1[4,4]=k[0]; k1[4,5]=k[1]
    k1[5,0]=k[4]; k1[5,1]=k[6]; k1[5,2]=k[5]; k1[5,3]=k[7]; k1[5,4]=k[1]; k1[5,5]=k[0]
    k2 = np.empty((6,6))
    k2[0,0]=k[8]; k2[0,1]=k[7]; k2[0,2]=k[11];k2[0,3]=k[5]; k2[0,4]=k[3]; k2[0,5]=k[6]
    k2[1,0]=k[7]; k2[1,1]=k[8]; k2[1,2]=k[11];k2[1,3]=k[4]; k2[1,4]=k[2]; k2[1,5]=k[4]
    k2[2,0]=k[9]; k2[2,1]=k[9]; k2[2,2]=k[12];k2[2,3]=k[6]; k2[2,4]=k[3]; k2[2,5]=k[5]
    k2[3,0]=k[5]; k2[3,1]=k[4]; k2[3,2]=k[10];k2[3,3]=k[8]; k2[3,4]=k[1]; k2[3,5]=k[9]
    k2[4,0]=k[3]; k2[4,1]=k[2]; k2[4,2]=k[4]; k2[4,3]=k[1]; k2[4,4]=k[8]; k2[4,5]=k[11]
    k2[5,0]=k[10];k2[5,1]=k[3]; k2[5,2]=k[5]; k2[5,3]=k[11];k2[5,4]=k[9]; k2[5,5]=k[12]
    k3 = np.empty((6,6))
    k3[0,0]=k[5]; k3[0,1]=k[6]; k3[0,2]=k[3]; k3[0,3]=k[8]; k3[0,4]=k[11];k3[0,5]=k[7]
    k3[1,0]=k[6]; k3[1,1]=k[5]; k3[1,2]=k[3]; k3[1,3]=k[9]; k3[1,4]=k[12];k3[1,5]=k[9]
    k3[2,0]=k[4]; k3[2,1]=k[4]; k3[2,2]=k[2]; k3[2,3]=k[7]; k3[2,4]=k[11];k3[2,5]=k[8]
    k3[3,0]=k[8]; k3[3,1]=k[9]; k3[3,2]=k[1]; k3[3,3]=k[5]; k3[3,4]=k[10];k3[3,5]=k[4]
    k3[4,0]=k[11];k3[4,1]=k[12];k3[4,2]=k[9]; k3[4,3]=k[10];k3[4,4]=k[5]; k3[4,5]=k[3]
    k3[5,0]=k[1]; k3[5,1]=k[11];k3[5,2]=k[8]; k3[5,3]=k[3]; k3[5,4]=k[4]; k3[5,5]=k[2]
    k4 = np.empty((6,6))
    k4[0,0]=k[13];k4[0,1]=k[10];k4[0,2]=k[10];k4[0,3]=k[12];k4[0,4]=k[9]; k4[0,5]=k[9]
    k4[1,0]=k[10];k4[1,1]=k[13];k4[1,2]=k[10];k4[1,3]=k[11];k4[1,4]=k[8]; k4[1,5]=k[7]
    k4[2,0]=k[10];k4[2,1]=k[10];k4[2,2]=k[13];k4[2,3]=k[11];k4[2,4]=k[7]; k4[2,5]=k[8]
    k4[3,0]=k[12];k4[3,1]=k[11];k4[3,2]=k[11];k4[3,3]=k[13];k4[3,4]=k[6]; k4[3,5]=k[6]
    k4[4,0]=k[9]; k4[4,1]=k[8]; k4[4,2]=k[7]; k4[4,3]=k[6]; k4[4,4]=k[13];k4[4,5]=k[10]
    k4[5,0]=k[9]; k4[5,1]=k[7]; k4[5,2]=k[8]; k4[5,3]=k[6]; k4[5,4]=k[10];k4[5,5]=k[13]
    k5 = np.empty((6,6))
    k5[0,0]=k[0]; k5[0,1]=k[1]; k5[0,2]=k[7]; k5[0,3]=k[2]; k5[0,4]=k[4]; k5[0,5]=k[3]
    k5[1,0]=k[1]; k5[1,1]=k[0]; k5[1,2]=k[7]; k5[1,3]=k[3]; k5[1,4]=k[5]; k5[1,5]=k[10]
    k5[2,0]=k[7]; k5[2,1]=k[7]; k5[2,2]=k[0]; k5[2,3]=k[4]; k5[2,4]=k[10];k5[2,5]=k[5]
    k5[3,0]=k[2]; k5[3,1]=k[3]; k5[3,2]=k[4]; k5[3,3]=k[0]; k5[3,4]=k[7]; k5[3,5]=k[1]
    k5[4,0]=k[4]; k5[4,1]=k[5]; k5[4,2]=k[10];k5[4,3]=k[7]; k5[4,4]=k[0]; k5[4,5]=k[7]
    k5[5,0]=k[3]; k5[5,1]=k[10];k5[5,2]=k[5]; k5[5,3]=k[1]; k5[5,4]=k[7]; k5[5,5]=k[0]
    k6 = np.empty((6,6))
    k6[0,0]=k[13];k6[0,1]=k[10];k6[0,2]=k[6]; k6[0,3]=k[12];k6[0,4]=k[9]; k6[0,5]=k[11]
    k6[1,0]=k[10];k6[1,1]=k[13];k6[1,2]=k[6]; k6[1,3]=k[11];k6[1,4]=k[8]; k6[1,5]=k[1]
    k6[2,0]=k[6]; k6[2,1]=k[6]; k6[2,2]=k[13];k6[2,3]=k[9]; k6[2,4]=k[1]; k6[2,5]=k[8]
    k6[3,0]=k[12];k6[3,1]=k[11];k6[3,2]=k[9]; k6[3,3]=k[13];k6[3,4]=k[6]; k6[3,5]=k[10]
    k6[4,0]=k[9]; k6[4,1]=k[8]; k6[4,2]=k[1]; k6[4,3]=k[6]; k6[4,4]=k[13];k6[4,5]=k[6]
    k6[5,0]=k[11];k6[5,1]=k[1]; k6[5,2]=k[8]; k6[5,3]=k[10];k6[5,4]=k[6]; k6[5,5]=k[13]

    KE = 1/((nu+1)*(1-2*nu)) *  np.concatenate([np.concatenate([k1  , k2, k3  , k4  ],axis=1),
                                                np.concatenate([k2.T, k5, k6  , k3.T],axis=1),
                                                np.concatenate([k3.T, k6, k5.T, k2.T],axis=1),
                                                np.concatenate([k4  , k3, k2  , k1.T],axis=1)])
    return (KE)

# ------------------------------
# --- Density Filter ---
def df(nelx,nely,nelz,rmin):
    nfilter = int(nelx*nely*nelz*((2*(np.ceil(rmin)-1)+1)**3))
    iH = np.zeros(nfilter)
    jH = np.zeros(nfilter)
    sH = np.zeros(nfilter)
    counter = 0
    for k in range(nelz):
        for i in range(nelx):
            for j in range(nely):
                row = k*nelx*nely + i*nely + j
                k2a = int(np.maximum(k-(np.ceil(rmin)-1),0))
                k2b = int(np.minimum(k+(np.ceil(rmin)),nelz))
                i2a = int(np.maximum(i-(np.ceil(rmin)-1),0))
                i2b = int(np.minimum(i+(np.ceil(rmin)),nelx))
                j2a = int(np.maximum(j-(np.ceil(rmin)-1),0))
                j2b = int(np.minimum(j+(np.ceil(rmin)),nely))
                for k2 in range(k2a,k2b):
                    for i2 in range(i2a,i2b):
                        for j2 in range(j2a,j2b):
                            col = k2*nelx*nely + i2*nely + j2
                            if counter>nfilter-1:
                                iH = np.concatenate((iH,np.array([0])),axis=1)
                                jH = np.concatenate((jH,np.array([0])),axis=1)
                                sH = np.concatenate((sH,np.array([0])),axis=1)
                            iH[counter] = row
                            jH[counter] = col
                            sH[counter] = np.maximum(0.0,rmin-np.sqrt((i-i2)**2+(j-j2)**2+(k-k2)**2))
                            counter = counter+1
    return(sH,iH,jH)

# ------------------------------
# --- Optimality Criterion ---
def oc(nelx,nely,nelz,x,volfrac,dc,dv,g):
    l1 = 0
    l2 = 1e9
    move = 0.2
    xnew = np.zeros(nelx*nely*nelz)
    while (l2-l1)/(l1+l2)>1e-3:
```

```
233         lmid = 0.5*(l2+l1)
            xnew[:] = np.maximum(0.,np.maximum(x-move,np.minimum(1.,np.minimum(x+move,x*np.sqrt(-dc/dv/lmid
     )))))
235         gt = g+np.sum((dv*(xnew-x)))
            if gt>0:
237             l1 = lmid
            else:
239             l2 = lmid
    return(xnew,gt)
241
# ------------------------------
243 # --- Visualization Optimum ---
def displayfigure(xPhys,nelx,nely,nelz):
245     xPlot=np.empty(nelx*nely*nelz)
    for ele in range(nelx*nely*nelz):
247         if xPhys[ele]>0.5:
                xPlot[ele]=1.
249         else:
                xPlot[ele]=0.
251     py,px,pz = np.mgrid[0:nely, 0:nelx, 0:nelz]
    fx = px.T.flatten()
253     fy = py.T.flatten()
    fz = pz.T.flatten()
255     for ele in range(nelx*nely*nelz):
        if xPlot[ele]>0.5:
257             point = mlab.points3d(fz[ele],fx[ele],--fy[ele], mode='cube', opacity=xPlot[ele],
      scale_factor=1., color=(1.,1.,1.))
    mlab.show()
259
# ============================================================================================
261 # --- Topology Optimization Main ---
# ============================================================================================
263 if __name__ == "__main__":
    main(nelx,nely,nelz,volfrac,penal,rmin)
```

# G

# Topology optimization algorithm for the concrete floor slab

```
"""
================================================================================================
--- Topology optimization of a Concrete Floor Slab Guided by Vacuumatic Formwork Constraints ---
================================================================================================
This code is discussed in the paper 'Topology optimization of a Concrete Floor Slab Guided by
Vacuumatic Formwork Constraints' by C.M. Magan (2016), which can be found on
repository.tudelft.nl

Author: Christopher Magan
Date:   July 2016
        Delft University of Technology
------------------------------------------------------------------------------------------------
"""
import numpy as np
from scipy.sparse import coo_matrix
from scipy.sparse.linalg import spsolve
import time
from traits.etsconfig.api import ETSConfig
ETSConfig.toolkit = 'qt4' # Force PyQt4 utilization for the GUI of MayaVi
from mayavi import mlab

# ================================================================================================
# --- Input by the User ---
# ================================================================================================
# --------------------
# --- Design Space ---
nelx = int(112)
nely = int(4)
nelz = int(40)
volfrac = float(0.575)
penal = float(3.0) # default value: 3.0
rmin = float(2.5) # default value: 1.5

# ------------------------
# --- Loop Parameters ---
maxloop = int(999) # default value: 200
tolx = float(0.01) # default value: 0.01

# ---------------------------
# --- Material Properties ---
Ecmax = float(37000000.) # default value: 1.0, for concrete: 37000000. kN/m^2
Etmax = float(12000000.) # default value: 1.0, for concrete: 12000000. kN/m^2
Emin = float(100.) # default value: 1e-9, for concrete: 100. kN/m^2
ft = float(0.) # default value: 0.0
nu = float(0.2) # default value: 0.3, for concrete: 0.2

# ---------------------------------
# --- Loads & Degrees of Freedom---
jl = np.array([nely])
[il,kl] = np.mgrid[1:nelx+2,1:nelz+2]
loadnid = (kl.T-1)*(nely+1)*(nelx+1)+il.T+jl*(il.T-1)
loaddof = 3*loadnid.T - 1
```

```
load = -5.*(7.*2.5)/((nelx+1.)*(nelz+1.)) # in kN/elementface, default value: -1
specweight = -24.*(7.*2.5*0.250)/(nelx*nely*nelz) # in kN/element

# -----------------------------------------------
# --- Boundary Conditions & Degrees of Freedom---
ibc = np.array([0,nelx])
jbc = np.array([0,0])
kbc = np.array([np.arange(0,nelz+1),np.arange(0,nelz+1)])
bcnid = np.hstack((kbc[0]*(nelx+1)*(nely+1)+ibc[0]*(nely+1)+(nely+1-jbc[0]) , kbc[1]*(nelx+1)*(nely+1)+
    ibc[1]*(nely+1)+(nely+1-jbc[1]))) - 2
bcdof = np.hstack((3*bcnid, 3*bcnid-1, 3*bcnid[0:nelz+1]-2)).flatten()

# ---------------------------
# --- Additional Options ---
activetop = 1 # active elements top layer = 1; no active elements = 0
activetopextra = 1 # active elements layer below top = 1; no active elements = 0
selfweight = 1 # self-weight is considered = 1; no self-weight = 0
castingconstraint = 1 # casting constraint is applied = 1; no casting constraint = 0

# ============================================================================================
# --- Topology Optimization Definitions---
# ============================================================================================
# ------------------
# --- Main Code ---
def main(nelx,nely,nelz,volfrac,penal,rmin):
    starttimer = time.clock()
    print "Minimum compliance topology optimization guided by manufacturability constraints."
    print "Number of elements: " + str(nelx) + " x " + str(nely) + " x " + str(nelz)
    print "Volume fraction:    " + str(volfrac)
    print "Penalty factor:     " + str(penal)
    print "Minimum radius:     " + str(rmin)
    # Prepare FEA
    nele = nelx*nely*nelz
    ndof = 3*(nelx+1)*(nely+1)*(nelz+1)
    F = np.zeros((ndof,1))
    F[loaddof-1,0] = load
    U = np.zeros((ndof,1))
    freedofs = np.setdiff1d(np.arange(1,ndof+1),bcdof)
    g = 0 # for oc
    KE = ElStiffnMat(nu)
    edofMat = np.zeros((nelx*nely*nelz,24),dtype=int)
    for elz in range(nelz):
        for elx in range(nelx):
            for ely in range(nely):
                el =  ely    + elx   * nely    + elz  *  nelx* nely
                n1 = (ely+1) + elx   *(nely+1) + elz  *  nelx*(nely+1) + (nely+1)*elz
                n2 = (ely+1) +(elx+1)*(nely+1) + elz  *  nelx*(nely+1) + (nely+1)*elz
                n3 = (ely+1) +(elx+1)*(nely+1) + (elz+1)*nelx*(nely+1) + (nely+1)*elz
                edofMat[el,:]=np.array( [3*n1+1, 3*n1+2, 3*n1+3, 3*n2+1, 3*n2+2, 3*n2+3, 3*n2-2, 3*n2
 -1, 3*n2, 3*n1-2, 3*n1-1, 3*n1, 3*n3+1, 3*n3+2, 3*n3+3, 3*n3+13+(3*nely-9), 3*n3+14+(3*nely-9), 3*
 n3+15+(3*nely-9), 3*n3+10+(3*nely-9), 3*n3+11+(3*nely-9), 3*n3+12+(3*nely-9), 3*n3-2, 3*n3-1, 3*n3
 ] )
    iK = np.kron(edofMat,np.ones((24,1))).flatten()
    jK = np.kron(edofMat,np.ones((1,24))).flatten()
    # Prepare density filter
    sH,iH,jH = df(nelx,nely,nelz,rmin)
    H  = coo_matrix((sH,(iH,jH)),shape=(nele,nele)).tocsc()
    Hs = H.sum(1)
    # Initialize iteration
    x = volfrac*np.ones(nele)
    xold = x.copy()
    xPhys = x.copy()
    loop = 0
    change = 1.
    ce = np.ones(nele)
    dc = np.ones(nele)
    dv = np.ones(nele)
    # Start iteration
    while change>tolx and loop<maxloop:
        loop = loop + 1
        # Update total load with self-weight
        if selfweight == 1:
            Fsw = sw(xPhys,specweight,ndof,nelx,nely,nelz)
            Ftot = F + Fsw
        # FEA
        sK = 1.
        for el in range(nele):
            if loop == 1:
                fele = 0.
            else:
                fn = np.dot(KE,U[edofMat[el,:]-1])
                fele = np.mean( np.array((fn[3],fn[6],fn[15],fn[18],-fn[0],-fn[9],-fn[12],-fn[21])) )
            if fele <= ft: # if the stress in the element is smaller than the tension strength, use
 Ecmax
                sK = np.hstack((sK, ((KE.flatten()[np.newaxis]).T*(Emin+(xPhys[el])**penal*(Ecmax-Emin)
 ).flatten()).flatten(order='F') ))
            if fele > ft:  # if the stress in the element is greater than the tension strength, use
 Etmax
                sK = np.hstack((sK, ((KE.flatten()[np.newaxis]).T*(Emin+(xPhys[el])**penal*(Etmax-Emin)
 ).flatten()).flatten(order='F') ))
        sK = sK[1:]
        K = coo_matrix((sK,(iK-1,jK-1)),shape=(ndof,ndof)).tocsc()
```

```
        K = (K+K.T)/2
        K = K[freedofs-1,:][:,freedofs-1]
        if selfweight == 1:
            U[freedofs-1,0] = spsolve(K,Ftot[freedofs-1,0])
        else:
            U[freedofs-1,0] = spsolve(K,F[freedofs-1,0])
        # Objective and sensitivities for a compliance problem
        ce[:] = ( np.dot(U[edofMat-1].reshape(nele,24),KE) * U[edofMat-1].reshape(nele,24) ).sum(1)
        c  = 1.
        dc = 1.
        for el in range(nele):
            fn = np.dot(KE,U[edofMat[el,:]-1])
            fele = np.mean( np.array((fn[3],fn[6],fn[15],fn[18],-fn[0],-fn[9],-fn[12],-fn[21])) )
            if fele <= ft: # if the stress in the element is smaller than the tension strength, use Ecmax
                c  = np.hstack((c, (Emin+xPhys[el]**penal*(Ecmax-Emin))*ce[el] ))
                dc = np.hstack((dc, -penal*xPhys[el]**(penal-1)*(Ecmax-Emin)*ce[el] ))
            if fele > ft:  # if the stress in the element is greater than the tension strength, use Etmax
                c  = np.hstack((c, (Emin+xPhys[el]**penal*(Etmax-Emin))*ce[el] ))
                dc = np.hstack((dc, -penal*xPhys[el]**(penal-1)*(Etmax-Emin)*ce[el] ))
        c  = (c[1:]).sum()
        dc = dc[1:]
        dv[:] = np.ones(nele)
        # Filtering
        dc[:] = np.asarray(H*(dc[np.newaxis].T/Hs))[:,0]
        dv[:] = np.asarray(H*(dv[np.newaxis].T/Hs))[:,0]
        # Optimality criteria update
        xold[:] = x
        (x[:],g)= oc(nelx,nely,nelz,x,volfrac,dc,dv,g,activetop,castingconstraint)
        xPhys[:]= np.asarray(H*(x[np.newaxis].T)/Hs)[:,0]
        # Compute change
        change = np.linalg.norm(x.reshape(nele,1)-xold.reshape(nele,1),np.inf)
        # Print iteration history
        print 'It.: {}     Obj.: {:.7f}     Defl.: {:.3f}     Vol.: {:.2f}     Ch.: {:.3f}'.format(loop,c,np.amin(U[3*nely+1::3*(nely+1)])*1000.*(nelz/2.5),np.mean(xPhys[:]),change)
    # Plot final design
    endtimer = time.clock()
    print 'Optimization finished in {:.0f}s. Preparing visualization...'.format(np.ceil(endtimer-starttimer))
    np.savetxt("xphys.csv", xPhys, delimiter="\t", fmt='%.1e', header='xPhys')
    displayfigure(xPhys,nelx,nely,nelz)
    print 'MayaVi finished.'

# ----------------------------------
# --- Elemental Stiffness Matrix ---
def ElStiffnMat(nu):
    A = np.array([[32, 6, -8, 6, -6, 4, 3, -6, -10, 3, -3, -3, -4, -8],[-48, 0, 0, -24, 24, 0, 0, 0, 12, -12, 0, 12, 12, 12]])
    k = 1/144. * A.T.dot( np.array([[1],[nu]]) )

    k1 = np.empty((6,6))
    k1[0,0]=k[0]; k1[0,1]=k[1]; k1[0,2]=k[1]; k1[0,3]=k[2]; k1[0,4]=k[4]; k1[0,5]=k[4]
    k1[1,0]=k[1]; k1[1,1]=k[0]; k1[1,2]=k[1]; k1[1,3]=k[3]; k1[1,4]=k[5]; k1[1,5]=k[6]
    k1[2,0]=k[1]; k1[2,1]=k[1]; k1[2,2]=k[0]; k1[2,3]=k[3]; k1[2,4]=k[6]; k1[2,5]=k[5]
    k1[3,0]=k[2]; k1[3,1]=k[3]; k1[3,2]=k[3]; k1[3,3]=k[0]; k1[3,4]=k[7]; k1[3,5]=k[7]
    k1[4,0]=k[4]; k1[4,1]=k[5]; k1[4,2]=k[6]; k1[4,3]=k[7]; k1[4,4]=k[0]; k1[4,5]=k[1]
    k1[5,0]=k[4]; k1[5,1]=k[6]; k1[5,2]=k[5]; k1[5,3]=k[7]; k1[5,4]=k[1]; k1[5,5]=k[0]
    k2 = np.empty((6,6))
    k2[0,0]=k[8]; k2[0,1]=k[7]; k2[0,2]=k[11];k2[0,3]=k[5]; k2[0,4]=k[3]; k2[0,5]=k[6]
    k2[1,0]=k[7]; k2[1,1]=k[8]; k2[1,2]=k[11];k2[1,3]=k[4]; k2[1,4]=k[2]; k2[1,5]=k[4]
    k2[2,0]=k[9]; k2[2,1]=k[9]; k2[2,2]=k[12];k2[2,3]=k[6]; k2[2,4]=k[3]; k2[2,5]=k[5]
    k2[3,0]=k[5]; k2[3,1]=k[4]; k2[3,2]=k[10];k2[3,3]=k[8]; k2[3,4]=k[1]; k2[3,5]=k[9]
    k2[4,0]=k[3]; k2[4,1]=k[2]; k2[4,2]=k[4]; k2[4,3]=k[1]; k2[4,4]=k[8]; k2[4,5]=k[11]
    k2[5,0]=k[10];k2[5,1]=k[3]; k2[5,2]=k[5]; k2[5,3]=k[11];k2[5,4]=k[9]; k2[5,5]=k[12]
    k3 = np.empty((6,6))
    k3[0,0]=k[5]; k3[0,1]=k[6]; k3[0,2]=k[3]; k3[0,3]=k[8]; k3[0,4]=k[11];k3[0,5]=k[7]
    k3[1,0]=k[6]; k3[1,1]=k[5]; k3[1,2]=k[3]; k3[1,3]=k[9]; k3[1,4]=k[12];k3[1,5]=k[9]
    k3[2,0]=k[4]; k3[2,1]=k[4]; k3[2,2]=k[2]; k3[2,3]=k[7]; k3[2,4]=k[11];k3[2,5]=k[8]
    k3[3,0]=k[8]; k3[3,1]=k[9]; k3[3,2]=k[1]; k3[3,3]=k[5]; k3[3,4]=k[10];k3[3,5]=k[4]
    k3[4,0]=k[11];k3[4,1]=k[12];k3[4,2]=k[9]; k3[4,3]=k[10];k3[4,4]=k[5]; k3[4,5]=k[3]
    k3[5,0]=k[1]; k3[5,1]=k[11];k3[5,2]=k[8]; k3[5,3]=k[3]; k3[5,4]=k[4]; k3[5,5]=k[2]
    k4 = np.empty((6,6))
    k4[0,0]=k[13];k4[0,1]=k[10];k4[0,2]=k[10];k4[0,3]=k[12];k4[0,4]=k[9]; k4[0,5]=k[9]
    k4[1,0]=k[10];k4[1,1]=k[13];k4[1,2]=k[10];k4[1,3]=k[11];k4[1,4]=k[8]; k4[1,5]=k[7]
    k4[2,0]=k[10];k4[2,1]=k[10];k4[2,2]=k[13];k4[2,3]=k[11];k4[2,4]=k[7]; k4[2,5]=k[8]
    k4[3,0]=k[12];k4[3,1]=k[11];k4[3,2]=k[11];k4[3,3]=k[13];k4[3,4]=k[6]; k4[3,5]=k[6]
    k4[4,0]=k[9]; k4[4,1]=k[8]; k4[4,2]=k[7]; k4[4,3]=k[6]; k4[4,4]=k[13];k4[4,5]=k[10]
    k4[5,0]=k[9]; k4[5,1]=k[7]; k4[5,2]=k[8]; k4[5,3]=k[6]; k4[5,4]=k[10];k4[5,5]=k[13]
    k5 = np.empty((6,6))
    k5[0,0]=k[0]; k5[0,1]=k[1]; k5[0,2]=k[7]; k5[0,3]=k[2]; k5[0,4]=k[4]; k5[0,5]=k[3]
    k5[1,0]=k[1]; k5[1,1]=k[0]; k5[1,2]=k[7]; k5[1,3]=k[3]; k5[1,4]=k[5]; k5[1,5]=k[10]
    k5[2,0]=k[7]; k5[2,1]=k[7]; k5[2,2]=k[0]; k5[2,3]=k[4]; k5[2,4]=k[10];k5[2,5]=k[5]
    k5[3,0]=k[2]; k5[3,1]=k[3]; k5[3,2]=k[4]; k5[3,3]=k[0]; k5[3,4]=k[7]; k5[3,5]=k[1]
    k5[4,0]=k[4]; k5[4,1]=k[5]; k5[4,2]=k[10];k5[4,3]=k[7]; k5[4,4]=k[0]; k5[4,5]=k[7]
    k5[5,0]=k[3]; k5[5,1]=k[10];k5[5,2]=k[5]; k5[5,3]=k[1]; k5[5,4]=k[7]; k5[5,5]=k[0]
    k6 = np.empty((6,6))
    k6[0,0]=k[13];k6[0,1]=k[10];k6[0,2]=k[6]; k6[0,3]=k[12];k6[0,4]=k[9]; k6[0,5]=k[11]
    k6[1,0]=k[10];k6[1,1]=k[13];k6[1,2]=k[6]; k6[1,3]=k[11];k6[1,4]=k[8]; k6[1,5]=k[1]
    k6[2,0]=k[6]; k6[2,1]=k[6]; k6[2,2]=k[13];k6[2,3]=k[9]; k6[2,4]=k[1]; k6[2,5]=k[8]
    k6[3,0]=k[12];k6[3,1]=k[11];k6[3,2]=k[9]; k6[3,3]=k[13];k6[3,4]=k[6]; k6[3,5]=k[10]
    k6[4,0]=k[9]; k6[4,1]=k[8]; k6[4,2]=k[1]; k6[4,3]=k[6]; k6[4,4]=k[13];k6[4,5]=k[6]
```

```
    k6[5,0]=k[11];k6[5,1]=k[1]; k6[5,2]=k[8]; k6[5,3]=k[10];k6[5,4]=k[6]; k6[5,5]=k[13]

    KE = 1/((nu+1)*(1-2*nu)) *  np.concatenate([np.concatenate([k1   , k2, k3   , k4  ],axis=1),
                                                np.concatenate([k2.T, k5, k6   , k3.T],axis=1),
                                                np.concatenate([k3.T, k6, k5.T, k2.T],axis=1),
                                                np.concatenate([k4   , k3, k2   , k1.T],axis=1)])
    return (KE)

# ----------------------------
# --- Density Filter ---
def df(nelx,nely,nelz,rmin):
    nfilter = int(nelx*nely*nelz*((2*(np.ceil(rmin)-1)+1)**3))
    iH = np.zeros(nfilter)
    jH = np.zeros(nfilter)
    sH = np.zeros(nfilter)
    counter = 0
    for k in range(nelz):
        for i in range(nelx):
            for j in range(nely):
                row = k*nelx*nely + i*nely + j
                k2a = int(np.maximum(k-(np.ceil(rmin)-1),0))
                k2b = int(np.minimum(k+(np.ceil(rmin)),nelz))
                i2a = int(np.maximum(i-(np.ceil(rmin)-1),0))
                i2b = int(np.minimum(i+(np.ceil(rmin)),nelx))
                j2a = int(np.maximum(j-(np.ceil(rmin)-1),0))
                j2b = int(np.minimum(j+(np.ceil(rmin)),nely))
                for k2 in range(k2a,k2b):
                    for i2 in range(i2a,i2b):
                        for j2 in range(j2a,j2b):
                            col = k2*nelx*nely + i2*nely + j2
                            if counter>nfilter-1:
                                iH = np.concatenate((iH,np.array([0])),axis=1)
                                jH = np.concatenate((jH,np.array([0])),axis=1)
                                sH = np.concatenate((sH,np.array([0])),axis=1)
                            iH[counter] = row
                            jH[counter] = col
                            sH[counter] = np.maximum(0.0,rmin-np.sqrt((i-i2)**2+(j-j2)**2+(k-k2)**2))
                            counter = counter+1
    return(sH,iH,jH)

# ----------------------------
# --- Optimality Criterion ---
def oc(nelx,nely,nelz,x,volfrac,dc,dv,g,activetop,castingconstraint):
    l1 = 0
    l2 = 1e9
    move = 0.2
    xnew = np.zeros(nelx*nely*nelz)
    while (l2-l1)/(l1+l2)>1e-3:
        lmid = 0.5*(l2+l1)
        xnew[:] = np.maximum(0.,np.maximum(x-move,np.minimum(1.,np.minimum(x+move,x*np.sqrt(-dc/dv/lmid
     )))))
        if activetop == 1:
            activeele = np.zeros(nelx*nely*nelz)
            activeele[::nely] = 1
            xnew[np.nonzero(activeele)] = 1
        if activetopextra == 1:
            activeele = np.zeros(nelx*nely*nelz)
            activeele[1::nely] = 1
            xnew[np.nonzero(activeele)] = 1
        if castingconstraint == 1:
            xnew = cc(xnew,nelx,nely,nelz)
        gt = g+np.sum((dv*(xnew-x)))
        if gt>0:
            l1 = lmid
        else:
            l2 = lmid
    return(xnew,gt)

# --------------------
# --- Self-weight ---
def sw(xPhys,specweight,ndof,nelx,nely,nelz):
    Fsw = np.zeros((ndof,1))
    # Self-weight corner nodes
    swnidcorner1 = np.sort(np.array([0])*(nelx+1)*(nely+1)+np.array([0])*(nely+1)+(nely+1-np.arange(1,
     nely+1)))
    swdofcorner1 = 3*swnidcorner1.T - 1
    for el in range(nely):
        Fsw[swdofcorner1-1] = specweight*(0.25*xPhys[el])
    swnidcorner2 = np.sort(np.array([0])*(nelx+1)*(nely+1)+np.array([nelx])*(nely+1)+(nely+1-np.arange
     (1,nely+1)))
    swdofcorner2 = 3*swnidcorner2.T - 1
    for el in range(nely):
        Fsw[swdofcorner2-1] = specweight*(0.25*xPhys[(nelx-1)*nely+el])
    swnidcorner3 = np.sort(np.array([nelz])*(nelx+1)*(nely+1)+np.array([0])*(nely+1)+(nely+1-np.arange
     (1,nely+1)))
    swdofcorner3 = 3*swnidcorner3.T - 1
    for el in range(nely):
        Fsw[swdofcorner3-1] = specweight*(0.25*xPhys[nelx*nely*nelz-nelx*nely+el])
    swnidcorner4 = np.sort(np.array([nelz])*(nelx+1)*(nely+1)+np.array([nelx])*(nely+1)+(nely+1-np.
     arange(1,nely+1)))
    swdofcorner4 = 3*swnidcorner4.T - 1
    for el in range(nely):
```

```
        Fsw[swdofcorner4-1] = specweight*(0.25*xPhys[nelx*nely*nelz-nely+el])
    # Self-weight surface nodes
    [j,k] = np.mgrid[1:nely+1,2:nelz+1]
    swnidsurface1 = (k.T-1)*(nely+1)*(nelx+1)+j.T
    swdofsurface1 = 3*swnidsurface1.T - 1
    for elz in range(nelz-1):
        for el in np.arange(elz*(nelx*nely),elz*(nelx*nely)+nely):
            Fsw[swdofsurface1-1] = specweight*(0.25*xPhys[el]+0.25*xPhys[el+nelx*nely])
    [j,i] = np.mgrid[1:nely+1,2:nelx+1]
    swnidsurface2 = (i.T-1)*(nely+1)+j.T
    swdofsurface2 = 3*swnidsurface2.T - 1
    for el in range(nely*(nelx-1)):
        Fsw[swdofsurface2-1] = specweight*(0.25*xPhys[el]+0.25*xPhys[el+nely])
    [j,i] = np.mgrid[1:nely+1,2:nelx+1]
    swnidsurface3 = (i.T-1)*(nely+1)+j.T+(nelx+1)*(nely+1)*nelz
    swdofsurface3 = 3*swnidsurface3.T - 1
    for el in range(nelx*nely*(nelz-1) , nelx*nely*(nelz-1)+nely*(nelx-1)):
        Fsw[swdofsurface3-1] = specweight*(0.25*xPhys[el]+0.25*xPhys[el+nely])
    [j,k] = np.mgrid[1:nely+1,2:nelz+1]
    swnidsurface4 = (k.T-1)*(nely+1)*(nelx+1)+j.T+nelx*(nely+1)
    swdofsurface4 = 3*swnidsurface4.T - 1
    for elz in range(nelz-1):
        for el in np.arange(elz*(nelx*nely)+nely*(nelx-1),elz*(nelx*nely)+nely+nely*(nelx-1)):
            Fsw[swdofsurface4-1] = specweight*(0.25*xPhys[el]+0.25*xPhys[el+nelx*nely])
    # Self-weight inside nodes
    [j,i] = np.mgrid[1:nely+1,2:nelx+1]
    for zlayer in range(nelz-1):
        swnidinside = (i.T-1)*(nely+1)+j.T + (nelx+1)*(nely+1) *(zlayer+1)
        swdofinside = 3*swnidinside.T - 1
        for el in range(nely*(nelx-1)):
            Fsw[swdofinside-1] = specweight*(0.25*xPhys[el+nelx*nely*zlayer]+0.25*xPhys[el+nely+nelx*
     nely*zlayer]+0.25*xPhys[el+nely*nelx+nelx*nely*zlayer]+0.25*xPhys[el+nely*nelx+nely+nelx*nely*
     zlayer])
    return(Fsw)

# ----------------------------
# --- Casting Constraint ---
def cc(xnew,nelx,nely,nelz):
    for col in range(nelx*nelz):
        for el in range(col*nely+nely-1,col*nely,-1):
            if xnew[el] > xnew[el-1]:
                xnew[el-1] = xnew[el]
    return(xnew)

# ------------------------------
# --- Visualization Optimum ---
def displayfigure(xPhys,nelx,nely,nelz):
    xPlot=np.empty(nelx*nely*nelz)
    for ele in range(nelx*nely*nelz):
        if xPhys[ele]>0.5:
            xPlot[ele]=1.
        else:
            xPlot[ele]=0.
    py,px,pz = np.mgrid[0:nely, 0:nelx, 0:nelz]
    fx = px.T.flatten()
    fy = py.T.flatten()
    fz = pz.T.flatten()
    for ele in range(nelx*nely*nelz):
        if xPlot[ele]>0.5:
            point = mlab.points3d(fz[ele],fx[ele],--fy[ele], mode='cube', opacity=xPlot[ele],
     scale_factor=1., color=(1.,1.,1.))
    mlab.show()

# ========================================================================================================
# --- Topology Optimization Main ---
# ========================================================================================================
if __name__ == "__main__":
    main(nelx,nely,nelz,volfrac,penal,rmin)
```