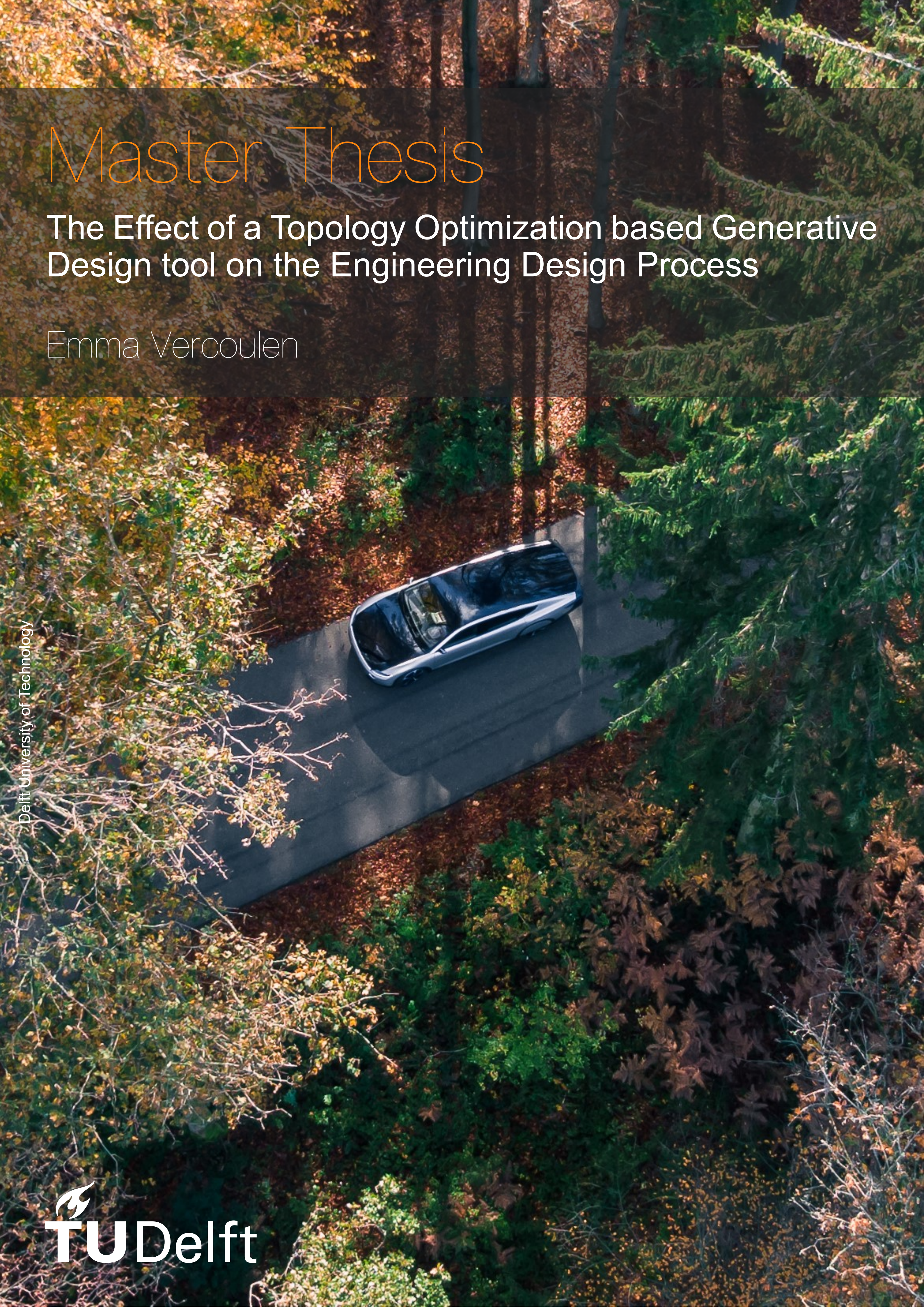
Master Thesis

The Effect of a Topology Optimization based Generative Design tool on the Engineering Design Process

Emma Vercoulen



Delft University of Technology

TU Delft

Master Thesis

The Effect of a Topology Optimization based Generative Design tool on the Engineering Design Process

by

Emma Vercoulen

to obtain the degree of

Master of Science

in Mechanical Engineering

at the Delft University of Technology,

to be defended publicly on 22 June 2023 at 12:00.

Student number:	4389662	
Project duration:	September 2022 – June 2023	
Thesis committee:	Dr. ir. M. Langelaar,	TU Delft, supervisor
	Dr. ir. Y. B. Eisma,	TU Delft, supervisor
	Dr. ir. G. Radaelli,	TU Delft

This thesis is confidential and cannot be made public until further notice

Cover: Lightyear 0 in Autumn

TU Delft

Acknowledgements

I would like to express my gratitude to the following individuals who have played a crucial role in the completion of this thesis:

First of all, I am very grateful to both of my supervisors, Matthijs Langelaar and Yke Bauke Eisma. Your guidance, constructive criticism, and thoughtful suggestions have elevated this thesis and helped me navigate through the complexities of the research process. I could not have wished for better supervisors as you both were always very approachable and willing to help. Your constant encouragement, expertise and insightful feedback have helped me shape this work and pushed me to achieve my best.

I would also like to express my appreciation to Andrea Carpi for his ideas and supervision during the first part of this thesis. This helped to lay a strong foundation for the rest of the project. I am grateful for the intellectual stimulation and discussions that have supported my understanding of the subject and its value in real life applications from an industry perspective.

I would like to thank my family for their love and support, which has helped me through not only this thesis, but all the years of studying at the TU Delft.

Last but not least, I would like to thank my partner for always being there for me, providing me with mental support and understanding.

To everyone who has contributed, directly or indirectly, to the completion of this thesis, I offer my gratitude. Your support, guidance, and encouragement have been invaluable in shaping this work and my personal growth.

Abstract

In the engineering industry, all structural parts have to be designed as efficient and lightweight as possible. Traditionally, the design process has been carried out through manual design iterations, which can be time-consuming and require significant engineering expertise. Over the last decades however, several computational design techniques like Topology Optimization and Generative Design have been developed to support engineers in the structural part design process. Even though these techniques can have a positive influence on the design process, they both also have their downsides. Topology Optimization only gives a single result that is often a local optimum, influenced by boundary conditions and numerical settings. Commercial Generative Design tools explore multiple design options in a single run using AI algorithms, but need cloud-based systems to carry out their demanding simulations which still take several hours per run. It is however expected that a combination of the two, a Topology Optimization based Generative Design approach in the form of an auxiliary tool, has potential to improve the early stages of the design process even more. With such a design approach, multiple design solutions are explored quickly to study the effect of boundary conditions or numerical settings. This can help designers by giving direction and insight in trade-offs between multiple objectives, early on in the design process when design decisions still have the highest impact.

The goal for this research was therefore to research the effect of such a Topology Optimization based Generative Design approach on the design performance and experience. In order to do so, a robust and user-friendly TOP-GD tool was created. In this tool, multiple design solutions are explored quickly by implementing a batch-run setup that varies several chosen parameters, without needing to manually run several optimizations consecutively. Calculations are done with a simple TO script using coarse geometries, and without taking into account manufacturing methods yet. This asks for less demanding, detailed and complicated calculations than AI-based Generative Design tools currently offer, while at the same time moving from a single TO result to generating a range of candidate solutions. A lot of effort was put in the user-friendliness of the TOP-GD tool, enabling an easy workflow for the setup of design problems and a clear presentation of the results by means of a simple GUI.

The use of the TOP-GD tool in the design process was evaluated in an experiment, where it was compared with a more simple TO tool and a basic manual design approach using just pen and paper. This was done by giving the participants of the experiments three simple design assignments, that they had to carry out using each of the design approaches one by one. Evaluation of the approaches was done by comparing the design performance, and assessing the design experience with a survey and using Eye-tracking techniques.

The results of this experiment did not show enough evidence to conclude that the different design approaches had an effect on the design performance for the simple assignments executed during the experiment. However, the results of the survey show a clear positive impact of both the TO tools on the design experience, compared to manually designing. Furthermore, the TOP-GD tool has the largest positive impact on the design experience and its use in the design process is considered a big improvement, especially in quickly exploring new design directions and creating overview. This confirms the expectation that a Topology Optimization based Generative Design approach has a positive effect on the early stages of the design process. The differences found with Eye-tracking between the TO tools support this, although a more extensive experiment should be done to convincingly confirm this conclusion.

Contents

List of Figures

Nomenclature

Abbreviations

Abbreviation	Definition
TO	Topology Optimization
GD	Generative Design
GUI	Graphical User Interface
BESO	Bi-directional Evolutionary Structural Optimization
SIMP	Solid Isotropic Material with Penalization
FE	Finite Element
OC	Optimality Criteria
MMA	Method of Moving Asymptotes
MO	Multi-Objective
AI	Artificial-Intelligence
CAD	Computer-Aided Design
IPOTP	Interior Point Optimizer
TOP-GD tool	Topology Optimization based Generative Design tool
DoF	Degrees of Freedom
HCI	Human-Computer Interaction

1
Introduction

1.1. Motivation, Aim and Approach

In many engineering industries, structural parts have to be designed as efficient as possible. That means that besides being able to withstand certain loads for given boundary conditions, they should also be lightweight. The structural part design process is therefore about obtaining the lightest geometry possible, that can still endure a certain set of loads under given boundary conditions. Traditionally, the design process has been carried out through manual design iterations, which can be time-consuming and require significant engineering expertise. Over the last decades however, several computational design techniques like Topology Optimization (TO) and Generative Design (GD) have been developed to support engineers in the structural part design process [1, 2, 3]. Topology optimization uses algorithms to generate optimal design solutions based on predefined constraints and objectives [4, 5, 6], while generative design uses artificial intelligence to generate multiple design alternatives at once, allowing for the exploration of multiple design options and support designers' creativity [1, 3].

Even though at many companies the preferred way of working is still by manual design, it is expected that a tool using computational design techniques can offer valuable guidance in part design also in this context. Especially at the early stages of the design process, a quick study using Topology Optimization can be used to quickly determine the optimal load path and provide a minimum mass target. Explorative techniques like Generative Design can be used to explore and compare different design options, especially when dealing with multi-objective optimizations by giving insight in the trade-off between stiffer or lighter solutions. This has the potential to improve the final design of structural parts, as well as the overall design process.

Topology Optimization platforms that perform a single simulation and give a single result, already exist. However, this single result is typically a local optimum which is strongly influenced by the starting point of the optimization, the boundary conditions and other numerical settings. A single result therefore gives no information on whether this initial setup gave the best result possible [7]. To explore different solutions or study the effect of the starting point, boundary conditions and numerical settings with such TO software, multiple runs have to be done iteratively. This increases the amount of work needed significantly since every time the optimization has to be set up again with different parameters. On the other hand, commercial Generative Design tools have been developed more recently, which explore multiple design options in a single run by using AI-based algorithms. The largest differences in the designs generated are usually due to the evaluation of different manufacturing methods or materials. However, these tools use cloud-based systems to carry out their demanding simulations, still take several hours per run and only produce a percentage of actually converged or usable results [3]. Moreover, although a GD tool can help with creating overview by presenting the generated solutions in different plots, it is still not an easy or trivial task for the user to select the 'best' result when dealing with multi-objective problems.

What is missing however, is something in between: a user-friendly tool that quickly helps to explore multiple design solutions, and study the effect of boundary conditions, numerical settings or starting points on the optimization problem. This can be done without needing to manually run several optimizations consecutively, by implementing a batch-run setup that varies several chosen parameters, through a user-friendly Graphical User Interface (GUI). Calculations can be done using coarse geometries and without taking into account manufacturing methods yet, to support the designers in the earliest stages of the part design process. This can help give direction earlier on, and give insight in trade-offs between multiple objectives. Gathering as much information as fast as possible enables early changes of the design, which is crucial to both the efficiency and performance of a design, and the cost of the final designed product [2, 8, 9]. This asks for less demanding, detailed and complicated calculations than AI-based Generative Design tools currently offer, while at the same time moving from a single TO result to generating a range of candidate solutions. The goal for the thesis is therefore to research whether the design process can be improved by an auxiliary tool using a combination of computational design techniques like described above. The main research question of this thesis is:

"What is the effect of using a Topology Optimization based Generative Design tool or a simple Topology Optimization tool compared to manual design on the design performance and experience?"

Instead of generally speaking about the design process, the design performance and experience are consciously separated in the main research question. It could be possible that in simple assignments, the effect of the proposed approach is less noticeable on the "performance" of the designed parts (e.g. its weight, compliance or maximum stress value), but does however improve the design experience of the engineer using the tool. This would still be a positive impact on the overall design process.

In order to answer the main research question, a working and robust user-friendly Topology Optimization based Generative Design tool has to be created and its potential has to be tested. The tool has to be able to perform multiple topology optimization runs in batches to explore the design space and the influence of boundary conditions and parameter settings. In combination with a GUI, the control of such a batch-run optimization has to be made possible. In order for the tool to have a positive impact on the design process, it should be easy for the designer to set up the problem, control the optimization and interpret the results. To make all these functionalities possible, a big part of the time set out for this thesis will be spent on the development of the proposed tool. After the creation of this tool, the main research of this thesis can be done. This will be focused on the influence of the tool on the design performance and experience. To be able to review this, it is interesting to do a comparison study of the design process with different approaches: designing manually without any aid of computational design techniques, using a simple 'single run' TO tool to design, or designing with the new proposed approach using a topology optimization based generative design tool. In this way it can be researched by means of an experiment what the effect of the different approaches is on both the design performance and experience, and how they compare to each other.

1.2. Scope

For the rest of the Thesis project, the scope is limited to researching and applying computational design techniques in the context of:

- 3D parts
- structural static load cases
- the Linear Elastic regime
- isotropic materials

To test the effect of a Topology Optimization based Generative Design tool on the design process, it has to be able to perform realistic 3D design assignments. Dynamic load-cases, non-linear deformations and anisotropic materials can potentially be implemented in future research, but are not of interest in this thesis. Moreover, different manufacturing methods do not have to be considered by this tool during the design process. As will be further elaborated in Section 2.1, manufacturing constraints are not of high importance yet in the earliest design stages.
The target group for this tool consists of Mechanical and Structural Engineers. It can be assumed that they have general engineering knowledge, but no specific knowledge on TO, GD or other computational design techniques.

1.3. Structure of the report

To arrive to this context, research question and project proposal, a literature research has been done prior to the thesis. In this literature review [10], both the design process as well as different computational design techniques have been reviewed to identify the above explained gap. Besides that, different software tools have been tested to see whether they would be usable for the development of a Topology Optimization based Generative Design tool within the scope of this thesis. The most important and relevant findings of this literature review are summarized in the next chapter, to make the thesis independently readable and give the necessary background information.

In Chapter 3, first the requirements for the tool are listed, followed by a description of its development process while explaining all functionalities. In this description, the GUI is presented and the workflow for the user is demonstrated as well. Next in Chapter 4, the methods for the performed experiments are discussed. The results of the experiments and the created tool are presented in Chapter 5. This is followed by a discussion of the results in Chapter 6. Lastly, the most important findings of the experiment and thesis are presented in the conclusion, followed by recommendations for future research.

2
Literature Review

To identify where there is most potential for computational design techniques to improve the design process, it is first important to describe what the engineering design process actually entails. This is followed by a section about the basics of Topology Optimization, focused only on the approach used in this thesis, with extra added details about the different parameters of importance. In Section 2.3, two different types of design exploration methods relevant for this thesis are explained. Lastly, different existing software tools are reviewed and one is selected to develop a Topology Optimization based Generative Design tool with.

2.1. The Design Process

The engineering design process consists of a series of stages or steps that are used to create new products or parts. Although there is no standard definition of what the specific steps entail, resemblances are found among all the variations. A literature comparison indicates that the engineering design process can generally be distributed into four steps: [11, 12]

1. Problem definition
2. Conceptualization
3. Preliminary / Prototype design and evaluation
4. Detailed design and evaluation

After finishing these stages, the production process can be prepared and started. However, the design process is highly iterative since it frequently involves a repetition of several steps as a result of insights later on in the process. In general, design modifications brought about by these iterations become more and more expensive to realize as the design process progresses [11].

Moreover, in many engineering companies, engineers are working on a very tight schedule. This makes it hard to do many design and testing iterations. This emphasizes the importance of a well chosen concept to work out in the following design process steps, and to be sure the concept is a good solution as early as possible. This will reduce the number of iterations needed later in the process to improve the design. Therefore, it is expected that the conceptual or early design stage has the most room for improvement within the design process.

This aligns with other remarks found in literature. The conceptual design stage is said to be key to both the efficiency of a design and the cost of the final designed product [2]. At the start of the design process, design decisions still have a big impact. But as the design develops, their impact reduces rapidly [9]. Consequently, the conceptual and preliminary design stages offer the largest window of opportunity to improve the process and the design, as the concept that is chosen to further develop throughout these stages has a large influence on the core features of the final design. It is quite difficult to make up for an inferior concept choice during the subsequent detailed design stage [9]. That is why it is important to collect as much information about the product as soon as possible in the design process, to prevent a poor concept choice by enabling early changes of the design [8].

2.2. Topology Optimization

Topology optimization is a form of structural optimization, where the goal is to determine the optimal layout of a structure within a predefined domain [6]. During this optimization process, the topology of the design is completely flexible. This means there are no prior assumptions about the shape or topology of that structure. Only the applied loads, the boundary conditions and the volume of the structure are defined before starting the optimization [6].

In topology optimization, the problem is generally formulated to minimize an objective function $F(x)$, by finding the optimal material distribution within a specified design domain. The objective can be the minimization of compliance, mass or stress. x is then the set of design variables, representing the distribution of material for which the optimal values have to be found. To ensure realistic results, the optimization problem is subjected to a set of (in)equality constraints, for example to ensure a nonzero volume [13, 14].

The principle of topology optimization was introduced for the first time by Bendøe and Kikuchi in 1988 [4]. In this paper, a homogenization approach is used that varies the micro structure of discretized elements to optimize the performance of the overall structure. After this first publication, topology optimization has developed enormously in a lot of different directions [14]. Multiple topology optimization approaches exist, such as the Level-set Method [15, 16] and the Bi-directional Evolutionary Structural Optimization (BESO) method [17, 18, 19]. In principle, any TO method could be used for this thesis. However, only the conventional density-based Solid Isotropic Material with Penalization (SIMP) method has been used, due to the software choice explained in Section 2.5. The principles of the SIMP method are described in the next section.

2.2.1. SIMP

The SIMP method is one of the most recognized and commonly used density-based methods. In density-based methods, the design domain is discretized into many small, finite elements in order to solve the fundamental topology optimization problem [7]. Each of these elements get assigned a density value, which all together are the design variables. An element with a density value of 1 represents a solid material element, and an element with a value of 0 density represents a void element. The material distribution described by all these element density values is optimized to minimize the objective function [13, 14].

Treating a problem with this discrete approach however, is very computationally expensive [20]. In the SIMP method, this is solved by using continuous variables for the densities of the elements instead. Having continuous density values allows for sensitivity analysis and more efficient gradient-based optimization. However, this regularization also means that elements can have an intermediate value between 0 and 1 for the density, which is physically hard to interpret and impossible to manufacture. The SIMP approach therefore penalizes these intermediate density values, which makes them unattractive for the optimization. This forces the design to a more distinct solid-void solution. The penalization is done by using a power-law to define the relationship between the elastic properties and the element density with the following formula [13]:

$$E\left(\rho_e\right) = \rho_e^p E_0, \quad p \geq 1 \tag{2.1}$$

with ρ_e being the element density, p as penalization parameter and E_0 is the Young's modulus of the solid material. For $p \geq 1$, intermediate densities get penalized, which makes the optimization algorithm favor clear solid-void solutions. This penalization effect only works if there is some volume constraint present [14].

Generally speaking, topology optimization algorithms like the density based SIMP method follow the same series of steps to get to a useful result, shown in the scheme in Figure 2.1 below.

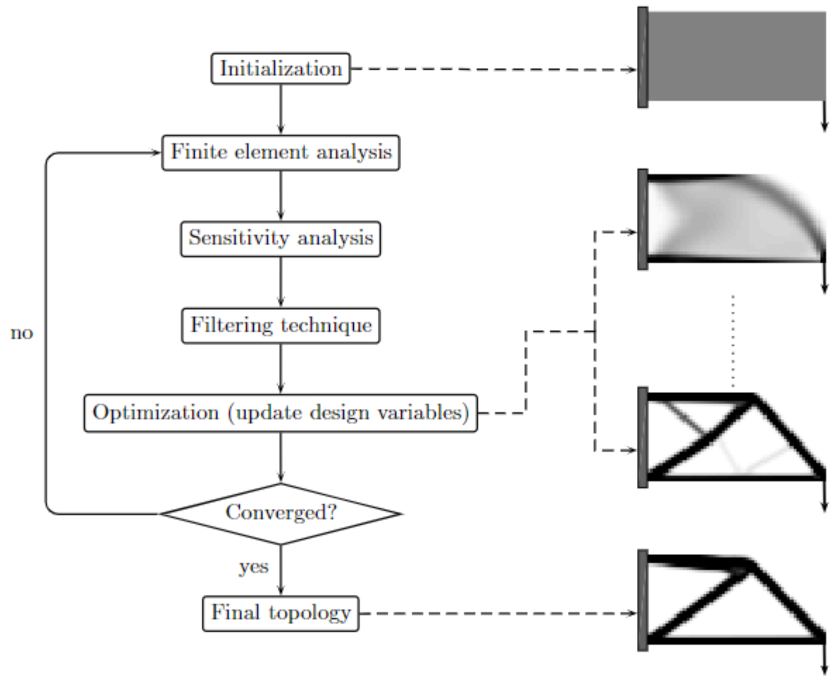


Figure 2.1: The general Topology Optimization Scheme [20]

For clarity, the order of steps shown here will be followed in the coming sections to explain the basic principles of topology optimization. First, the structural problem is initialized by setting up the geometry, the finite element (FE) mesh, the loads and boundary conditions and initializing the density distribution [20]. At this point, the optimization loop can be started where the equilibrium equations are assembled and solved using Finite Element Analysis. This is followed by a Sensitivity Analysis, applying a filter and updating the design variables in the optimization step. The updated design variables are consequently checked for convergence to decide whether another loop is started or the result has converged. What happens during these last mentioned steps is described in the following sections.

2.2.2. Sensitivity Analysis

In an optimization problem, the displacement field is a function of the design variables. By calculating the derivatives of the displacements, or other structural performance quantities, with respect to the continuous design variables, a sensitivity analysis can be carried out. A sensitivity analysis can be used to understand the effect of changing each design variable. In density-based methods like SIMP, an element's sensitivity corresponds to the change in the structure's overall compliance when this element is removed [21]. This indicates the effect of a change in density. This information is used later in the optimization step to update the design variables in each iteration, based on the elemental sensitivity values. In this way, the most efficient elements will stay in the structure, while the elements that do not have a large influence on the total compliance of the structure, get removed. Gradient-based optimization makes use of this sensitivity information to update the design. In addition, when this information is visualized, a sensitivity analysis can provide a designer valuable insights on the impact of specific design variables or particular constraints on the objective(s).

In Topology Optimization a problem usually includes a large amount of design variables that need to be considered. Therefore, the adjoint method is most effective for doing a sensitivity analysis. The derivatives of the displacement are not explicitly calculated in this method [6].

2.2.3. Filtering

Before the design variables can be updated, first a filtering technique should be applied to the obtained sensitivity field. This is because the obtained solutions are dependent on the level of mesh refinement when using the SIMP method, which can be problematic. When applying the same loads and boundary conditions, an increased mesh density results in a different and more detailed solution with more members of a smaller size, compared to a coarse mesh [22]. This is illustrated in Figure 2.2.

Figure 2.2: Mesh refinement dependency for the optimal topology. Solutions for a discretization with: 1350, 2400 and 8600 elements [6]

This mesh dependency effect can be decreased by applying a blurring filter on the density or sensitivity field to smooth out the values. This removes patterns with fine details, and only leaves the core features of the design. This is a highly efficient method to achieve mesh-independency [6]. The filtering is done per element with a distanced-weighted averaging of the sensitivities (calculated in the previous step) within a certain region around that element. The sensitivities of the elements in close proximity of the concerned element therefore contribute more than the values of the elements on the outer edge of the filtered area. The value for the filter radius determines the size of the total filtered area, and how many elements are included in the averaging for each element. Therefore, this also controls the size of the features that are maintained in the solution. A larger filter radius results in a larger minimum member size. This gives less detailed solutions, but often gives a smoother and more realistic overall result.

2.2.4. Optimization Methods

After a filter has been applied on the sensitivity field, the design variables can be updated in the optimization step. Usually, topology optimization problems are large-scale nonlinear optimization problems that can be handled using a large variety of different numerical optimization approaches [23]. As was previously stated, a large amount of design variables need to be taken into account, for which gradient-based methods converge much faster than non-gradient-based methods. In topology optimization, therefore the two most used methods are the Optimality Criteria (OC) [24, 25] and the Method of Moving Asymptotes (MMA) [26, 27].

OC methods are very effective in solving optimization problems with few constraints in comparison to the number of design variables. Especially when dealing with just one constraint, like in a standard compliance minimization problem with a single volume constraint, the OC algorithm is very useful [28]. The reason why OC is an effective algorithm, is because each design variable is updated independently of the others in every iteration. This is done by computing the Langrange multipliers for the (active) constraints in every iteration, which are then used with the gradients to update the design variables in such a way that they satisfy the optimality conditions obtained from the Lagrangian [6].

Compared to OC methods, the Method of Moving Asymptotes is more versatile. When problems become more complicated, large scale and with multiple constraints, MMA has better convergence properties. For simple compliance minimization problems as considered in this thesis however, the OC method is the best choice [6].

2.2.5. Convergence Check

After the design variables have been updated, the updated density values can be compared with the design variables from the previous iteration. If these values differ a lot, it means that a lot of change has still been implemented in the last iteration step, and that it is useful to repeat the optimization loop once more. As shown in Figure 2.1, the loop will start again from the FEA step. However if the value for the change in the design variables is below a certain threshold, it means that the updated topology differs minimally from the previous iteration and has converged. The optimization loop can be exited and the final topology for this optimization run has been determined.

In 2001, a simple 99-line code written in Matlab was presented using the SIMP method [29], which follows the optimization scheme shown in Figure 2.1 as well. This code formed the basis for many other codes and improvements. In this code, the optimization loop can be terminated in two ways. A threshold ”change” parameter is set, which terminates the optimization loop as described above when the topology has converged. However if for example time is limited, the optimization loop can also be terminated by setting a maximum amount of iterations. If after these amount of iterations the topology has still not converged according to the set change threshold, the optimization loop will terminate anyway.

2.3. Design Exploration

As was explained in Section 2.1, a single TO run does not contribute to the exploration of different solutions that are comparably optimal but vary geometrically, especially in multi-objective optimizations [1]. Therefore it is also interesting to look at Pareto solution sets and the use of computational design techniques like Generative Design during the conceptual design stage. Both will be discussed in the sections below.

2.3.1. Multi-objective Optimization

Objective functions in algorithms are typically presented as minimization functions for the provided design parameters. In the simplest case, an optimization method has just one objective to guide the design process. Nonetheless, it commonly occurs that an engineer needs to take multiple objectives into account while defining a problem. These Multi-objective (MO) problems are more challenging to solve, especially when the different objectives are conflicting. Instead of one unique optimal solution, this typically results in multiple different solutions that each meet the requirements in their own way. When various objectives or criteria have to be considered, Multi-objective optimization is a method to find solutions. In structural design problems where different criteria such as minimizing weight and maximizing stiffness are frequently in conflict, this approach is quite useful [1, 30].

The aim of a MO optimization is to simultaneously consider all the criteria and find the best trade-off solutions with respect to the relevant objectives. These solutions will inevitably improve in one or more aspects, but worsen in others [30, 1]. By collecting a set of these equal trade-off solutions that cannot improve with respect to any of the objectives without compromising another, a Pareto-optimal front is obtained in the objective space [32]. An example of the Pareto frontier is depicted in Figure 2.3. Here, f1 and f2 stand for two objectives. The minimum values for both these individual objectives combined provide the ”Utopia” point. To find the closest feasible Pareto frontier point, the minimum distance criterion is used. This point is denoted with ”UPF” in the figure [31].

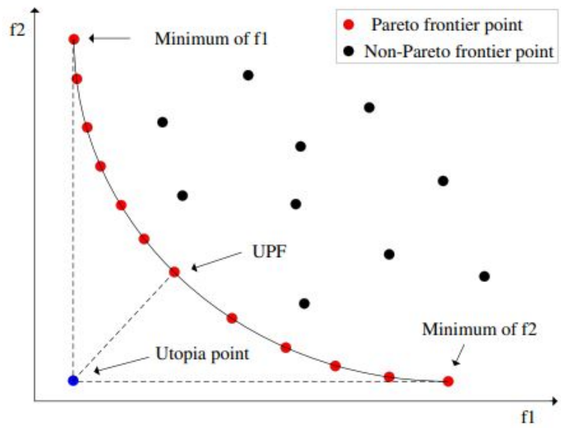


Figure 2.3: The Pareto frontier [31]

When the exploration of multiple designs is desired, generating such a Pareto set offers numerous advantages. The Pareto set helps the designer to make an informed decision by providing a variety of solutions that are all optimal from an ”overall” point of view. In a single-objective optimization, this trade-off viewpoint is likely to be overlooked. Moreover, a Multi-objective approach is also helpful in understanding and exploring the consequences of a design decision with respect to all the relevant objectives considered [30].

There are multiple ways to explore the Pareto-optimal front, but one of the simplest ways to find different optimized results is to iterate optimization runs while altering some optimization settings such as the volume fraction, target objectives or material parameters [1].

2.3.2. Generative Design

Generative Design is a broad term used in numerous fields and applications, with no clear definition [1]. In some literature, the terms Generative Design and Topology Optimization are occasionally even used interchangeably. In the context of heat conduction, Lohan for example calls the SIMP method already a ”Generative Design Algorithm”, because parameters get evolved parameters over time and a (single) design s generated [33]. It is therefore important to clarify that in this thesis, the term Generative Design is used for methods to create not one, but multiple designs.

There are several methods for doing this. Traditionally, only a small number of parameters concerning the geometry or the problem definition were modified in generative design, but this results in a set of solutions with limited diversity [21, 34]. Moreover, Topology Optimization algorithms can be used to generate different designs by doing multiple TO runs consecutively. This means that for example the

Multi-objective optimization approach described in the previous section, is a form of Generative Design as well.

However, the most common form of Generative Design described in literature, are the tools or approaches that employ Artificial-Intelligence (AI) techniques to generate a set of different design options, while respecting the provided objectives and constraints. Therefore, the rest of this section is focused on these AI-based Generative Design tools. The key difference of these techniques compared to TO-based generative design is the ”first level” generation of a set of multiple solutions simultaneously, which subsequently can be explored [3].

Only recently a number of commercial Computer-Aided Design (CAD) software programs were able to add generative design modules using AI, due to the substantial rise of available computing power [1, 35]. Compared to traditional designing methods, the advantage of such AI-based Generative Design tools lies especially in the fact that it can propose a variety of ’out of the box’ design possibilities which otherwise would not have been considered by the designer [35].

Examples of AI-techniques that are used in Generative Design tools are neural networks and genetic algorithms [35]. Unfortunately, both methods require a lot of computing power to produce flexible designs of high quality. Besides that, some systems do not integrate a mechanical analysis in the generative design process, making it hard to guarantee the engineering performance of the generated parts [21].

Nonetheless, a few GD tools included in commercial CAD software run simulations on a cloud-based platform, enabling demanding GD studies even when the designer works with a limited computer. After setting up the design problem, the tool traverses the design space while generating many optimized geometries [35]. After the cloud-based GD study is finished, the found geometries are presented to the designer. In Figure 2.4 a series of solutions is shown that are generated by Autodesk’s Fusion 360 Generative Design tool [36].

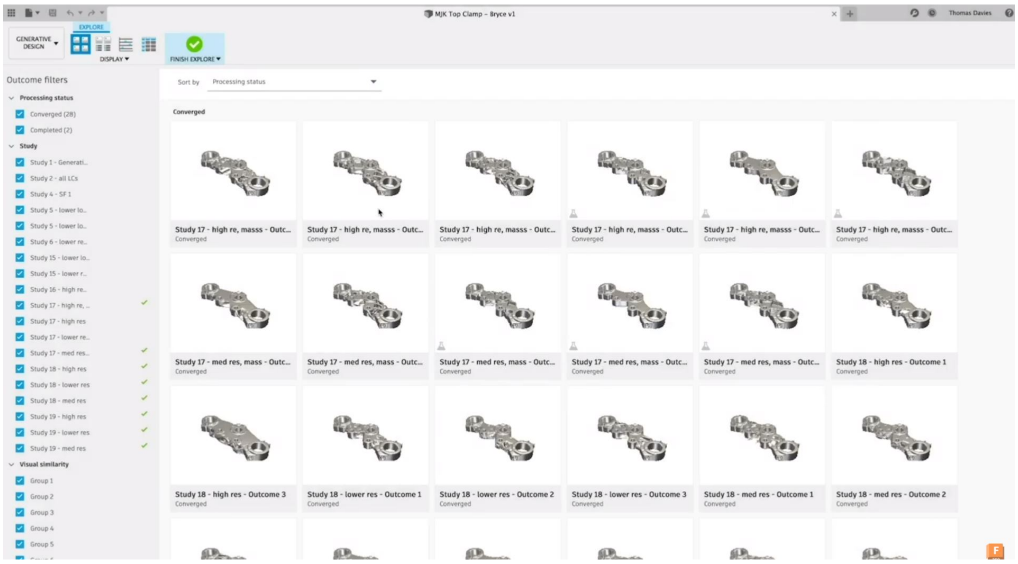


Figure 2.4: Screenshot from a video by Autodesk Fusion 360 Generative Design tool presenting results [36]

Tools like these help designers to identify the best concept for their study, by enabling comparison and trade-off studies based on specified performance indicators [35]. In Figure 2.5 below a plot by Autodesk’s Fusion 360 Generative Design tool can be seen, showing the mass vs. the max displacement of various solutions generated. This plot, which is actually a type of Pareto plot, demonstrates how these commercial GD tools also apply the MO optimization principles like discussed in Section 2.3.1 to support trade-off studies.

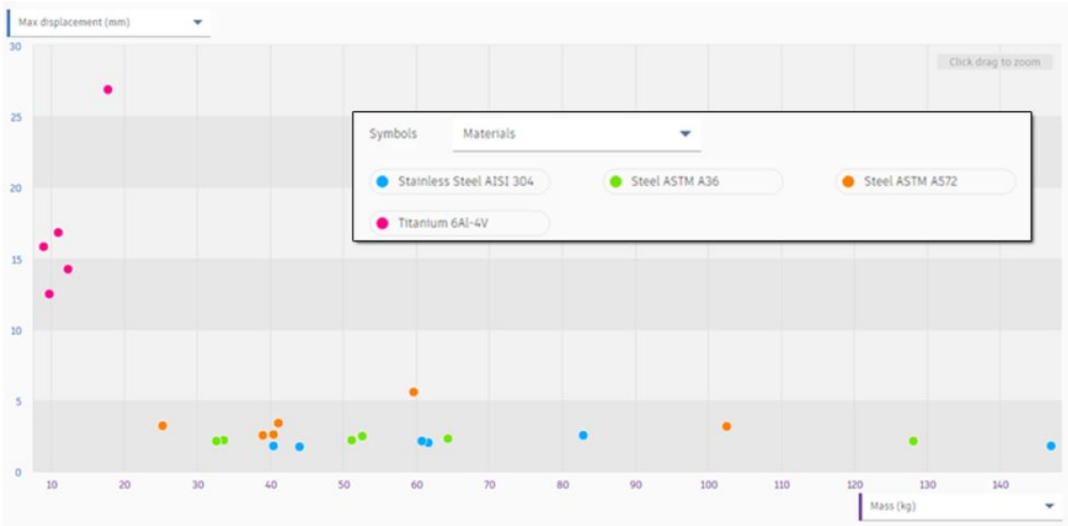


Figure 2.5: Plots from an explorative study done by Buonamici et al. in the Autodesk Fusion 360 Generative Design tool [3]

Commercial AI-based GD tools are excellent at fostering creativity when it comes to design exploration. However, even though cloud-based systems are used to run these substantial simulations, it still takes several hours for a simulation to complete. A topology optimization run only needs a few minutes for the same design problem. The GD tool does provide a lot more diverse solutions during its processing time, but many of them are less optimal and of worse quality than those that more advanced TO tools can generate [1]. The significant computational processing power required and the lesser quality of results therefore make these AI-based Generative Design tools less favourable for the tool required for this thesis. Furthermore, within the time limits of this thesis project, it is extremely challenging to build an AI-based Generative Design tool from scratch, without using the commercially available tools. However, the way AI-based Generative Design tools present multiple solutions to the designer and show a Pareto plot for better comparison, can be used in a Topology Optimization based Generative Design approach as well.

2.4. Computational Design Techniques in Research Approach

Having more information about the design process, Topology Optimization and useful Design Exploration techniques, the research aim and approach can be further specified. To improve the overall design process, the focus for this thesis is set on improving the early design stages. Computational design techniques are expected to support the early design stages, in several ways. The use of Topology Optimization during conceptual design can give beneficial guidance in part design by determining the optimal load path, and giving a minimum weight potential for the given loads and the design space available [1]. This can serve as a mass target later on during the preliminary or detailed design stages, which gives information on whether a design is more or less converged. With that, a better estimation can be made on whether there is still potential for significant mass reduction or the iterating process can be stopped, which can save valuable time from engineers.

Traditionally however, topology optimization only provides a single optimal solution, that is typically a local optimum as well [3]. Consequently, a single TO run does not contribute to the exploration of different solutions that are comparably optimal but vary geometrically, especially in multi-objective optimizations [1]. Therefore it is also interesting to use a computational design technique like Generative Design during the conceptual design stage, and look at Pareto solution sets. When implementing Generative Design, alternative solutions can be generated that the designer did not think of or consider with one single TO run, while these could possibly be better solutions for the design problem [1, 3]. Assessing as many alternative solutions as possible in an early stage of the design process also helps in making a more confident final decision on which design to continue with.

Moreover, it is important to understand that in topology optimization, the defined boundary conditions and numerical settings also influence the solution. Besides that, most topology optimization methods still rely on their starting points [7]. One single run does therefore not give any information on this influence of starting guesses, physical and numerical parameters, and whether by varying these a better overall solution can be found. Especially when a designer is unfamiliar with topology optimization and its settings, it can be helpful if a tool using computational design techniques provides understanding on the effects of these settings and boundary conditions. When doing more topology optimization runs consecutively, a range of numerical and physical parameters can be explored, which results in a set of data points that describe different parameter combinations. This can be used to understand the macroscopic behavior of the optimization problem, and the influence of numerical settings or boundary conditions on the design problem. In this way the designer not only gets information and understanding of the part itself, but it also makes topology optimization less of a ”black box” approach.

Recapitulating that, the conceptual design stage offers plenty of opportunities for improvement. The goal for the thesis project is therefore to utilize computational design techniques in the form of an auxiliary tool, to quickly explore a variety of concepts at the very start of the design process in order to provide guidance and inspiration. Manufacturing constraints are not of high importance yet at this early stage and can be considered later in the design process. This also simplifies and speeds up the optimization runs. By quickly exploring multiple solutions, it is possible to obtain not only one single point solution like in TO, but also a notion of how good solutions are distributed or how they relate to each other statistically. This will give guidance and supports exploration, understanding and making well-founded and unbiased decisions. Potentially, this will contribute to the improvement of the conceptual design stage, and with that the overall design process.

It is however very important that the tool to be developed is practical and user-friendly. If the tool gets too complicated or time-consuming to run, it will most probably not get used by designers during the conceptual design stage at all. This means that for example ’quick and dirty’ TO results are preferred during the conceptual design stage, as it is about quickly exploring design directions and giving an idea of the ideal load path and a rough minimum weight. It is not necessary to run optimizations with high resolution for that, and a short computational time better serves the engineers during this stage. Besides that, a simple workflow that does not need a lot of implementation time and can be easily learned by new users is highly preferable. For a tool, simplicity both in functionality and usability is very important. Even though the scope of a tool may be more limited because of that, it will be easier to use and more likely to get adopted by users. Additionally, simpler tools are usually more robust [37].

2.5. Software Choice

After reviewing multiple Topology Optimization and Generative Design approaches and their applications, a gap has been identified in a simple and quick exploration approach of multiple solutions. As explained in the previous section and introduction, this has been translated into a Generative Design tool that does not rely on AI and cloud-based systems, but uses multiple Topology Optimization runs with different settings to explore the design space and give insight in the trade-off between multiple objectives. For the development of such a tool, suitable software had to be found. Therefore, different software programs that use topology optimization or generative design techniques have been searched and evaluated on their potential to be used or extended for the creation of the tool for this thesis. At the basis for this was a comparative study of Tyflopoulos and Steinert that was recently published and looked at the application of different commercial and open source software for Topology Optimization [38]. Open-source software has the preference, to save costs and enable wider usage. Two software platforms with the most potential, that were also included in the library of the Tyflopoulos and Steinert paper [38], have been reviewed more extensively at the start of this thesis. These platforms can potentially be used as basis for a tool, but need to be extended to provide the necessary functionalities. Besides these existing platforms, the new generation 3D version of the 99-line code in MATLAB [29], called top3D125 [39] is also considered as an option to build a tool with in MATLAB from scratch, and is included in the comparison.

2.5.1. Z88Arion

The first platform reviewed further was Z88Arion [40], which is a complete desktop app that already includes a GUI as well. Although Z88Arion is free to use and has great capabilities in terms of pre- and post-processing and user-friendliness during the setup of the problem, the source code has not been made available. This makes the software hard to edit or extend. Correspondence with the makers of the software unfortunately did not result in any easy option to automate the workflow of the software either, which made the use of Z88Arion less favourable for this thesis.

2.5.2. Toptimiz3D

The second software platform that seemed promising was Toptimiz3D, which is a python-coded GUI working on Linux, that solves TO problems with the C++ open-source FreeFem++ software. The GUI allows the designer to set up the topology optimization problem in a user-friendly environment. All relevant data for defining a TO problem can be specified in the application, different options for solving can be selected and it can be solved directly from the interface. The software uses the SIMP method for compliance minimization problems, with three different optimization methods: MMA, Interior Point Optimizer (IPOPT) and OC. For finite element analysis, the MFEM library is used [41]. A screenshot of the Toptimiz3D GUI is shown in Figure 2.6 below.

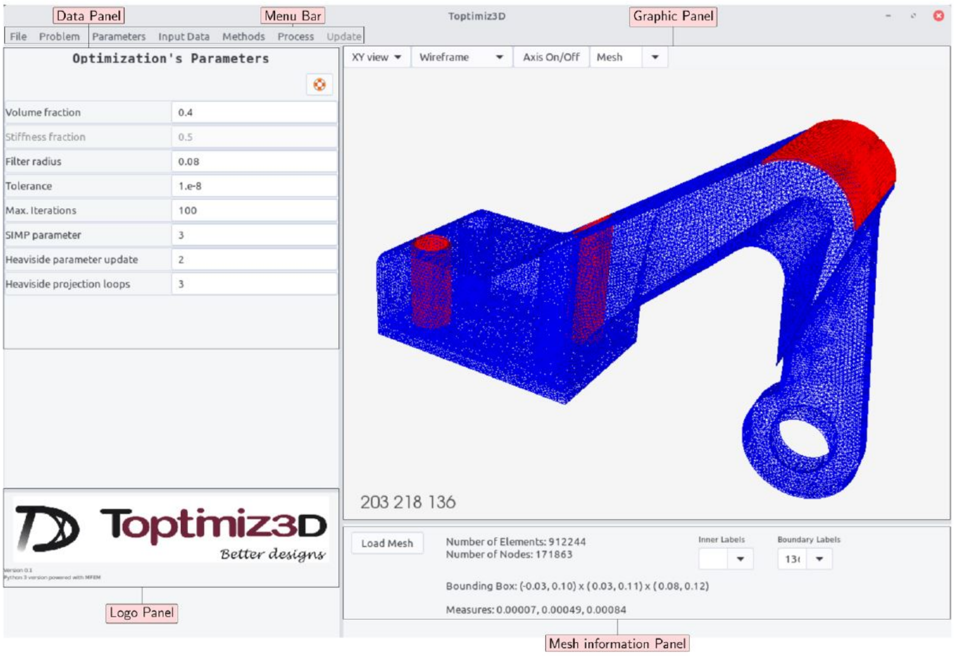


Figure 2.6: GUI available in Toptimiz3D [41]

The software is not able to generate meshes itself, so a suitable mesh has to be build in an external tool and imported in Toptimiz3D, which complicates the workflow. When the mesh has been imported and loaded, it can be visualized and moved in the graphic panel of the GUI. After all information is entered by the user in this python coded interface, a C++ code gets generated, compiled and executed. This makes a relatively fast solving speed possible, which is an advantage when wanting to multiple runs consecutively. During the optimization run, a separate window pops up showing the result of each iteration. When the optimization is completely finished the final design is shown in the graphic panel of the GUI again. Density, stress and deformed configuration plots are available for reviewing. The software is then able to export the results as well, however only in VTK format for post-processing with ParaView [41]. Before the result can be used by a CAD program it therefore first has to be converted in another external program to for example .STL format.
The source code of Toptimiz3D has been made available on GitLab (`https://gitlab.com/e-aranda/topt-mfem#ipopt`), and although it seemed a promising platform for this thesis, testing and installing was more complicated than expected. First of all, Linux was installed, followed by the installation of Toptimiz3D. However, the manual provided on GitLab was missing a lot of crucial installation information and lacked clear and user-friendly instructions. Moreover, the installation did not seem very robust as different errors kept popping up.

After finishing the installation however, the program could be tried out further and compared with the more simple MATLAB code approach. For the same amount of design variables, an optimization run in Toptimiz3D was about 10 times faster than the top3D125 code in MATLAB. However, its biggest downsides turned out to be editability and a complicated workflow. Even though the GUI provided was made with the wxPython GUI toolkit, editing the layout of the GUI would require so many changes at the basis, that it would probably be easier to build a new GUI from scratch. Especially with limited experience in Python and having the difficulties during the installation in mind, this is not the most user-friendly environment to build, edit or share a tool. Therefore, it is very probable that a lot of implementation time will be needed to edit the existing tool which leaves less time for developing new functionalities. Besides that, the fact that for both meshing and post-processing an external program has to be used next to Toptimiz3D when working with an .STL file as input and output format, makes the workflow more cumbersome as well. It has the preference to provide all these functionalities in one tool, to make the workflow as user-friendly as possible.

2.5.3. MATLAB

Therefore, the approach of using the top3D125 [39] code as a basis in MATLAB to build a tool from scratch, was also considered more seriously. This code is a simple 3D compliance minimization algorithm using the SIMP method, which follows the same steps as shown in Figure 2.1. Together with its predecessors, the 99-line code [29] and the 88-line code [42], there is moreover a lot of documentation on how the individual pieces of code in the algorithm work or can be extended. Starting from scratch to build a GUI was first considered less favorable. However, this would also allow to start at an easy level and progress in complexity of the tool during the project. Compared to dealing with complete and complex GUI right from the start with less documentation available, this is probably more efficient.

To design a tool, MATLAB has an app development environment called ”App Designer”, which provides a user-friendly graphical interface. This drag-and-drop interface allows the creation of a visual representation of the tool’s functionality, which can help to quickly iterate on the design and make changes as needed. Matlab App Designer includes a large number of pre-built components and libraries that can be used to add functionalities to the tool. These components range from basic buttons and sliders to more advanced visualization tools and data analysis functions. This can save time and effort in building the tool’s functionalities, and helps to ensure that the tool becomes robust and reliable. Besides that, MATLAB is a widely used tool for scientific computing, and has a large community of users and developers who have created lots of resources and documentation for both the programming language, and the App Designer environment. When working under time constraints, this can make it easier and faster to develop a tool. The fact that MATLAB is used widely in the scientific community also improves the conditions for sharing or distributing the tool. Using Matlab App Designer ensures that it runs correctly on other systems, is therefore easily accessible and can be used to its full potential, in contrast to the complicated, timely and not robust installation of for example Toptimiz3D on Linux. On a more personal note, the exisiting familiarity with MATLAB also lowers learning and implementation time which leaves more time for the actual development of new functionalities in a tool.

The top3D125 code only provides simple compliance minimization functionalities, and is quite a bit slower in solving time compared to Toptimiz3D. However, when using simple parts and coarse geometries as is the intended functionality of this tool in the early design stages, this does not outweigh the advantages of using MATLAB as the development platform for the tool. Moreover, MATLAB has the ability to both import and export .STL files, and generate meshes without needing any other external software. This makes it possible to provide all these functionalities within the tool when using MATLAB, which makes the workflow a lot more user-friendly than if Toptimiz3D were to be used together with multiple external programs. As explained in Section 2.1, a user-friendly workflow is important as it makes it more likely that a tool will actually be implemented by designers and have a positive impact on the design process.

Altogether, it was therefore decided to use MATLAB and its App Designer environment as the main platform in this this thesis for the creation of a new tool, from scratch. In this tool, the top3D125 code [39] forms the basis for solving compliance minimization problems.

3

TOP-GD tool

3.1. Introduction and Requirements

As was explained in the introduction and substantiated in the Literature Review, a new approach is proposed using an auxiliary Topology Optimization based Generative Design tool in the early stages of the design process. This tool should enable the quick exploration of multiple design solutions, and study the effect of boundary conditions and numerical settings. This can be accomplished by implementing a batch-run setup that varies several chosen parameters through a user-friendly GUI, rather than having to manually run several optimizations consecutively or make use of demanding AI-based calculations. This can help give direction earlier on, and give insight in trade-offs between multiple objectives. Potentially, this has a large positive impact on the design performance and experience. In order to evaluate this potential in an experiment, a robust and working product has to be tested, which emphasizes the importance of spending quite some time set out for this thesis on the development of the TOP-GD tool and implementing the proposed functionalities. Moreover, the tool should be practical, simple and user-friendly, while still being able to deal with realistic structural problems. A user should be able to setup and edit a problem in the tool in an easy and quick manner, and an exploration run should not take too long. This makes it more likely that the tool will get adopted by users. Therefore, requirements for the functionalities have been set up prior to the development of the TOP-GD tool, which are summarized below. The tool should be able to:

- import and export 3D .STL geometry files
- enable a simple setup process with a GUI
- deal with multiple load cases
- solve compliance minimization problems
- give stress information
- evaluate multiple material options
- evaluate ranges of input parameters
- generate a clear overview of the results

With these requirements and the important user-friendliness aspect in mind, a Topology Optimization based Generative Design (TOP-GD) tool was developed in MATLAB App Designer. A simple TO script formed the starting point for the tool, adding more functionalities step by step. The following section will describe its development process and present the final GUI of this tool.

3.2. Development Process TOP-GD tool

The starting point for the development process was the compact 3D extension of the compliance topology optimization code written in MATLAB by Ferrari and Sigmund in 2020, top3D125 [39]. This code is a function containing 125 lines, and is a successor of the well known 99-line [29] and 88-line [42] MATLAB codes. The complete MATLAB code can be downloaded from the website `https:`

//www.topopt.mek.dtu.dk/apps-and-software. A schematic overview of the working principles of the top3D125 code in MATLAB is shown in Figure 3.1.

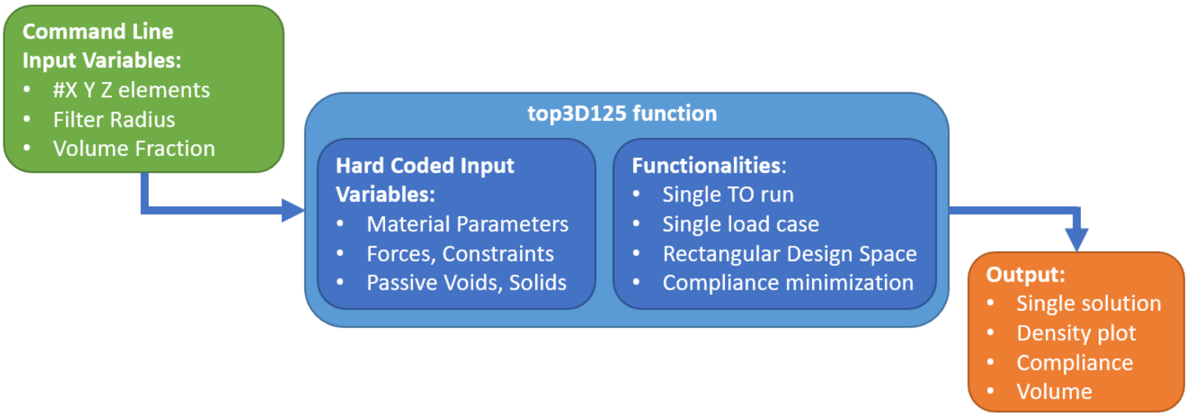


Figure 3.1: Schematic Overview top3D125 in MATLAB

In the top3D125 code, the design domain is simplified to a rectangular grid that is discretized by cubic finite elements [29, 39]. This keeps the numbering of elements and their corner nodes relatively simple, but also puts a limit on the shape of the input domain. The top3D125 function can be called with one line in the Command Window, while giving some variables as input. Inside the function, more input variables are provided hard coded, to set up the rest of the structural problem. Constraints and loads are defined by selecting nodes and the corresponding Degrees of Freedom (DoF) in the targeted direction by number. Besides that, passive void or solid areas can be defined by the targeted element numbers. Even though the numbering system is straight forward, it takes quite some time to setup a new problem. The standard version of the code performs a single compliance minimization run, with a single load case. It shows a simple density plot of the solution while running, and prints the values of the compliance and volume of the part in each iteration, next to some optimization settings.

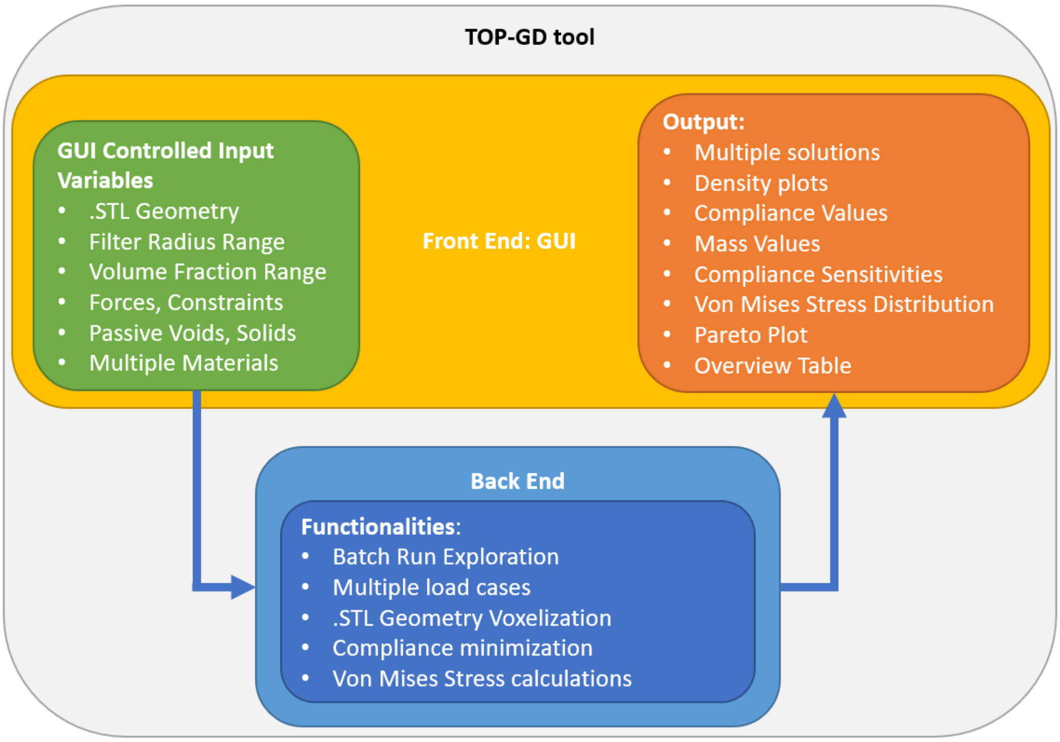


Figure 3.2: Schematic Overview working principles TOP-GD tool

To satisfy all the requirements defined in Section 3.1 and implement a more user-friendly workflow, the top3D125 code was extended and complemented with a GUI. This extensive development process took around 4 months, resulting in the TOP-GD tool. The complete source code of the TOP-GD tool can be found in appendix A, totalling around 2300 lines. A schematic overview of its working principles and functionalities is shown in Figure 3.2.

The TOP-GD tool is an app created in MATLAB App Designer, which contains a GUI and has structured functions running in the background to provide all functionalities. In the GUI, the problem can be set up using a visual 3D representation of the design domain, and several components to control all input variables. Once the problem is set up, all information is send to the topology optimization function, which performs a batch run of all given combinations of input settings and material parameters. This function then sends the results back to the GUI of the TOP-GD tool, where all solutions are displayed and can be compared by the user.

The development approach to create the TOP-GD tool can be roughly divided in 4 steps; Extending the top3D125 code, Translation to GUI controlled input variables, Moving from a single to multiple solutions, and Optimizing the presentation of results. To keep the description of the code and development process within limits, a summary of these steps is presented below.

3.2.1. Extending the top3D125 Code

As was explained, the starting point for the development process is the top3D125 code [39]. Here the design domain is simplified to a discretized rectangular grid, which enables a simple numbering system for elements end their corner nodes. This numbering system is used to define all hard-coded input variables, such as the loads and constraints. A convenient feature of the code is that it is able to set elements to passive "Void" elements or passive "Solid" elements as well. All elements are initialized with a density value of 0. Defining an element as part of the passive void or passive solid set, excludes them from the active design variables. This means that the density values of the elements in the passive void set are kept at 0 throughout the optimization, and the density values of the elements in the passive solid set are set to 1. After an .STL file is imported using the MATLAB function stlread [43], this feature of passive elements can be utilized together with a voxelization approach to enable the optimization of arbitrarily shaped design domains. During voxelization, a continuous geometric object such as an .STL file or a 3D triangular mesh is converted to a discretized voxel-based representation [44]. This is schematically illustrated in Figure 3.3. A rectangular design domain that fits around the complete geometry is defined, and divided into small cubic "voxels" or elements. All elements in this larger rectangular domain that do not correspond to the freely shaped continuous geometry, are identified with the VOXELISE.m function [45], and consequently set to passive void elements that do not participate in the optimization. The finer the voxel grid is set up, the better the voxelized model will approximate the geometry in the .STL file, but the more demanding the TO runs will be. Since the goal for the TOP-GD was to explore multiple solutions quickly using coarse geometries, the voxelization approach is an effective way to enable arbitrary design domain optimizations in combination with the top3D125 code.

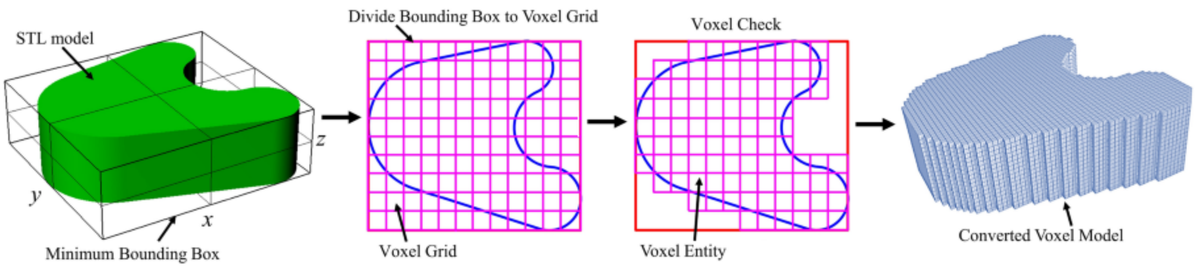


Figure 3.3: Voxelization of an arbitrarily shaped 3D geometry [46] (edited)

Furthermore, the code was extended to be able to handle multiple load cases, by extending the force and displacement vectors to multiple column vectors and changing the calculation of the objective function to the sum of the calculated compliances for each load case. This was done with the help of the instructions given in the 88-line code paper [42] and applying those with some adjustments to the newer top3D125 code. The number of columns of the force and displacement vectors will be coupled to the number of load cases defined in the GUI in the next development step.

In order to give the user information on the stresses in the part, the code was also combined with a part of the 146-line stress-based topology optimization code written by Deng et al. [47]. This is used to calculate the Von Mises stress in each element of the topology, and find the maximum value. To make sure these calculations do not slow down the optimization too much, the Von Mises stress distribution is only calculated for the last iteration of the compliance optimization run, purely informative and not providing stress constraint functionality. Still, this enables the user to check the maximum Von Mises stress value in the solution, and take this into account when picking a solution.

3.2.2. Translation to GUI Controlled Input Variables

As was explained above and can be seen in Figure 3.1, the top3D125 code is a function that has two types of input variables. Only a few basic settings are given as input variables to the function in the command line, e.g. the amount of elements of the design domain in x, y and z direction, the volume fraction and the filter radius. The rest of the problem definition and settings, such as the loads, constraints or material parameters are provided hard-coded inside the function. This is however not user-friendly at all, since the user has to go through the code manually, and figure out the numbering system of elements and nodes in 3D. Furthermore, every time a new problem is to be investigated, the code has to be edited. Since one of the main attention points for the TOP-GD tool is user-friendliness, the TO code is therefore altered to be usable with a newly created GUI to control all settings. In order to achieve this, all the interesting hard-coded settings are defined as input parameters for the TO function, just like the command line input variables. Next, each of these input variables were coupled to a controllable input element in the GUI. In this way, the setup of a design problem is made possible entirely from the GUI in a visual and user-friendly way, without needing to have a single look at the TO code running in the background. The TOP-GD GUI consists of multiple 'tabs', the first one being the "Setup" tab shown in Figure 3.4, where all needed input variables are defined to run an optimization.

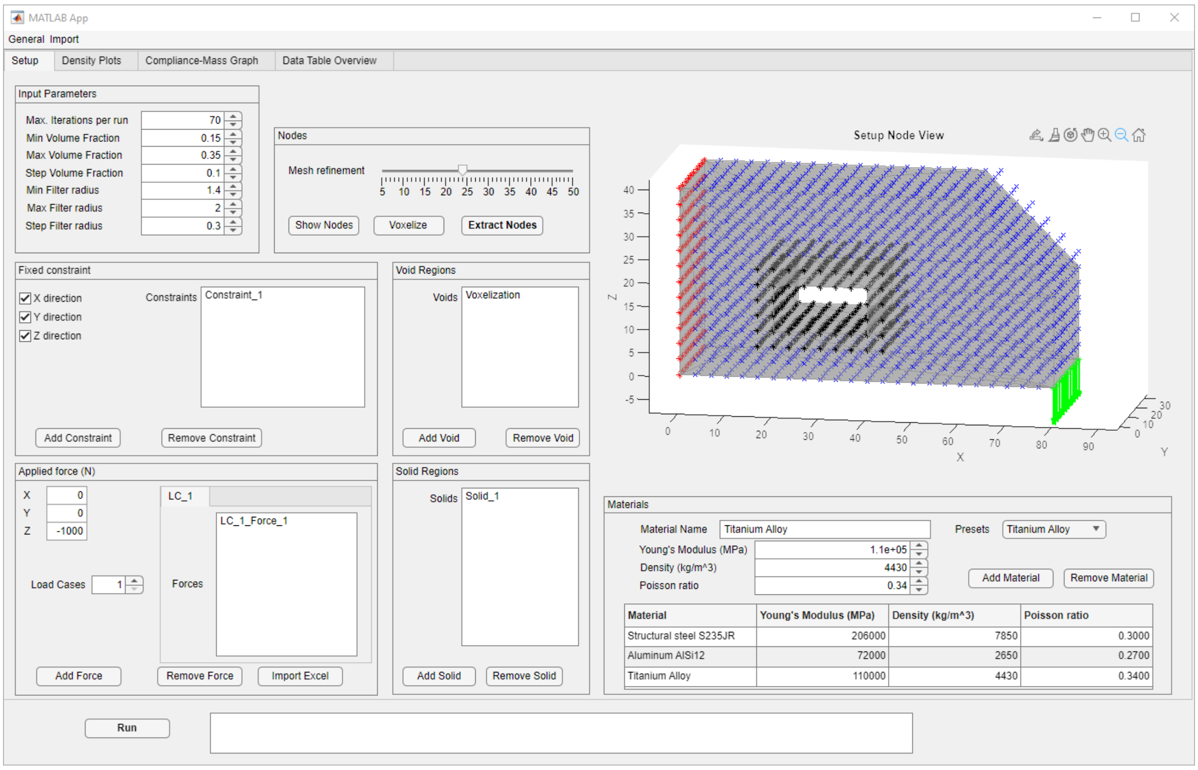


Figure 3.4: Overview of all elements on the Home Screen "Setup" Tab of the TOP-GD tool

The easiest input parameters to define are controlled by so called 'spinners' in the GUI, shown in the "Input Parameters" section. These are simple numerical value boxes, of which the value can be edited by the user. These values will be directly used as input variables for the TO function once the problem is set up. The input variables defined here are the Maximum iterations per run, the volume fraction and the filter radius. Besides that, the material parameters that should be considered for the part are defined by the user with more spinners for the Young's modulus, density value and Poisson ratio of the material, which are directly used as input variables as well.

To define the constraints, forces, passive solid and void elements as input variables, the workflow involves a few more steps. First of all, a geometry is imported as an .STL file with the "Import Geometry" button. This opens the file explorer for the user to select an .STL file. This file is imported and shown in the "Setup Node View", where the geometry can be rotated and moved using MATLAB's 3D navigation features. In the Nodes section, the user can subsequently press the "Show Nodes" button, which plots a rectangular grid of nodes, representing the elements of the mesh, over the geometry. The mesh refinement slider can be used to control the amount of elements that are used for the optimization. In case a non rectangularly shaped geometry is used, the "Voxelize" button can be pressed to set redundant areas to passive void elements, as was explained in the previous section. This will also visually remove the corresponding nodes in the "Setup Node View". The visual representation of all active nodes can now be used to define the rest of the problem, using MATLAB's "brush" functionality. This makes the user able to select nodes in the "Setup Node View", and defining them as constraints and forces. This is done by extracting the targeted nodes, defining which direction should be constrained or loaded, and clicking the "Add Constraint" or "Add Force" buttons. How this works in the back-end is that the selected nodes in the figure are coupled to the node numbering system of the rectangular grid, and the targeted DoF's are added to an array. These arrays are used as input variables to the TO function once the complete problem is set up. To define solid or void regions, nodes can be selected and extracted in the same way as for forces and constraints. The only difference with selecting individual nodes for constraints or forces, is that all 8 corner nodes of an element should be selected in order to target it completely. If that is the case, the corresponding element number will be added to either the passive void or solid array. By working with arrays in this way, multiple constraints, loads, passive void or solid elements can be added after each other, or simultaneously.

To define multiple load cases, an extra load cases spinner is added to the Force setup area in the lower left corner of the GUI. If the value in this input box is for example set to 2, an extra load cases tab is created which contains another independent force list. Each tab corresponds to one load case, and clicking on them will show only that load case in the "Setup Node View" figure. This value is also used as input to change the number of columns of the force and displacement vectors in the back-end of the tool, enabling the combined objective calculations.

Every time a constraint, load, solid or passive region is added, an item is added to the corresponding list of these sections in the lower left part of the GUI. By clicking on the name of an item, the nodes that are linked with it are highlighted in the "Setup Node View". In this way it is clear to the users what constraints, loads, solid or passive regions have been added so far and where they are located. Any of these items can be selected and removed again as well, which also removes the node or element numbers from the corresponding input variable array.

When the user has completed the problem setup by adding at least one constraint and one force, and filled in the optimization settings and material parameters, pressing the "Run" button will start the TO code. All values and arrays that have been defined with the help of the GUI as described above, are transferred as input variables to the TO function in the background.

3.2.3. Moving from Single to Multiple Solutions

After enabling the complete setup of a topology optimization problem in the new TOP-GD GUI, a single optimization could be ran from there, which was already a user-friendly Basic TO tool. For the new approach however, the goal was to explore multiple solutions and the influence of optimization settings automatically, without the need for the user to repeat the same setup process. Therefore, the GUI was extended in such a way that the user can give a range of values to be explored for certain optimization settings, instead of providing just one value. This was done for the volume fraction and the filter radius, because these parameters influence the geometries and performance values of the designs significantly. Varying this therefore contributes to a diverse set of solutions, and increases the understanding of the effect of these parameters on the results. In the "Input Parameter" section in the top left part of the GUI (see Figure 3.4), extra spinners were added to specify the minimum and maximum values for the volume fraction and filter radius to be explored. Moreover, a step size value was added for both of these settings to control the level of detail the range is explored with, and the time necessary for the combined runs.

Regarding the material section, the exploration of multiple materials was also enabled. Different material parameters like the Young's modulus and density strongly influence the compliance and mass values of the output solutions, and are therefore interesting to explore for multi-objective problems. Instead of giving the material parameters of one material as input variables, these values are therefore added to a material array and shown in the materials table in the lower right area of the setup tab of the GUI, shown in Figure 3.4. To this array, the material parameters of other materials can optionally be added as well, which will be shown in a new row in the materials table. To make the workflow easier for the user, some presets have been defined for common used materials like Structural Steel, Aluminium and a Titanium alloy. These can be selected from a drop down menu, after which the corresponding material parameters appear in the spinner boxes. These can first be edited, or added directly to the material array by pressing the "Add Material" button. A material can be removed from the material array again by selecting the corresponding row in the table and pressing the "Remove Material" button. All parameters of the materials present in the material array are used as input variables for the TO function once the "Run" button is pressed.

With these ranges of input settings and different materials, multiple nested for loops were build in the back-end of the TOP-GD tool to explore all combinations of settings. This lets the TOP-GD tool quickly provide a range of solutions, faster than a user could edit the settings and repeatedly run the single TO tool. All these solutions are plotted one by one while the tool is running on the Density Plots tab, shown in Figure 3.5. In the title of each plot, information is given on their weight, compliance, material and optimization settings used. This shows to the user live how the exploration run with different settings is changing the solutions, and makes the waiting time while the TOP-GD tool is running useful and insightful. However, next to just plotting all density plots of the solutions side by side, there was still potential to generate a better overview of the influence of different settings and boundary conditions and compare the solutions.

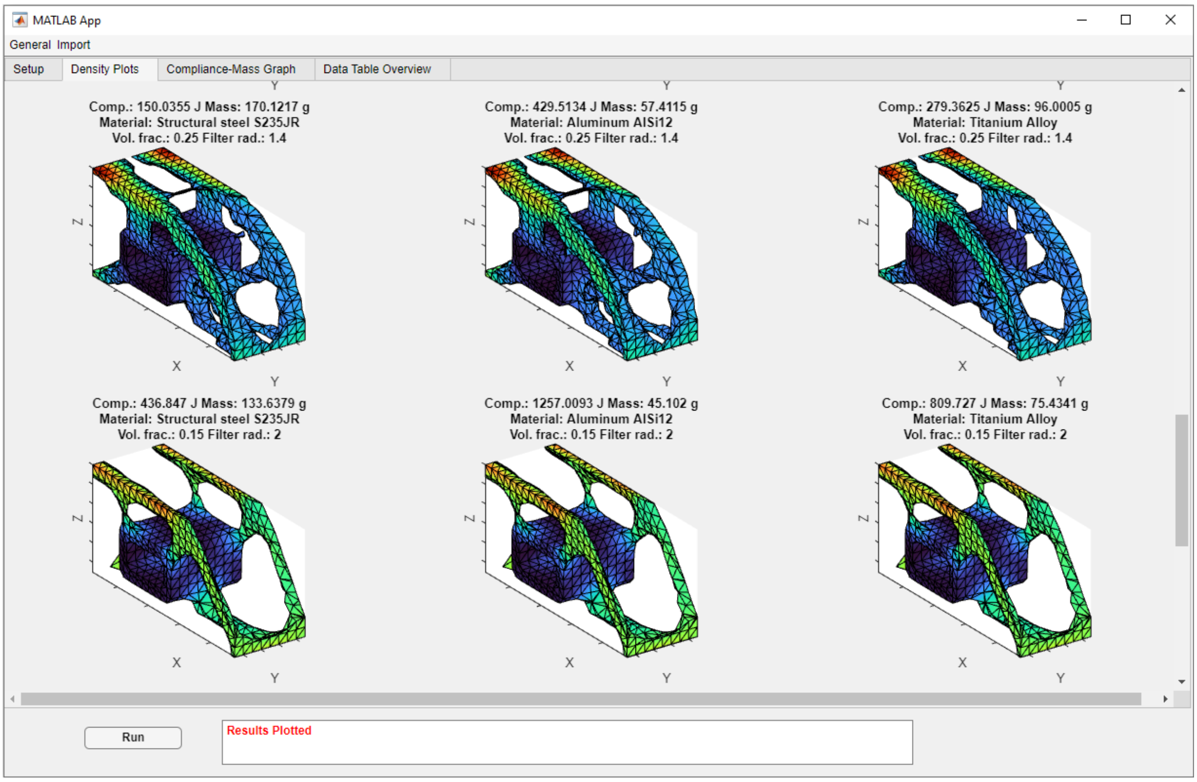


Figure 3.5: Overview of the "Density Plots" Tab of the TOP-GD tool

3.2.4. Optimizing the Presentation of Results

With different solutions as data points, the next challenge was to present the output of the explorative TOP-GD run in a user-friendly way. As was explained, it is of added value to show the trade-off between multiple objectives because it helps the designer to understand the essence of a design problem, and make well-founded decisions. For the compliance minimization problems solved in the TOP-GD tool, the interesting objectives to look at are the mass and compliance values of each solution. To visualise the trade-off between these objectives, another tab was therefore created in the GUI (shown in Figure 3.6) with a Compliance-Mass graph showing every solution plotted as a data point. This makes it easier to compare the performances of the solutions in terms of each of these objectives. Different colors are used in the Compliance-Mass Graph for the data points corresponding to each material, to highlight their influence on the solutions' performance.

To further inspect a solution, the user can select a data point which will show all the corresponding performance values and settings used in a table below the graph: the Compliance, Mass and Maximum Von Mises Stress values, and the Volume Fraction, the Filter Radius and the Material. Besides that, the solution's geometrical features will be displayed next to the graph, twice (see Figure 3.6). In the top right figure, extra information is added to the density plot by means of a coloring scheme. This coloring scheme corresponds to the compliance sensitivity values calculated in the TO code, and have been transformed in such a way that they could be visualised. Consequently, useful extra information is provided to the user, because critical areas of the solution with a higher compliance sensitivity (see Section 2.2.2) are clearly highlighted. The Von Mises Stress values calculated in the last iteration of the optimization were transformed in the same way as the compliance sensitivities, resulting in the lower right density plot. Here, the coloring scheme represents the Von Mises stress per element, giving information on the distribution of stress in the solution and visualizing possible stress concentrations.

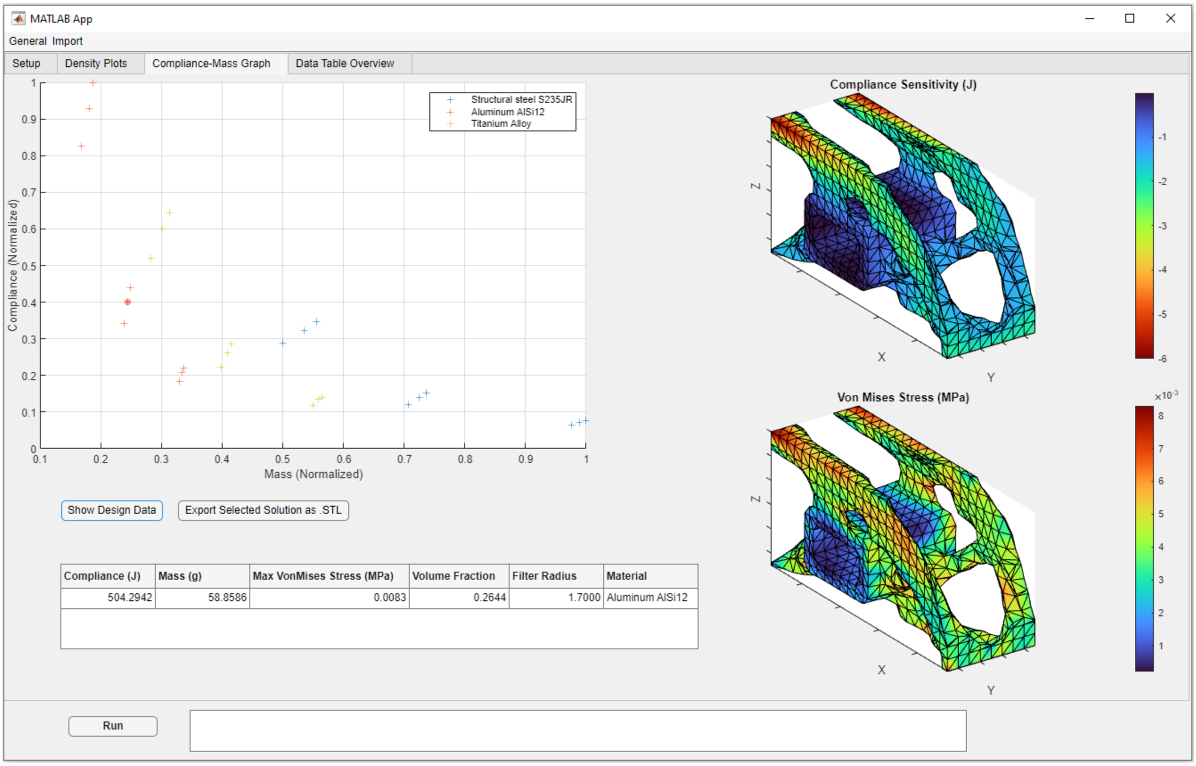
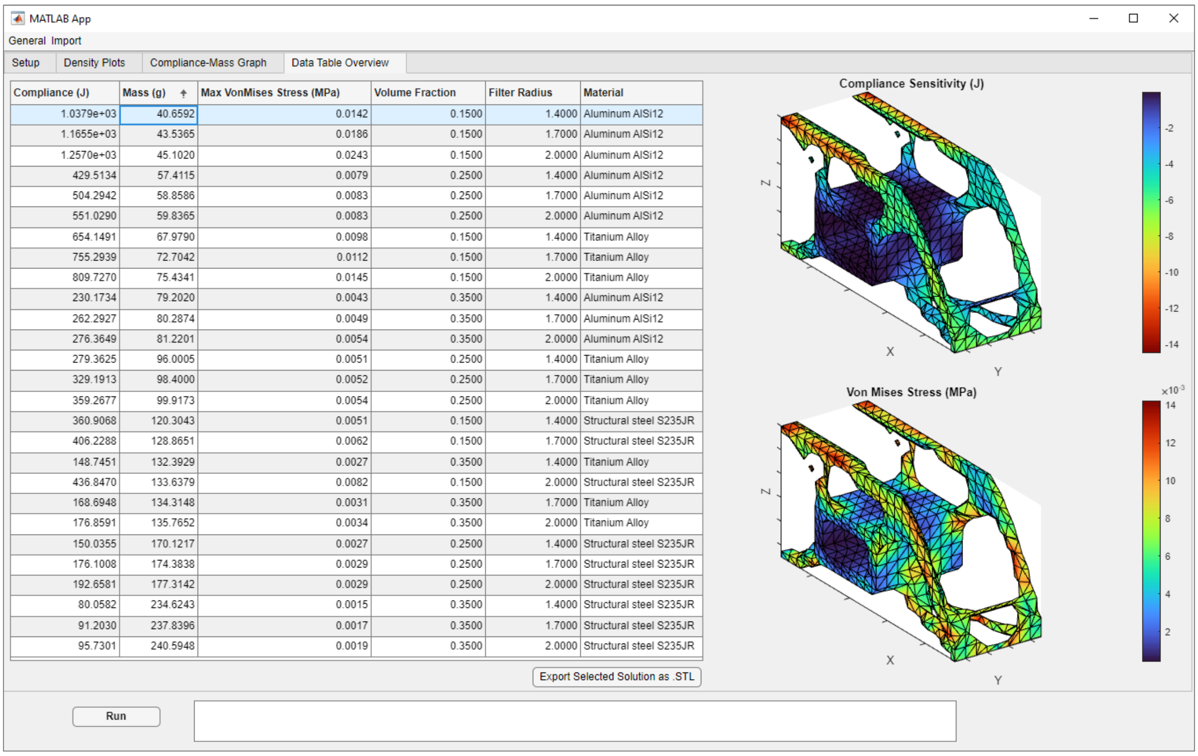


Figure 3.6: Overview of the "Compliance-Mass Graph" Tab of the TOP-GD tool

Both the coloring schemes showing the compliance sensitivities and Von Mises stress per element, moreover provide the user with extra information on for example the boundary conditions or design space. If one specific area of the design space turns out to be critical in most solutions, the designer can decide to locally enlarge the design space there, or move the force application location if that is possible. This is usually a more effective measure to improve the efficiency of the part, compared to finding the optimum within worse boundary conditions. Because the TOP-GD tool is quick and easy to use and is meant for the early design stages, it is still possible to make design changes and very useful to have this kind of information early on in the design process.

After the Density Plots tab and the Compliance-Mass Graph tab, another tab was added to the GUI to help the user in having more overview. This "Data Table Overview" tab, depicted in Figure 3.7, is a table overview where each row represents one of the solutions. The different columns contain both the input settings information, and the resulting compliance, mass and maximum Von Mises Stress present in the geometry. What makes this table interesting however, is that each column can be sorted with a simple mouse click. This shows all solutions instantly in ranked order, from e.g. stiffest to most compliant, or from the lowest mass to the highest mass. Because all input settings are visible simultaneously, this data table overview can help to discover patterns between all solutions, or improve the understanding of the effect of the different input settings. Each row in the table can be selected to show both a compliance sensitivity plot and a Von Mises Stress plot of the corresponding solution next to the table again.



Compliance (J)	Mass (g)	Max VonMises Stress (MPa)	Volume Fraction	Filter Radius	Material
1.0379e+03	40.6592	0.0142	0.1500	1.4000	Aluminum AlSi12
1.1655e+03	43.5365	0.0186	0.1500	1.7000	Aluminum AlSi12
1.2570e+03	45.1020	0.0243	0.1500	2.0000	Aluminum AlSi12
429.5134	57.4115	0.0079	0.2500	1.4000	Aluminum AlSi12
504.2942	58.8586	0.0083	0.2500	1.7000	Aluminum AlSi12
551.0290	59.8365	0.0083	0.2500	2.0000	Aluminum AlSi12
654.1491	67.9790	0.0098	0.1500	1.4000	Titanium Alloy
755.2939	72.7042	0.0112	0.1500	1.7000	Titanium Alloy
809.7270	75.4341	0.0145	0.1500	2.0000	Titanium Alloy
230.1734	79.2020	0.0043	0.3500	1.4000	Aluminum AlSi12
262.2927	80.2874	0.0049	0.3500	1.7000	Aluminum AlSi12
276.3649	81.2201	0.0054	0.3500	2.0000	Aluminum AlSi12
279.3625	96.0005	0.0051	0.2500	1.4000	Titanium Alloy
329.1913	98.4000	0.0052	0.2500	1.7000	Titanium Alloy
359.2677	99.9173	0.0054	0.2500	2.0000	Titanium Alloy
360.9068	120.3043	0.0051	0.1500	1.4000	Structural steel S235JR
406.2288	128.8651	0.0062	0.1500	1.7000	Structural steel S235JR
148.7451	132.3929	0.0027	0.3500	1.4000	Titanium Alloy
436.8470	133.6379	0.0082	0.1500	2.0000	Structural steel S235JR
168.6948	134.3148	0.0031	0.3500	1.7000	Titanium Alloy
176.8591	135.7652	0.0034	0.3500	2.0000	Titanium Alloy
150.0355	170.1217	0.0027	0.2500	1.4000	Structural steel S235JR
176.1008	174.3838	0.0029	0.2500	1.7000	Structural steel S235JR
192.6581	177.3142	0.0029	0.2500	2.0000	Structural steel S235JR
80.0582	234.6243	0.0015	0.3500	1.4000	Structural steel S235JR
91.2030	237.8396	0.0017	0.3500	1.7000	Structural steel S235JR
95.7301	240.5948	0.0019	0.3500	2.0000	Structural steel S235JR

Figure 3.7: Overview of the "Data Table Overview" Tab of the TOP-GD tool

When the user is done exploring the different solutions and has picked one to continue working with, they have the possibility to export that solution in .STL format. Both on the Compliance-Mass Graph and Data Table Overview tab shown in Figure 3.6 and Figure 3.7 respectively, a solution can be selected and exported using the "Export Selected Solution as .STL" button. This opens the file explorer of the computer to save the solution.

3.2.5. Extra Added Functionalities

Besides the functionalities described in the previous sections that already satisfy all the requirements mentioned in Section 3.1, some small extra functionalities were added to improve the user experience while using the TOP-GD tool. For example, it was made possible to save and import a setup file, containing all the necessary information regarding a geometry, loads, constraints, solid and void regions, materials and input parameter settings. This enables a user to continue with a setup at a later time, without needing to repeat the setup process after the TOP-GD app has been closed. Next to that, a reset button was added to start with a fresh problem without needing to close and reopen the TOP-GD tool or removing all loads and constraints one by one.

To speed up the process of adding multiple forces or load cases, it is also possible to import an Excel into the TOP-GD tool. In this Excel worksheet, the load case, attachment location, magnitude and direction of each force should be specified. Moreover, it is possible to define the force attachment location as a line, circle or sphere with a variable radius. Especially when working with various load cases and forces, this feature is more efficient compared to manually selecting nodes and specifying a direction and magnitude for each force. Besides that, this feature makes it possible to select nodes internally in the geometry, which is harder to do with the brush functionality in the 3D ”Setup Node View” provided by MATLAB.

A complete video tutorial has been made to show the TOP-GD tool and explain the workflow for users, which is used for the experiment described in the next chapter. This video can be found on YouTube with the following link: `https://youtu.be/UauV1bRjx8M`

4 Experiment Methods

After the creation of the TOP-GD tool, its influence on the design performance and experience had to be tested and this is therefore the focus of the experiment performed for this thesis. Like was explained in the introduction, it is interesting to do a comparison study of the design process with different approaches to be able to review this. Three approaches of the design process are therefore formulated to compare within the experiment: designing manually without any aid of computational design techniques, using a simple 'single run' TO tool to design, and designing with the new topology optimization based generative design TOP-GD tool. The main research question of this thesis therefore was: "What is the effect of using a Topology Optimization based Generative Design tool or a simple Topology Optimization tool compared to manual design on the design performance and experience?"

4.1. Experimental design

In order to compare these three approaches, a within-subject experiment has been set up with multiple design assignments of similar complexity to study the effect of the different approaches on the design performance and experience.

Independent Variable: Design approach

The independent variable is therefore the approach used to design. The three different approaches are defined below:

1. To design manually, the participants were only given pen and paper, and no access to any additional aid or computer.

2. For the Basic 'single run' TO tool, a simpler earlier version of the tool was used in MATLAB, with a GUI that was as similar to the advanced app as possible. The GUI of this simple app is shown in Figure 4.1 below. This GUI only has one screen, showing the same setup sections as the setup tab of the TOP-GD tool. The workflow for the setup is exactly the same as in the TOP-GD tool, and it has the same functionalities in terms of for example multi-load case problems and importing or exporting .STL files. The only difference is that under the "Input Parameters" section, no ranges can be set for different parameters, but only one value can be given. Moreover, instead of adding multiple materials that need to be evaluated in a separate material section, the material parameters of one type of material can be entered under the "Input Parameters" section. When hitting the "Run" button, a single TO run is performed, and the result is directly plotted in the lower right corner of the screen. The coloring scheme representing the sensitivity information has been left out and only the final volume, mass and compliance values are shown in the text area. Another run can then be performed by adjusting the input parameters or the constraints, forces, void and solid regions, which overwrites the resulting geometry shown.

3. For the last Topology Optimization based Generative Design approach, the TOP-GD tool is used with all functionalities as described in the previous chapter.

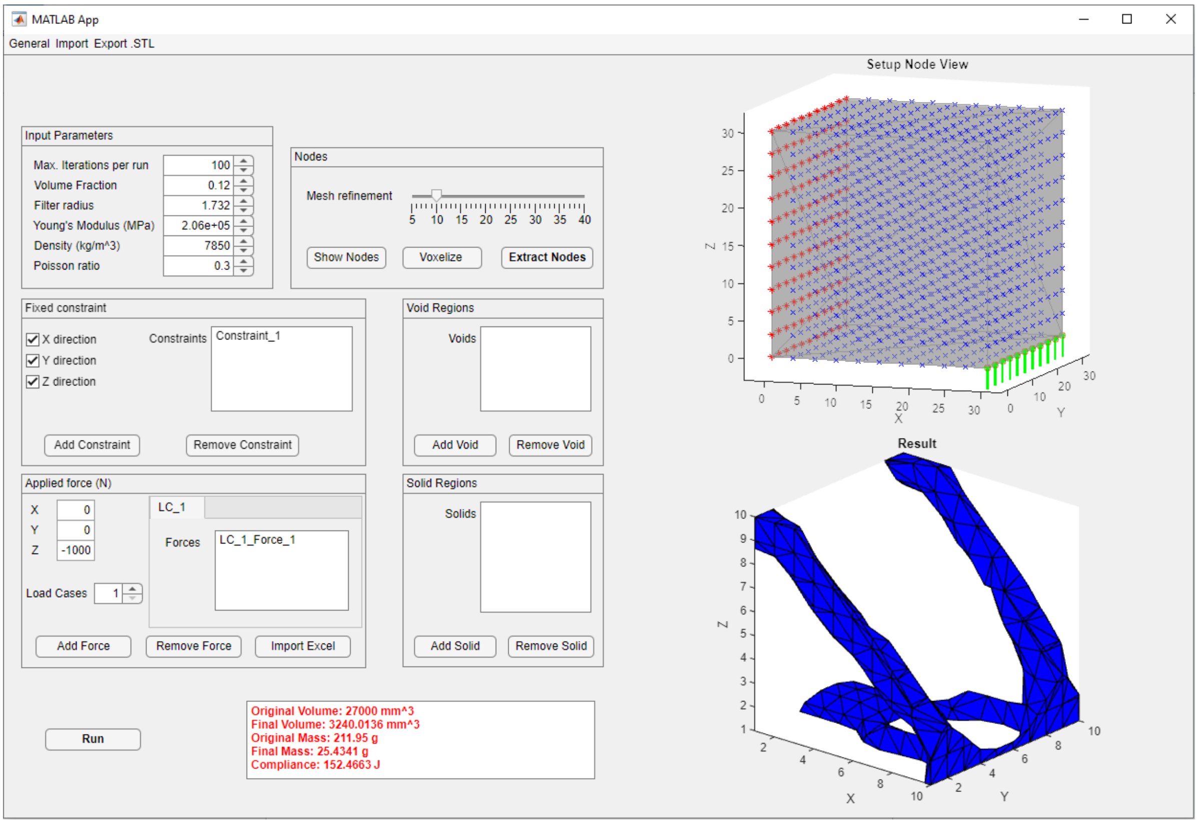


Figure 4.1: Simple version of TO App with single-run functionalities

Participants

The participants for this experiment are all Mechanical and Structural Engineers, working at the Dutch solar car company Lightyear. They therefore have general engineering knowledge, and experience with designing structural parts. However, not all of them are experienced with Topology Optimization or Generative Design and therefore need a basic instruction to partake in the experiment.

This thesis was originally started in cooperation with Lightyear, and the experiments were planned to be held at their facilities in Helmond. However, just before the experiment sessions were executed, Lightyear was declared bankrupt in the beginning of February 2023. The originally 15 volunteers for the experiment were therefore unfortunately reduced to only 3 engineers willing and able to come to the TU Delft to participate in the experiment set out for this thesis. It was considered to try to find more participants at other engineering companies, but since this would mean more delay in the thesis project and would provide a less homogeneous set of participants, it was decided in consultation with the supervisors to continue with the experiment in this smaller setting.

Design Task

Three different design assignments of similar complexity were formulated for this experiment. One of the assignments was to design a structural arm, another was about designing a bracket and the third assignment was to design a kitchen step. All three assignments including instructions are included in Appendix B, exactly as they were given during the experiment. The participants were given 15 minutes to finish each of the assignments, and come up with a solution that was both realistic and as efficient as possible.

Dependent variables

The effect of the different approaches on the design performance and experience can be measured in multiple ways. First of all, the performance of the parts can be measured with different performance measures. Of the designed parts, therefore the mass and compliance are taken as dependent variables that can be measured and compared.

Besides the part performance, survey questions are asked to asses the subjective design experience each participant had with the different design approaches, and be able to compare them. The answers to these questions are for example used to asses the user-friendliness of both the tools.

And lastly, Eye-tracking is used to evaluate and compare the two digital tools in the Basic TO approach and the TOP-GD approach. Eye-tracking is a method used to record and analyze where individuals focus their visual attention. This technique has been used in a variety of research fields, including human-computer interaction (HCI) [48], to understand how users interact with software tools. Eye-tracking data can provide several insights into how individuals use software tools. For example, it can reveal the areas of the tool that users focus on the most, indicating which features are most salient and attracting the user's attention. Besides that, eye-tracking data can be used to identify where users experience difficulties or challenges when using the tool. This can be indicated by prolonged fixation times or saccades, which suggest that users are struggling to locate or process information.

4.2. Materials and Equipment

For manual design, only pen and paper were used to execute the design assignment.
Both the Basic TO tool and the TOP-GD tool were installed and evaluated on the ROG Zephyrus S GX531. This is a 64-bit laptop with Windows 10, Intel core i7 processor and 16 GB of RAM [49]. For the installation, first MATLAB and the necessary toolboxes were installed on the laptop, after which both tools immediately ran smoothly and could be used for the experiments.
The Eye-tracking part of the experiment has been done with the EyeLink Portable Duo [50], shown below in Figure 4.2:

Figure 4.2: SR Research' EyeLink Portable Duo

Figure 4.3: Remote Head Tracking Stickers

When a participant puts a Head Tracking Sticker (shown on the lower left corner of the box in Figure 4.3) on their forehead, this device is able to track both their eyes and pupils after calibration. This allows the collection of gaze data on the screen of the laptop used. The complete setup of the EyeLink Portable Duo mounted on top of the ROG Zephyrus laptop as was used in the experiment is shown in Figure 4.4.

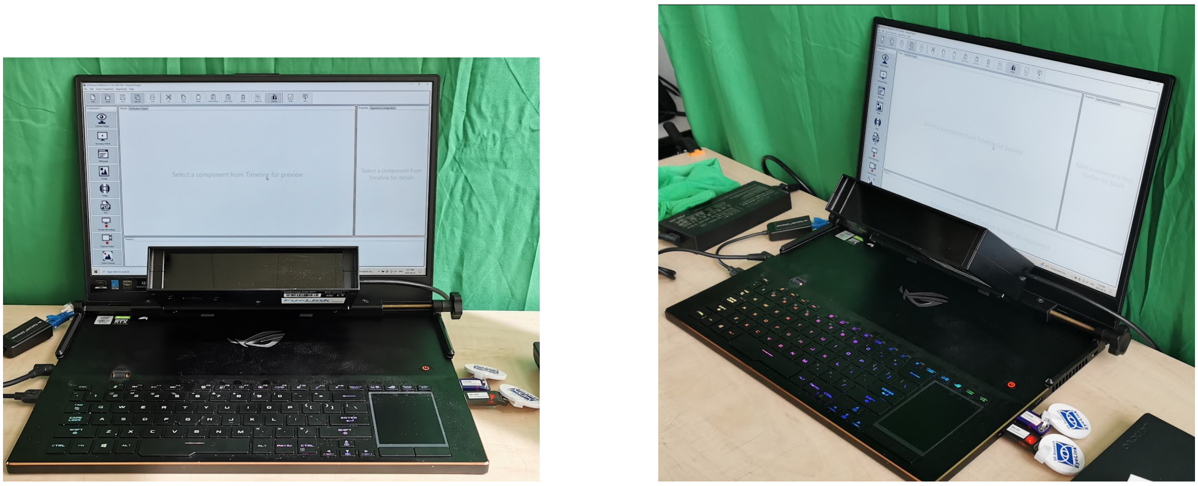


Figure 4.4: Front View Experiment Setup

The software used together with the equipment for Eye-tracking is WebLink [51]. In this software the tracker can be calibrated and screen recordings can be started in order to start data collection sessions during the design tasks with the Basic TO and TOP-GD tool in the experiment. Once the experiments have been finished, WebLink can be used to export the data in Excel format to further process it in MATLAB.

4.3. Procedure

A test experiment was conducted twice, with both supervisors separately. The feedback given during these test experiments was processed before performing the final experiments. The steps of the procedure followed during each experiment are enumerated chronologically below:

1. (5 min) Explanation of what the experiment will entail
2. (5 min) Survey Part 0, basic information on experience participant in Google Forms
3. (9 min) Watch Video: Crash Course Basics Topology Optimization
4. (15 min) Design task 1: Manual design assignment with pen and paper
5. (5 min) Survey Part 1 about Manual Designing in Google Forms
6. (9 min) Watch Video: Instruction Basic Topology Optimization tool
7. (2 min) Set up and calibrate Eye-tracker
8. (15 min) Design task 2: Design assignment with Basic TO tool
9. (5 min) Survey Part 2 about designing with the Basic TO app in Google Forms
10. (7 min) Watch Video: Instruction TOP-GD tool
11. (2 min) Set up and calibrate Eye-tracker
12. (15 min) Design task 3: Design assignment with TOP-GD tool
13. (5 min) Survey Part 3 about designing with the TOP-GD tool in Google Forms

Before the start of the experiment, a short introduction and explanation of what to expect during the experiment was given to each participant. A general survey (part 0) containing questions about their engineering background and experience on Topology Optimization was given on a laptop using Google Forms. The complete survey can be found in Appendix C. Because some basic knowledge on Topology Optimization is necessary to be able to usefully use both the basic TO tool and the TOP-GD tool, a video was recorded explaining the basics of TO. Each participant was shown this same video, to make sure all participants had the same background information regardless of their experience with TO. The introduction video can be found on YouTube with this link: `https://youtu.be/hmw3SqCsua0`.
At this point the first manual design task was given to the participant, with only the availability of pen and paper. After 15 minutes, the next part of the survey was given with questions about their experience while manually designing.

Another video was made introducing the participants to the Basic TO tool made for this experiment. The most important functionalities needed to set up a structural problem and run an optimization are shown in this video, making them able to use the tool on their own for the next design task. This video introducing the Basic TO tool can be found on YouTube with this link: `https://youtu.be/FpK0-JoCjkI`. After the Eye-tracker had been activated and calibrated on the participant, the second design assignment was given with the Basic Tool. Their experience of designing with the Basic Tool was reviewed in the next survey part.

Finally, a third video was shown to the participants introducing the TOP-GD tool and the extra functionalities available. This video can be found on YouTube with the following link: `https://youtu.be/UauV1bRjx8M`. The third assignment was then performed followed by the last survey questions to review their design experience with the TOP-GD tool.

The three assignments (shown in Appendix B) were alternated among the three participants to minimize the influence of differences within the assignments on the results. This was done in the following way:

Task #:	1: Design Manually	2: Design with basic TO tool	3: Design with TOP-GD tool
Participant 1	Assignment A	Assignment B	Assignment C
Participant 2	Assignment B	Assignment C	Assignment A
Participant 3	Assignment C	Assignment A	Assignment B

Table 4.1: Order of Assignments given per Participant and per Design Approach

The use of repetitions using more assignments per participant was considered, but since the total experiment already takes up to 2 hours per participant it was decided to do the experiment with just these three assignments.

Because of the many instructions and tutorials needed in the experiment, it was deemed more logical to stick to this chronological order of the design tasks, increasing in complexity from designing manually to the Basic TO tool and ending with the TOP-GD tool. Even though there might be a slight learning curve effect in the results, it does not make sense to do the experiment in another order. This would mean already seeing all introduction videos right at the start of the experiment, which totals about a half an hour of information. This takes a lot of the attention span of the participants and can also influence their experience in for example the Basic TO app, by knowing there are extra functionalities that are not available for use yet. Therefore, starting simple and increasing in complexity and given information during the experiment was chosen as approach.

5

Experiment Results

The experiment was conducted at the TU Delft and only 3 participants took part in the final study, due to Lightyear's bankruptcy. They were all Dutch, male and under 35. All participants completed the experiment successfully, generating at least one solution for each design task within 15 minutes. As was explained in Chapter 4, the effect of the different approaches on the design performance and experience is measured in multiple ways. The geometries of the designed parts are presented in the next section, together with the compliance and mass values of the two solutions generated in the Basic TO and TOP-GD app. Furthermore, a survey was conducted to assess the design experience of the participants, of which the results are presented in Section 5.2. Lastly, the Basic TO and TOP-GD tool have been tested while collecting eye-tracking data. This data is presented in Section 5.3.

5.1. Part Performance Results

An overview of the designed and selected solutions for assignment A (see Appendix B) can be seen in Figure 5.1. The manually designed solution is still rectangularly shaped on the outside like the design space, but with two internal crossing members. The solutions generated with the Basic TO and TOP-GD tool are shaped more like a parallelogram, and have a plated but hollow section near the fixed ring on the top right corner. The TOP-GD solution has a more open structure compared to the Basic TO solution. Moreover, the Basic TO solution shows some disconnected areas around the outer rings, due to a coarse mesh choice.

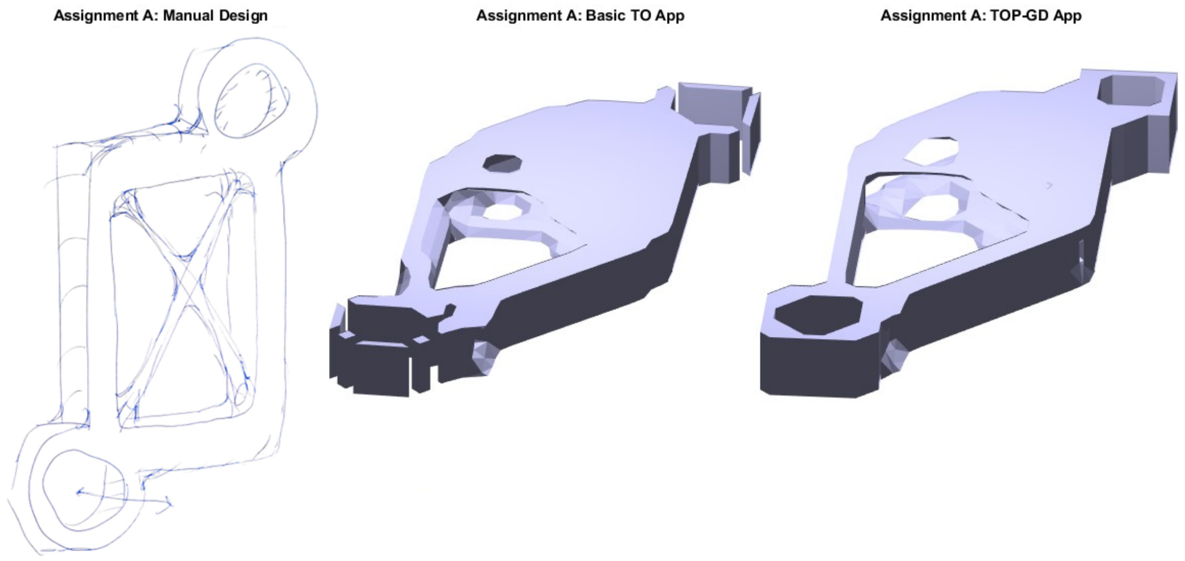


Figure 5.1: Results for Assignment A for each approach

An overview of the designed and selected solutions for assignment B (see Appendix B) can be seen in Figure 5.2. The manually designed solution is similar but simpler to the digitally designed solutions, with only one crossing member. Again, the TOP-GD solution has a more open structure compared to the Basic TO solution in the middle part of the geometry.

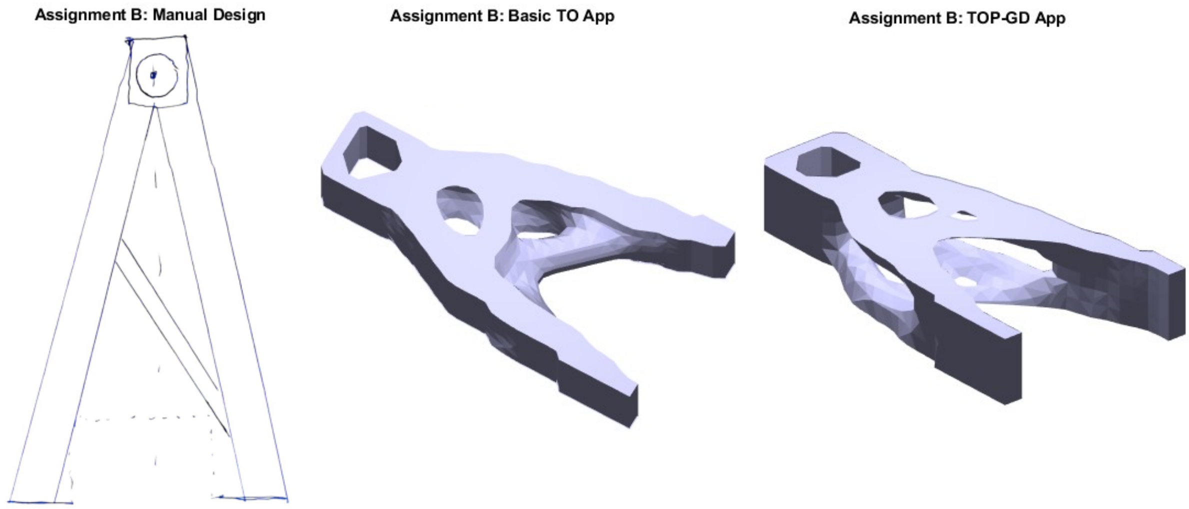


Figure 5.2: Results for Assignment B for each approach

An overview of the designed and selected solutions for assignment C (see Appendix B) can be seen in Figure 5.3. In this case, the manually designed solution is more complex with many internal connecting bars. The Basic TO solution has two organically shaped legs both on the front and backside. The TOP-GD solution has two legs supporting the backside, and three legs in the front.

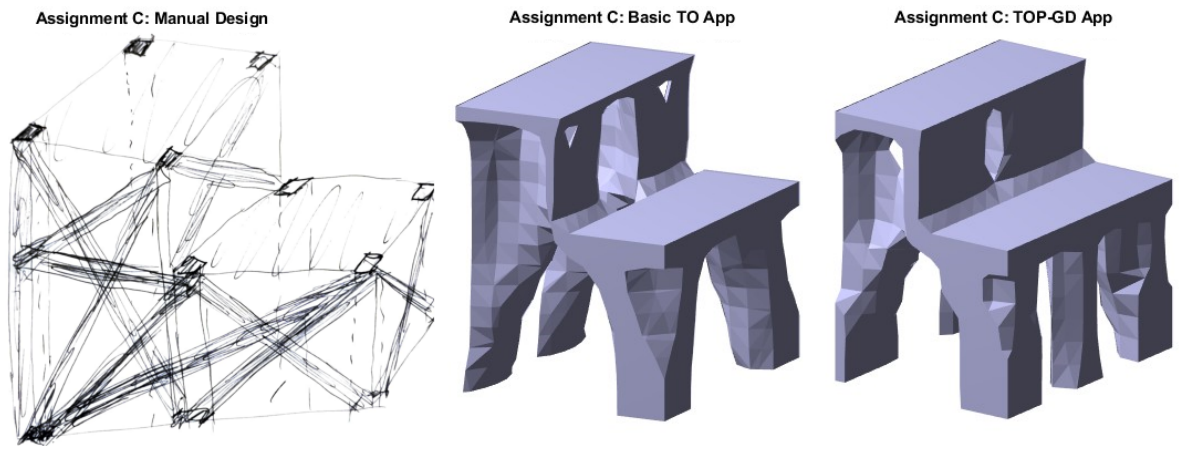


Figure 5.3: Results for Assignment C for each approach

The compliance and mass values for the solutions generated with the Basic TO and TOP-GD tool are given in Table 5.1. The lowest compliance and mass values are highlighted in green per assignment. Moreover, the compliance and mass values have been combined in a total performance score. For a simple tensile bar, half the compliance can be achieved with twice the mass. Therefore, the product of the compliance and mass values is an indication of optimality, where a lower score is better. For assignment A, both the compliance and mass values are lower for the TOP-GD solution, resulting in the lowest product score as well. For both assignment B & C, one solution has a lower mass, and the other a lower compliance value. The Mass*Compliance score is best for the Basic TO tool in Assignment B, and the TOP-GD tool scores better for assignment C. A two-tailed paired T-test shows no significant difference for the Compliance*Mass scores between the Basic TO and TOP-GD tools for the three assignments, with a p-value of 0.4226 far above the 5% significance level.

Assignment	Tool used	Compliance (J)	Mass (kg)	Compliance*Mass (J*kg)
A	Basic TO	1.786 E+13	43.96	7.851 E+14
A	TOP-GD	1260	12.39	1.561 E+4
B	Basic TO	54.99	0.077	4.234
B	TOP-GD	342.3	0.022	7.531
C	Basic TO	208.1	12.25	2549
C	TOP-GD	148.4	14.49	2150

Table 5.1: Objective Values of the Solutions Generated with the Basic TO and TOP-GD tool in the Experiment

5.2. Survey Results

An overview of the survey questions can be found in Appendix C. In most questions, the participants were asked to rate the different approaches on certain aspects with a number between 1 and 5. The results of their answers on those questions have been presented in graphs below. The scores for the overall experience of each approach in the Design Process, are shown in Figure 5.4 below. The TOP-GD approach scores best with a mean score of 4.33, followed by the Basic TO approach and lastly the Manual design approach with a mean score of 3.

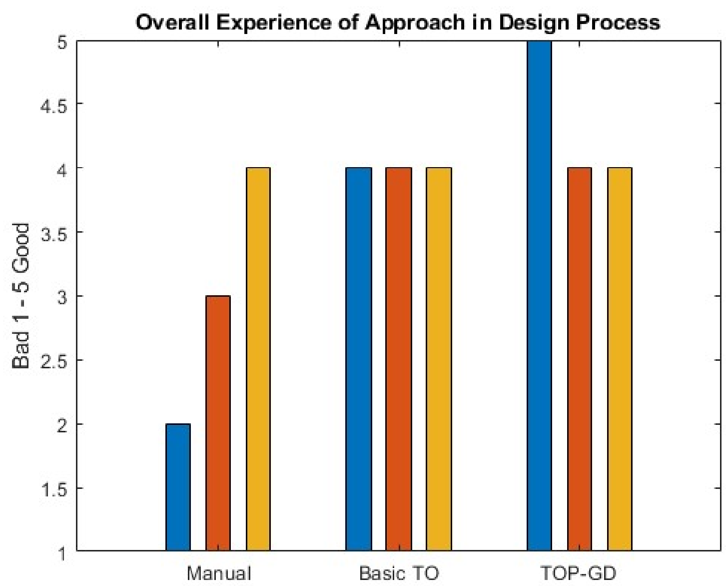


Figure 5.4: Overall Experience in Design Process per Approach

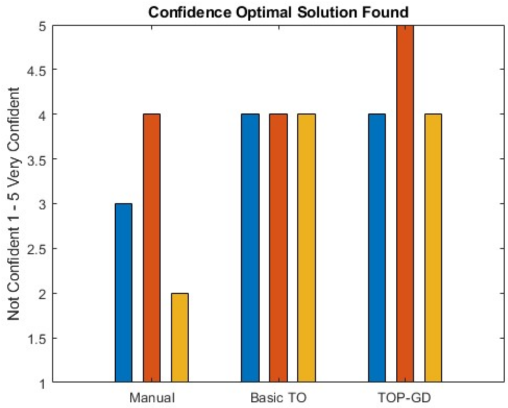


Figure 5.5: Confidence Optimal Solution found

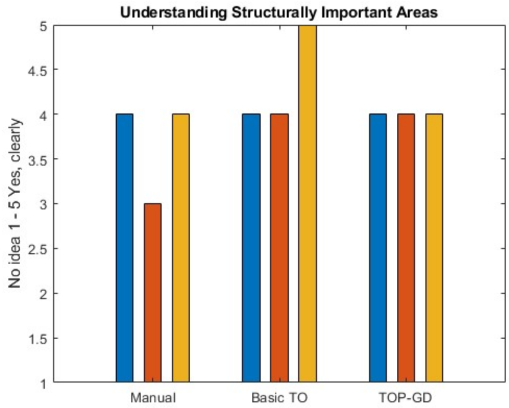


Figure 5.6: Understanding of Structurally Important Areas

Moreover, after each design assignment the participants were asked how confident they were that they found the optimal solution for the design problem. The scores per approach are shown in Figure 5.5. The TOP-GD approach gave the most confidence with a mean score of 4.33, the manual design approach the least again with a mean score of 3.

The participants were also asked after each design assignment whether they understood what were structurally the most important areas of the part to be designed. The scores per approach are shown in Figure 5.6. The Basic TO approach scored slightly better than the TOP-GD tool, mainly because of the 5-point score of 1 participant. Both tools gave a better understanding than when designing manually.

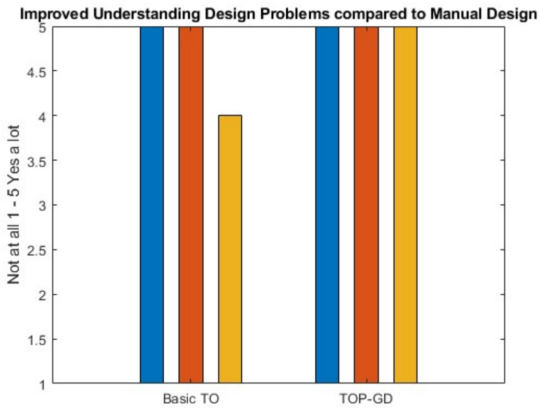


Figure 5.7: Improved Understanding of Design Problems compared to Manual Design

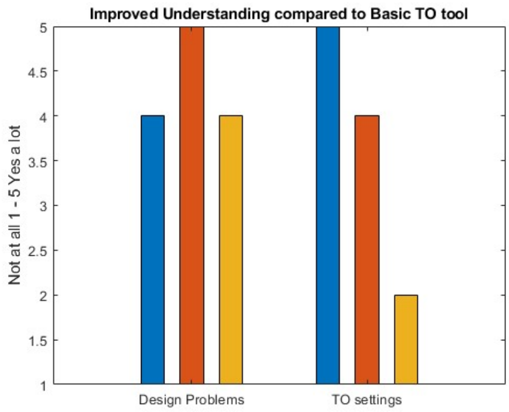


Figure 5.8: Improved Understanding compared to Basic TO tool

After the design tasks with the Basic TO tool and the TOP-GD tool, the participants were asked whether the use of this tool improved their understanding of the design problems, compared to manually designing. The scores of the answers are shown in Figure 5.7. The TOP-GD tool scored the maximum possible mean score of 5. Also the Basic TO tool considerably improved the participant's understanding of the design problems compared to manually designing, with a score of 4.67.

After the last design tasks performed with the TOP-GD tool, the participants were asked whether they thought the TOP-GD tool approach improved their understanding of Design problems compared to the Basic TO tool, and whether the TOP-GD app improved their understanding of Topology Optimization and its settings compared to the Basic TO tool. The scores of the given answers are given in Figure 5.8, with a mean of 4.33 and 3.67 for each aspect respectively.

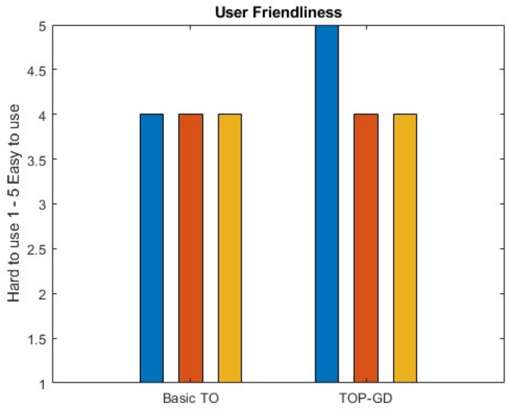


Figure 5.9: User Friendliness of Tool

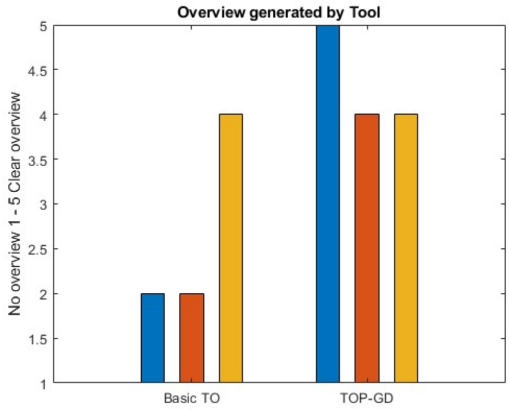


Figure 5.10: Overview generated by Tool

Moreover, the participants were asked to rate both the user friendliness of the Basic TO and the TOP-GD tool, and the Overview generated by each tool. The scores are plotted in Figure 5.9 and Figure 5.10 respectively. The TOP-GD tool was rated slightly more user-friendly than the Basic TO tool, with a mean score of 4.33 compared to 4. The overview generated by the TOP-GD tool scored a lot higher than the overview given by the Basic TO tool, with a mean score of 4.33 compared to 2.67.

Lastly, the participants were asked after the design task performed with the Basic TO tool and the TOP-GD tool, whether they considered the use of that tool in the design process as an improvement. This directly relates to the main research question. The scores of their answers are shown in Figure 5.11. Both tools were actually considered an improvement, but the TOP-GD tool scores highest with a mean score of 4.67 out of 5.

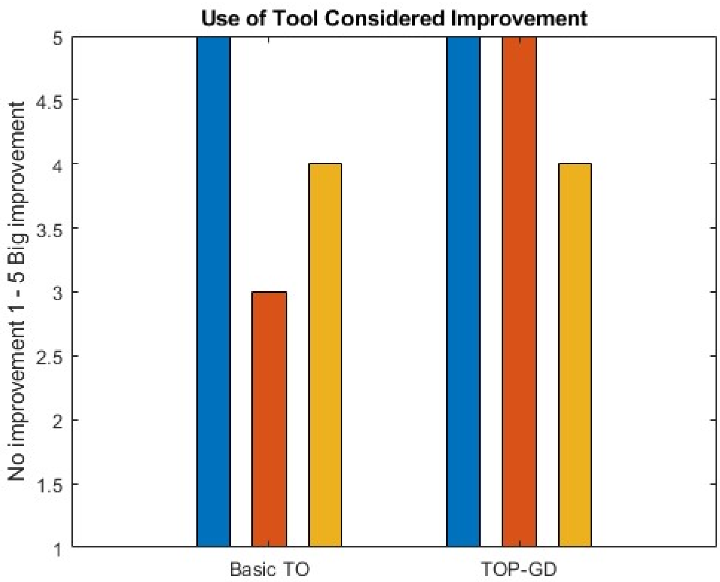


Figure 5.11: Use of tool considered Improvement in the Design Process

Besides the scoring questions, some open questions were asked in the survey as well. For each approach, the participants were asked to point out its positive and negative aspects, and what could be improved. For designing manually, the positive aspects that were mentioned are the following:

1. Manually sketching allows for fast iterations
2. As a designer, you really have to think about what is the optimal solution, and why

The negative aspects of manual design mentioned by the participants are:

1. Limitations due to drawing skills in 3D
2. Not knowing where to start
3. The tendency to get 'fixed' on a certain idea and get tunnel vision
4. Easy to overlook certain options or design directions because the design is based on previous experience

For the Basic TO tool, the positive aspects mentioned are enumerated below:

1. Harder problems can be tackled compared to manually designing (e.g. with more load cases or constraints)
2. The use of the app gives more confidence since it is backed with calculations instead of guesses
3. Not limited by drawing skills
4. Clear what needs to be done to setup a problem
5. Easy to change some settings to investigate the influence
6. The 3D handling is very user friendly
7. Node selection is very easy
8. Clear and concise naming of buttons
9. Not too many parameters you have to play with to make it work

The majority of these mentioned positive aspects can be applied to the TOP-GD tool as well, as the setup of a problem is almost the same. The negative aspects or points for improvement mentioned of the Basic TO tool are:

1. Not able to change the mesh size without losing the other settings for forces and constraints
2. Voxelise button could be automated
3. 3D space is a bit too small
4. Not able to compare different designs next to each other

The first three negative aspects also hold for the TOP-GD tool, but comparing different designs next to each other is of course something that has been integrated in the TOP-GD tool.
For the TOP-GD approach in the design process, the positive aspects mentioned are:

1. Much better understanding of influence of input parameters to the final design
2. You will not get tunnel vision on one idea, as by playing around in the tool new solutions arise (e.g. 3 legs on a stair instead of 2)
3. The tool gives fast feedback on influence of materials and mass on stiffness
4. The tool gives clear feedback on important areas in the design space
5. The Compliance-Mass graph explains a lot and helps to clearly understand the best design principles
6. Easy selection of different materials
7. Optimization method can be clearly seen looking at each iteration, which helps to understand how the tool works and what the important structural features are of the design problem

The negative aspects mentioned or points for improvement of the TOP-GD tool are:

1. The mesh size cannot be changed without needing to repeat the setup of the problem
2. Have the option to select multiple designs and compare the topology next to each other
3. Automatic extraction of nodes when selecting them
4. Have an indication of the selected range of options or amount of runs the tool is going to do (and how long that will take approximately)

Again, these aspects overlap with the functionalities of the Basic TO tool as well, but contain useful tips for the improvement of the TOP-GD tool, which will be discussed in the recommendations in Section 7.2.

5.3. Eye-tracking Results

As was explained in the previous chapter, eye-tracking was used during the experiment to asses and compare the two digital tools in the Basic TO approach and the TOP-GD approach. A lot of data was gathered with the the Eye-tracking equipment and WebLink software described in Section 4.2. This data has been processed and evaluated in MATLAB. Eye movements are typically analyzed with regard to the gaze location, fixations and saccades. Fixations are moments that the eyes stay in one position, focusing on a certain point of interest. Saccades are fast ocular movements, generally occurring when the gaze is reoriented to a new target, between fixations [52].

About 10% of the collected data was discarded due to system failures in tracking the eye position by the eye tracker. This track loss was linked to blinking and squinting by the participant, or moving of the body or head outside the trackable range of the Eyelink Portable Duo tracker. Besides that, the gaze location was sometimes tracked outside the 1920x1080 pixel range of the computer screen displaying the tools. This data has also been removed from the dataset.

First of all, a heat map could be created for both of the tools, based on the gaze location data collected during the experiments. The coloring of the heat map shows what spots of the interfaces have been looked at the most. For the Basic TO tool, the generated heat map showing the gaze location of all participants combined is shown in Figure 5.12 below. Clear highlights can be seen for the Input Parameters section, the Mesh Refinement slider, the Setup Node View and the Result view. The highlighted area in the top middle is attributed to the popup figure MATLAB shows of the evolving solution while running an optimization. More subtle highlights are visible at the Constraints, Forces, Void and Solid definition areas. To see the differences in gaze density between the participants, separate heat maps have been generated for each participant during the Basic TO design assignments, which can be found in Appendix D.

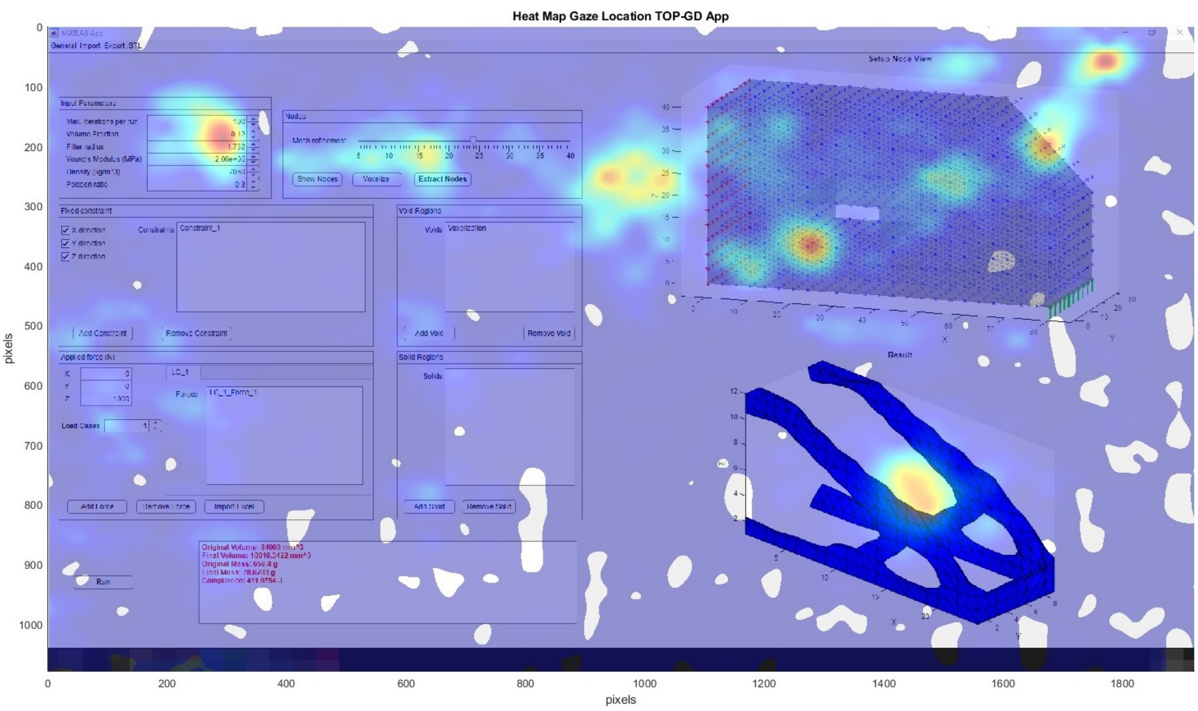


Figure 5.12: Heat Map Gaze Location Basic TO tool

A heat map summarizing the gaze location of all participants combined for the TOP-GD tool is also generated. However, this tool consists of multiple tabs that the participants have been switching between. Therefore, the gaze heat map has been plotted over the two most used tabs, the Setup tab in Figure 5.13 and the Compliance-Mass Graph tab in Figure 5.14. On the Setup tab, highlighted areas can be attributed to the Input Parameters section, the Mesh Refinement slider and the Setup Node View again. On the Compliance-Mass Graph tab, more subtle highlights are visible around the data points in the Compliance-Mass Graph, and the sensitivity and Von Mises stress plots of the selected solution shown on the right top and bottom of the screen. The highlighted area in the top middle again corresponds to the location of the popup figures MATLAB shows of the evolving solutions while running each optimization. To see the differences in gaze density between the participants, separate heat maps have been generated for each participant during the TOP-GD design assignments, which can be found in Appendix D.

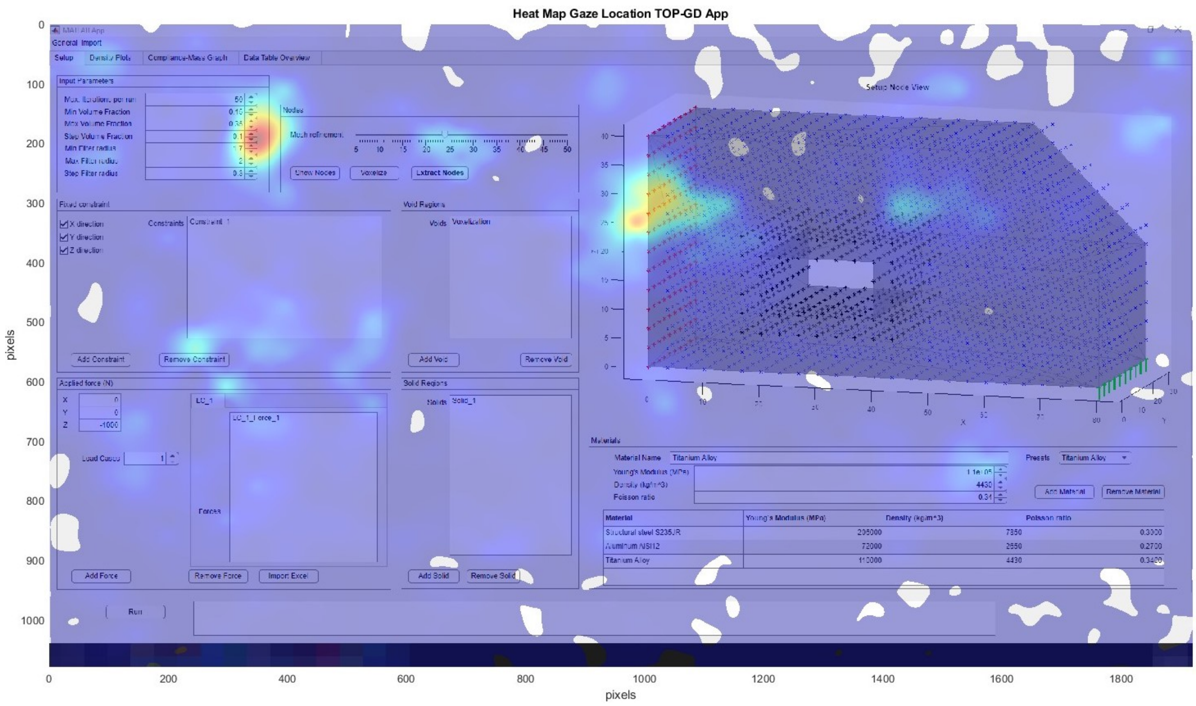


Figure 5.13: Heat Map Gaze Location TOP-GD tool setup tab

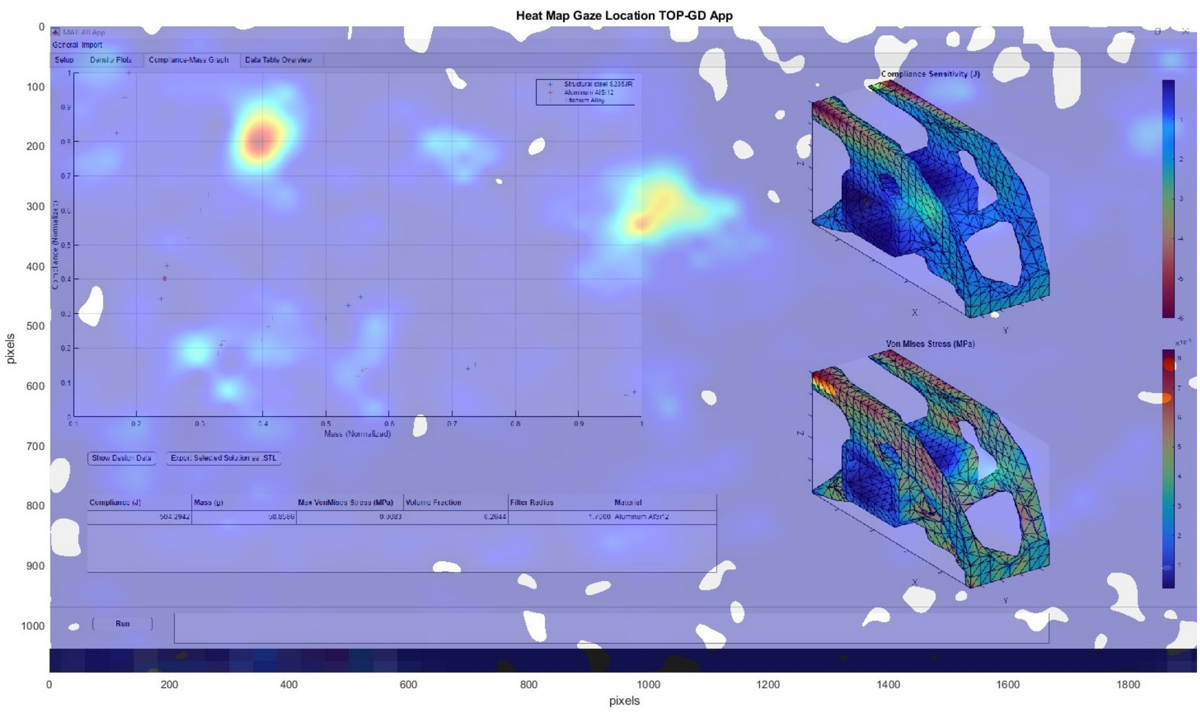


Figure 5.14: Heat Map Gaze Location TOP-GD tool Compliance-Mass Graph tab

Regarding eye fixations, especially their duration is interesting to look at. A box plot containing the data of all fixations during the experiments is shown in Figure 5.15, visualising the differences in fixation duration for each tool. Outliers have been removed of the data outside the 3 sigma (99.7%) interval.

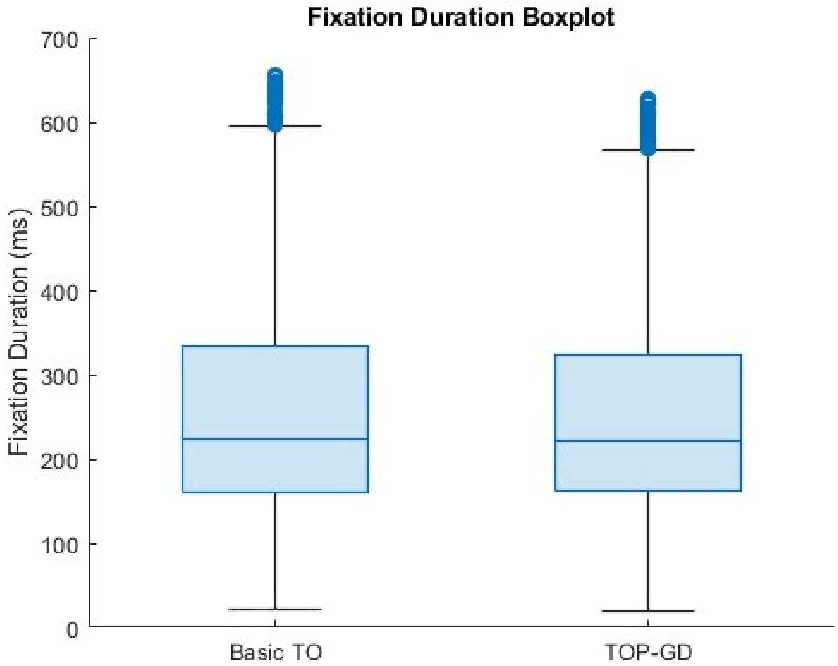


Figure 5.15: Boxplots showing the distribution of Fixation durations per tool

The mean value for the fixation duration of the Basic TO tool is 258.6 ms, with a standard deviation of 135.3 ms. For the TOP-GD tool, the mean value is 253.6 ms, and the standard deviation 126.5 ms. A fitted distribution is plotted for both tools in Figure 5.16. As can be seen in these distributions and the box plot, the mean values of the fixation duration are really close for both tools. The standard deviation differs slightly more, giving a wider distribution for the fixation duration data of the Basic TO tool. Comparing the data sets in a two-tailed unpaired T-test gives a p-value of 0.0966. This is just above the 5% significance level, which means there is no proof for a significant statistical difference between the fixation durations during the use of the two tools.

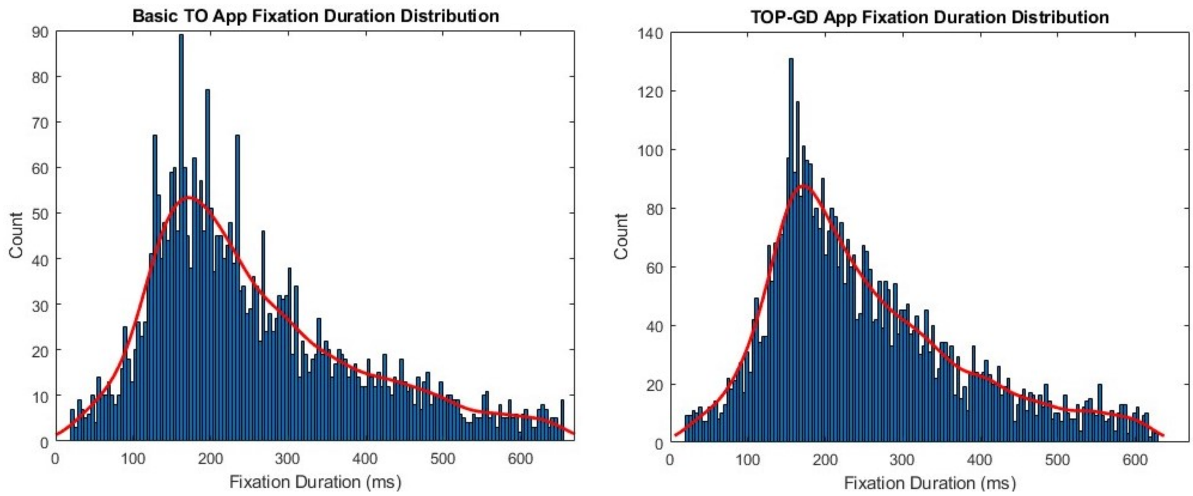


Figure 5.16: Fitted distributions of the fixation duration data for the Basic TO tool (left) and the TOP-GD tool (right)

When looking at saccades, the amplitude represents the distance travelled by a saccade during an eye movement [53] and is measured by visual degrees. A Boxplot containing the data of all saccades during the experiments is shown in Figure 5.17, visualising the differences in saccade amplitudes for each tool. Again, outliers have been removed of the data outside the 3 sigma (99.7%) interval.

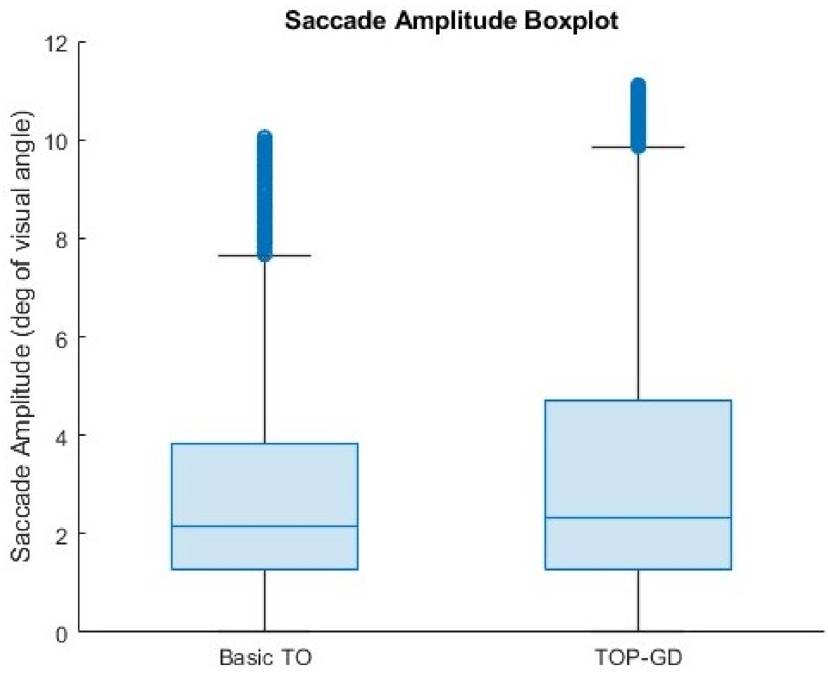


Figure 5.17: Boxplots showing the distribution of saccade amplitudes per tool

The mean value for the saccade amplitude of the Basic TO tool is 2.878 degrees, with a standard deviation of 2.160 degrees. For the TOP-GD tool, the mean value is 3.368 degrees, and the standard deviation 2.750 degrees. A fitted distribution is plotted for both tools in Figure 5.18. Both the mean and standard deviation of the saccade amplitude is slightly higher for the TOP-GD tool. Comparing the data sets in a two-tailed unpaired T-test gives a p-value of 1.784E-21. This is far below the 5% significance level, which means there is a significant statistical difference between the saccade amplitudes during the use of the two tools.

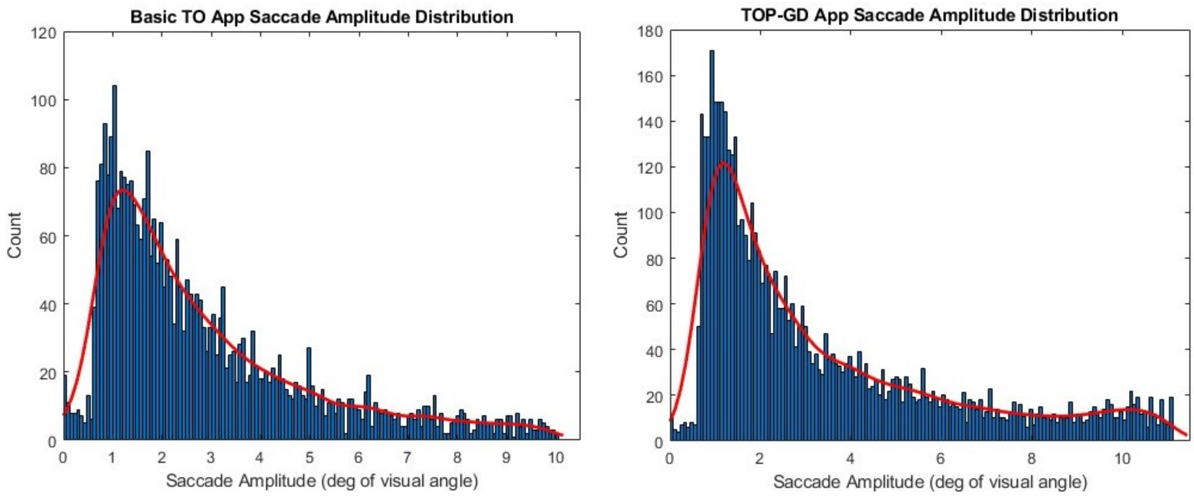


Figure 5.18: Fitted distributions of the saccade amplitude data for the Basic TO tool (left) and the TOP-GD tool (right)

The interpretation of all these results will be discussed in the next chapter.

6
Discussion

In the experiment conducted, the goal was to research the effect of different design approaches on the design performance and experience, in order to answer the main research question: *"What is the effect of using a Topology Optimization based Generative Design tool or a simple Topology Optimization tool compared to manual design on the design performance and experience?"*. This was done by giving participants simple design tasks while letting them use different designing approaches. During the experiments, different types of data have been collected, of which the results have been presented in the previous chapter. Among this data was the part performance data to assess the design performance, subjective answers to the survey questions to assess the design experience, and eye-tracking data to assess the user interaction of the Basic TO tool and the TOP-GD tool.

An important first remark to make regarding all the data of the experiment, is that only 3 engineers were available to participate due to the unfortunate bankruptcy of Lightyear. Therefore, only limited data has been collected, and any conclusion drawn from this data is statistically not significant. However, it is still interesting to look at the data that has been collected during the experiment.

6.1. Part Performance Data Interpretation

To assess the design performance, the geometries of the designed solutions are assessed visually and the Compliance and Mass values of the solutions design with the Basic TO and TOP-GD tool are compared in Section 5.1. The geometries of the solutions show that there is especially a difference between manually designed solutions and the other TO based solutions. The manually designed solutions have not been interpreted digitally to assess their compliance and mass values, since many assumptions would have to be made on their geometry. This interpreted data would therefore not be accurate or scientifically meaningful. Therefore, no clear conclusion can be drawn from a numerical comparison of their performance values. Based on their topology, it is however likely that their performance is inferior to the solutions generated with the Basic TO and TOP-GD tool.

The solutions generated by the Basic TO tool and the TOP-GD tool show more similarities, which was to be expected as both use TO to get to a result. The compliance and mass values for these TO generated solutions, shown in Table 5.1, do not show a clear and significant difference regarding the part performance. For assignment A, both the compliance and mass values are lower for the TOP-GD solution. This can however be attributed to the coarse mesh used in the Basic TO solution, resulting in some disconnected parts in the rings of the generated geometry. Mesh dependency is something that can occur in both tools, and therefore does not necessarily means an inferior performance for the Basic TO tool. Furthermore, the design tasks had to be performed within 15 minutes, which may have influenced the mesh refinement choice made by the participant, and the lack of time to redo the design. For both assignment B & C, one solution has a lower mass, and the other a lower compliance value. This shows the conflicting multi-objective trade-off designers have to choose between [1, 30]. Looking at the total score of Compliance*Mass of each solution, the Basic TO tool scores better for assignment B and the TOP-GD tool for assignment C. It is therefore hard to say, based on the part performance data, that either the TOP-GD or Basic TO approach results in an overall better part performance in this limited in experiment. There are moreover very few data points, and a T-test does not show a significant difference in the part performance values per approach either. More extensive research and experiments are needed to determine whether the different approaches have an effect on the design performance.

6.2. Survey Data Interpretation

Regarding the design experience, the answers given by the engineers in the survey show a quite consistent trend. Both the use of the Basic TO tool and the TOP-GD tool are considered an improvement in the design process compared to manually designing, and give a better overall design experience. The TOP-GD tool moreover outperforms the Basic TO tool in almost all aspects. Participants feel more confident to have found the optimal solution when using the TOP-GD tool for a design task, and besides that their understanding of both design problems and TO settings improved. The biggest difference between the Basic TO tool and the TOP-GD tool is the overview generated. This was to be expected since the Basic TO tool does not provide side by side comparisons of different solutions, and a new run overwrites the previous result. In the development process of the TOP-GD tool, a lot of focus was put on generating overview, by implementing a Pareto Compliance-Mass graph and a separate tab with a sortable table showing all the results. As was explained in Section 2.3.1, a Multi-Objective approach and showing a Pareto set helps the designer to have a trade-off overview and understand the consequences of a design decision with respect to all the relevant objectives [30]. This therefore corresponds with the results found with the survey.

Furthermore, the TOP-GD tool was rated slightly more user friendly than the Basic TO tool. This means that the extra functionalities the TOP-GD tool provides are not considered too complex, and not having them is experienced as a limitation. The setup of the problem in the TOP-GD tool only requires a few extra input settings, but is hardly more complex. The extra information given and the overview generated by the TOP-GD tool contribute to the understanding and confidence of the participants, which increases their design experience and would also explain why they consider the TOP-GD tool more user friendly. However, it should be kept in mind that the higher scoring of the TOP-GD tool on user-friendliness and overall experience can be partially due to the learning effect of the participants in the experiment. The setup of the Basic TO tool and the TOP-GD tool was kept as similar as possible. Therefore, when the participants got to the third design task with the TOP-GD tool, they already practiced the setup in the previous design task with the Basic TO tool.

The Basic TO tool only scores slightly better on the understanding of structurally important areas compared to the TOP-GD tool, due to one 5-point score of one participant. This does not follow the trend of the rest of the survey. Basic Topology Optimization does show important areas by altering the topology of the design during an optimization run, but the TOP-GD tool does the same and adding to that gives information on the compliance sensitivities by means of a coloring scheme. This higher scoring of the Basic TO tool is moreover in conflict with the other aspect scores. It could be possible that due to the chronological setup of the experiment, this participant especially felt the increase in understanding of structurally important areas of TO compared to manually designing.

The answers to the open questions further confirm that both tools improve the design experience compared to manually designing, and that the TOP-GD tool is considered to have the biggest positive effect on the design process. It is reported to generate a much better understanding of the influence of input parameters to the final design, and making Topology Optimization less of a black box approach. The tool is said to give fast feedback on the influence of materials and mass on stiffness, and the sensitivity coloring gives clear feedback on important areas in the design space. One participant explained afterwards that with this information, he could also decide early on in the design process to for example have another look at the design space, because giving more space to a certain critical narrow area will probably be more effective than optimizing a part within that narrow design space. This aligns with the remarks made in Section 2.1, where it is explained that it is important to collect as much information as soon as possible in the design process to prevent a poor concept choice by enabling early changes of the design [8], and that design decisions at the start of the design process have a bigger impact [9].

Also the advantage of Generative Design, where alternative and possibly better solutions can be generated and explored that the designer did not think of themselves [1, 3], is mentioned as a positive aspect of the TOP-GD tool in the survey. This proves that also a Topology Optimization based

Generative Design approach can have this effect, without the need for extensive cloud-based AI implementations.

However, for both tools the participants pointed out some details in the workflow that were not ideal, so there is definitely still room for improvement on user-friendliness as well.

6.3. Eye-tracking Data Interpretation

As shown in Section 5.3, the heat maps for both tools show a good distribution of the gaze location, which indicate a good use of space. However, for both the Basic TO tool (Figure 5.12) and the setup tab of the TOP-GD tool (Figure 5.13), the lower left area of the screen shows a lower gaze density which indicates it is an area of less relevance for the users. This area contains the lists of constraints, forces, passive void and solid regions, which are only needed for a part of the setup process. Although these areas do have to be present, they could have been designed occupying a smaller part of the total setup tab. In this way, more space would have been available for the 3D areas or the input parameter section that did attract more attention.

The rest of the Eye-tracking data showed a slight difference between the Basic TO tool and the TOP-GD tool. The mean fixation duration for both tools was very close with 258.6 ms and 253.6 ms respectively, which corresponds to a general fixation duration of 200-300 ms mentioned in literature [53, 54]. Also the typical positively skewered distribution of the fixation duration compared to a normal Gaussian aligns with other data and literature [53, 55, 56]. The Basic TO fixations were distributed a bit wider as shown in Figure 5.15, showing slightly more long fixations. Longer fixations could indicate deeper cognitive processing [53, 54, 57]. Shorter mean fixation duration indicate that users are spending less time looking at each item on the screen, which suggests that they are processing the information more quickly and efficiently in the TOP-GD tool. However, the difference between the mean fixation duration of the tools is very small and the two-tailed unpaired T-test showed no statistical difference within the 5% significance level. Therefore, no clear conclusion can be drawn from the differences measured in fixations of the Basic TO and TOP-GD approach.

A somewhat larger difference between the tools is shown in the distribution of the saccade amplitudes measured during the experiment. The TOP-GD tool has a larger mean saccade amplitude compared to the Basic TO tool, with a wider distribution as well. The two-tailed unpaired T-test comparing the saccade amplitudes measured during the use of both tools therefore showed a clear statistical difference within the 5% significance level. Saccade amplitudes often lower as task complexity and cognitive load increase [53, 58, 59]. The larger saccade amplitudes found with the TOP-GD tool suggest that users are moving their eyes more extensively across the screen. This can indicate that users are navigating through the TOP-GD tool's features more efficiently and may be able to complete tasks more quickly.

When looking at the combination of the two measures, an increase in fixation duration and a decrease in saccadic amplitude is said to indicate an increased task difficulty and the need to gather more fine-grained information [60]. This is consistent with what was found for both measures, although again the difference in fixation duration is too small to be a convincing difference.

Altogether, the eye-tracking data suggests that the TOP-GD tool is experienced as less complex and more effective to use than the Basic TO tool, which also aligns with the interpretation of the survey results. However, an important note should again be made regarding the effect of learning behaviour. The complexity of the design task in the TOP-GD tool can be experienced as less complex because the participants are more used to the setup and the GUI already, after completing the design task with the Basic TO tool. This can also explain the slightly lower fixation duration and higher saccade amplitude found for the TOP-GD tool during the experiment, indicating a more efficient navigation through the GUI. Nevertheless, it is clear that the TOP-GD tool does definitely not perform worse than the Basic TO tool on complexity and user-friendliness despite its extended functionalities, even if the difference in eye-tracking data is completely attributed to the learning effect. Together with the positive assessment of the TOP-GD tool by the participants in the survey, it is therefore concluded that the TOP-GD tool outperforms the Basic TO app and has the largest positive impact on the design experience.

7 Conclusion

7.1. Main Findings

The goal for this thesis was to research the effect of different design approaches on the design performance and experience, in order to answer the main research question:

“What is the effect of using a Topology Optimization based Generative Design tool or a simple Topology Optimization tool compared to manual design on the design performance and experience?”

This research was set up because of the expectation that there was potential to improve the early stages of the design process, by implementing a topology optimization based generative design approach in the form of an auxiliary tool. With such a design approach, multiple design solutions are explored quickly to study the effect of boundary conditions or numerical settings. This can help designers by giving direction and insight in trade-offs between multiple objectives, early on in the design process when design decisions still have the highest impact. In order to test the effect of this approach on the design process, a robust and user-friendly Topology Optimization based Generative Design tool had to be created. The development process to create such a tool resulted in the TOP-GD tool presented in Chapter 3. In the TOP-GD tool, multiple design solutions are explored quickly by implementing a batch-run setup that varies several chosen parameters, without needing to manually run several optimizations consecutively. Calculations are done with a simple TO script using coarse geometries, and without taking into account manufacturing methods yet. This asks for less demanding, detailed and complicated calculations than AI-based Generative Design tools currently offer, while at the same time moving from a single TO result to generating a range of candidate solutions. A lot of effort was put in the user-friendliness of the TOP-GD tool, enabling an easy workflow for the setup of design problems and a clear presentation of the results by means of a simple GUI.

The use of the TOP-GD tool in the design process was evaluated in an experiment, where it was compared with a more simple TO tool and a basic manual design approach using just pen and paper. This was done by giving the participants of the experiments three simple design assignments, that they had to carry out using each of the design approaches one by one. Evaluation of the approaches was done in threefold. First of all, the design performance was assessed by visually inspecting the geometry of the designed solutions, and comparing the mass and compliance values of the solutions generated with the Basic TO and TOP-GD tools. The design experience of the participants was mapped with an extensive survey, asking them to judge the different approaches on numerous aspects. Lastly, the user interaction of the Basic TO and TOP-GD tool was assessed using Eye-tracking techniques, by looking at gaze location data, fixations and saccades. The results of this experiment showed no clear or significant difference in the design performance between the solutions designed with the TOP-GD tool and the Basic TO tool. For the design experience however, a clear difference was found between all approaches. The manual design approach was outperformed by both the Basic TO approach and the TOP-GD approach, on all aspects, which was expected. Moreover, the TOP-GD approach scored higher on almost all aspects assessed compared to the Basic TO approach. The TOP-GD approach is rated as more user-friendly, helps to better understand design problems and the influence of topology optimization settings, and especially improved the overview generated during the design process. The

TOP-GD approach also gave the best overall experience in the design process, and its implementation in the design process was considered a big improvement. This was further substantiated with the data gathered with Eye-tracking. A slightly smaller mean fixation duration was found for the TOP-GD tool, however this difference is not statistically significant when analyzed with a T-test. Besides that, a larger mean saccade amplitude was found for the TOP-GD tool, which is a clear significant difference according to the T-test. This indicates that the participants navigated through the tool more efficiently, and experienced a lower task complexity. However, it is also possible that these differences are due to the learning effect experienced by the participants during the experiment. Therefore further analysis is required in a more extensive experiment to determine the exact cause of these differences.

To come back to the main research question; there was not enough evidence to conclude that the TOP-GD approach had a significant effect on the design performance compared to the Basic TO approach for the simple assignments executed during the experiment. However, the results of the survey show a clear positive impact of both the TO tools on the design experience compared to manually designing. Furthermore, the TOP-GD tool has the largest positive impact on the design experience and its use in the design process is considered a big improvement, especially in quickly exploring new design directions and creating overview. This confirms the expectation that a Topology Optimization based Generative Design approach has a positive effect on the early stages of the design process. The small differences found with Eye-tracking between the TO tools support this, although more research should be done to convincingly confirm this with Eye-tracking data.

7.2. Recommendations

First of all, the experiment performed during this thesis with only 3 participants, was unfortunately limited. As was mentioned as first remark in the discussion, therefore the validity of any conclusions drawn is questionable and could be improved by gathering more data through the repetition of the experiment with more test subjects.

Regarding the TOP-GD tool, there is still room for improvement as well, even though it is already considered to be user-friendly and have a positive effect on the design process. The Eye-tracking data showed that the space used by the elements in the GUI is not always proportionate to the gaze density in that location. The layout of the GUI could therefore be iterated to give more space to elements that get more attention, and tested more extensively with Eye-tracking experiments.

For further development of the TOP-GD tool's capabilities, for example natural frequency optimization and stress constraints functionalities can be added next to the standard compliance minimization functionality present, to enable the exploration of more complex and realistic problems that appear in the engineering industry. Next to that, the diversity of the solutions could also be improved by varying the starting points of the different optimization runs, by for example implementing stochastic techniques instead of only varying input parameters like the filter radius and the volume fraction. Moreover, it would be interesting to add manufacturing constraints. Even though this is not a requirement in a first exploration in the earliest stages of the design process, it is still of added value to have the possibility to look for easily manufacturable solutions.

Furthermore, the experiment showed that the workflow of the TOP-GD tool can still be improved as well. Even though the app was already experienced as easy to use, minor changes such as automating the voxelization of the geometry, or the automatic extraction of nodes after selecting them, would make the GUI handling even more intuitive. Besides that, a big limitation was experienced by the participants due to the inability to change the refinement of the mesh after the problem had been set up. This is not possible yet in this version of the TOP-GD tool, because the node numbering that is defined with the mesh refinement slider is used to define all other aspects of the problem setup. However, it is undoubtedly possible with more development time to enable the transformation of all constraints, forces, passive solid and void elements to another node numbering system if the mesh refinement is changed at a later point in the problem setup.

And lastly, with some more development time, the TOP-GD tool could be extended in such a way that it becomes more 'intelligent'. This includes giving more predictive feedback, such as a range of volume fractions that are deemed interesting to explore, or providing the user with an indication of solving time. Moreover, the tool could be programmed differently to explore the design space first in a rough manner with big parameter steps, and then with increasing detail explore the most interesting parameter combinations.

References

[1] Daria Vlah, Roman Žavbi, and Nikola Vukašinović. “Evaluation of topology optimization and generative design tools as support for conceptual design”. In: *Proceedings of the design society: DESIGN conference*. Vol. 1. Cambridge University Press. 2020, pp. 451–460.

[2] Uwe Schramm and Ming Zhou. “Recent developments in the commercial implementation of topology optimization”. In: *IUTAM symposium on topological design optimization of structures, machines and materials*. Springer. 2006, pp. 239–248.

[3] Francesco Buonamici et al. “Generative design: an explorative study”. In: *Computer-Aided Design and Applications* 18.1 (2020), pp. 144–155.

[4] Martin Philip Bendsøe and Noboru Kikuchi. “Generating optimal topologies in structural design using a homogenization method”. In: *Computer methods in applied mechanics and engineering* 71.2 (1988), pp. 197–224.

[5] Martin P Bendsøe and Ole Sigmund. “Material interpolation schemes in topology optimization”. In: *Archive of applied mechanics* 69.9 (1999), pp. 635–654.

[6] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2003.

[7] Evangelos Tyflopoulos et al. “State of the art of generative design and topology optimization and potential research needs”. In: *DS 91: Proceedings of NordDesign 2018, Linköping, Sweden, 14th-17th August 2018* (2018).

[8] Andreas Wilhelm Meyer, Sándor Vajna, et al. “Support of searching for solutions by automated structural optimization”. In: *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*. 2018, pp. 369–380.

[9] Lihui Wang et al. “Collaborative conceptual design—state of the art and future trends”. In: *Computer-aided design* 34.13 (2002), pp. 981–996.

[10] Emma S. Vercoulen. “Improving the Engineering Design Process by using Computational Design Techniques”. In: (2022). Literature Research.

[11] Madara Ogot and Gul Kremer. *Engineering design: a practical guide*. Trafford Publishing, 2004.

[12] Yousef Haik, Sangarappillai Sivaloganathan, and Tamer M Shahin. *Engineering design process*. Third Edition. Cengage Learning, 2015.

[13] Chao Wang et al. “A comprehensive review of educational articles on structural and multidisciplinary optimization”. In: *Structural and Multidisciplinary Optimization* 64.5 (2021), pp. 2827–2880.

[14] Ole Sigmund and Kurt Maute. “Topology optimization approaches”. In: *Structural and Multidisciplinary Optimization* 48.6 (2013), pp. 1031–1055.

[15] Stanley Osher and James A Sethian. “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *Journal of computational physics* 79.1 (1988), pp. 12–49.

[16] Haipeng Jia et al. “Evolutionary level set method for structural topology optimization”. In: *Computers & Structures* 89.5-6 (2011), pp. 445–454.

[17] Osvaldo M Querin, Grant P Steven, and Yi Min Xie. “Evolutionary structural optimisation (ESO) using a bidirectional algorithm”. In: *Engineering computations* (1998).

[18] XY Yang et al. “Bidirectional evolutionary method for stiffness optimization”. In: *AIAA journal* 37.11 (1999), pp. 1483–1488.

[19] Xiaodong Huang and Yi-Min Xie. “Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method”. In: *Finite elements in analysis and design* 43.14 (2007), pp. 1039–1049.

[20] Shun Wang. “Krylov subspace methods for topology optimization on adaptive meshes”. In: (2007).

[21] Hongbo Sun and Ling Ma. “Generative design by using exploration approaches of reinforcement learning in density-based structural topology optimization”. In: *Designs* 4.2 (2020), p. 10.

[22] M Zhou, YK Shyy, and HL Thomas. “Checkerboard and minimum member size control in topology optimization”. In: *Structural and Multidisciplinary Optimization* 21 (2001), pp. 152–158.

[23] Susana Rojas-Labanda and Mathias Stolpe. “Benchmarking optimization solvers for structural topology optimization”. In: *Structural and Multidisciplinary Optimization* 52.3 (2015), pp. 527–547.

[24] GIN Rozvany and Ming Zhou. “The COC algorithm, part I: Cross-section optimization or sizing”. In: *Computer Methods in Applied Mechanics and Engineering* 89.1-3 (1991), pp. 281–308.

[25] M Zhou and GIN Rozvany. “The COC algorithm, Part II: Topological, geometrical and generalized shape optimization”. In: *Computer methods in applied mechanics and engineering* 89.1-3 (1991), pp. 309–336.

[26] Krister Svanberg. “The method of moving asymptotes—a new method for structural optimization”. In: *International journal for numerical methods in engineering* 24.2 (1987), pp. 359–373.

[27] Krister Svanberg. “A class of globally convergent optimization methods based on conservative convex separable approximations”. In: *SIAM journal on optimization* 12.2 (2002), pp. 555–573.

[28] Raphael T. Haftka and Zafer Gürdal. *Elements of structural optimization*. Third revised and expanded edition. Springer Science & Business Media, 1992.

[29] Ole Sigmund. “A 99 line topology optimization code written in Matlab”. In: *Structural and multidisciplinary optimization* 21.2 (2001), pp. 120–127.

[30] Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. “Pareto multi objective optimization”. In: *Proceedings of the 13th international conference on, intelligent systems application to power systems*. IEEE. 2005, pp. 84–91.

[31] Long Tang et al. “Adaptive heuristic search algorithm for discrete variables based multi-objective optimization”. In: *Structural and Multidisciplinary Optimization* 48.4 (2013), pp. 821–836.

[32] Kalyanmoy Deb. “Multi-objective optimization”. In: *Search methodologies*. Springer, 2014, pp. 403–449.

[33] Danny J Lohan, Ercan M Dede, and James T Allison. “Topology optimization for heat conduction using generative design algorithms”. In: *Structural and Multidisciplinary Optimization* 55.3 (2016), pp. 1063–1077.

[34] Sivam Krish. “A practical generative design method”. In: *Computer-Aided Design* 43.1 (2011), pp. 88–100.

[35] Loris Barbieri and Maurizio Muzzupappa. “Performance-Driven Engineering Design Approaches Based on Generative Design and Topology Optimization Tools: A Comparative Study”. In: *Applied Sciences* 12.4 (2022), p. 2106.

[36] Autodesk. *Autodesk Fusion 360 Generative Design tool*. URL: `https://www.autodesk.com/solutions/generative-design` (visited on 10/07/2022).

[37] Filippo A Salustri, Nathan L Eng, and Janaka S Weerasinghe. “Visualizing information in the early stages of engineering design”. In: *Computer-Aided Design and Applications* 5.5 (2008), pp. 697–714.

[38] Evangelos Tyflopoulos and Martin Steinert. “A Comparative Study of the Application of Different Commercial Software for Topology Optimization”. In: *Applied Sciences* 12.2 (2022), p. 611.

[39] Federico Ferrari and Ole Sigmund. “A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D”. In: *Structural and Multidisciplinary Optimization* 62 (2020), pp. 2211–2228.

[40] Z88. *Z88Arion*. URL: `https://en.z88.de/z88arion/` (visited on 07/12/2022).

[41] Ernesto Aranda, José Carlos Bellido, and Alberto Donoso. “Toptimiz3D: A topology optimization software using unstructured meshes”. In: *Advances in Engineering Software* 148 (2020), p. 102875.

[42] Erik Andreassen et al. “Efficient topology optimization in MATLAB using 88 lines of code”. In: *Structural and Multidisciplinary Optimization* 43 (2011), pp. 1–16.

[43] MathWorks. *stlread*. URL: `https://nl.mathworks.com/help/matlab/ref/stlread.html` (visited on 04/26/2023).

[44] Daniel Cohen-Or and Arie Kaufman. “Fundamentals of surface voxelization”. In: *Graphical models and image processing* 57.6 (1995), pp. 453–461.

[45] Adam H. Aitkenhead. *MathWorks File Exchange*. 2013. URL: `https://nl.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation` (visited on 04/26/2023).

[46] Xinchang Zhang, Wenyuan Cui, and Frank Liou. “Voxel-based geometry reconstruction for repairing and remanufacturing of metallic components via additive manufacturing”. In: *International Journal of Precision Engineering and Manufacturing-Green Technology* (2021), pp. 1–24.

[47] Hao Deng, Praveen S Vulimiri, and Albert C To. “An efficient 146-line 3D sensitivity analysis code of stress-based topology optimization written in MATLAB”. In: *Optimization and Engineering* (2021), pp. 1–29.

[48] Robert JK Jacob and Keith S Karn. “Eye tracking in human-computer interaction and usability research: Ready to deliver the promises”. In: *The mind’s eye*. Elsevier, 2003, pp. 573–605.

[49] Asus. *Asus ROG Zephyrus S GX531*. URL: `https://rog.asus.com/nl/laptops/rog-zephyrus/rog-zephyrus-s-gx531-series/spec/` (visited on 04/25/2023).

[50] SR Research. *EyeLink Portable Duo*. URL: `https://www.sr-research.com/eyelink-portable-duo/` (visited on 04/25/2023).

[51] SR Research. *WebLink*. URL: `https://www.sr-research.com/weblink/` (visited on 04/25/2023).

[52] Dario D Salvucci and Joseph H Goldberg. “Identifying fixations and saccades in eye-tracking protocols”. In: *Proceedings of the 2000 symposium on Eye tracking research & applications*. 2000, pp. 71–78.

[53] Bhanuka Mahanama et al. “Eye movement and pupil measures: A review”. In: *Frontiers in Computer Science* 3 (2022), p. 733531.

[54] Keith Rayner. “Eye movements in reading and information processing.” In: *Psychological bulletin* 85.3 (1978), p. 618.

[55] Boris M Velichkovsky et al. “Visual fixations and level of attentional processing”. In: *Proceedings of the 2000 symposium on eye tracking research & applications*. 2000, pp. 79–85.

[56] Adrian Staub and Ashley Benatar. “Individual differences in fixation duration distributions in reading”. In: *Psychonomic Bulletin & Review* 20 (2013), pp. 1304–1311.

[57] Timothy A Salthouse and Cecil L Ellis. “Determinants of eye-fixation duration”. In: *The American journal of psychology* (1980), pp. 207–234.

[58] Matthew H Phillips and Jay A Edelman. “The dependence of visual scanning performance on search direction and difficulty”. In: *Vision research* 48.21 (2008), pp. 2184–2192.

[59] James G May et al. “Eye movement indices of mental workload”. In: *Acta psychologica* 75.1 (1990), pp. 75–89.

[60] Canan Karatekin. “Eye tracking studies of normative and atypical development”. In: *Developmental Review* 27.3 (2007). Developmental Cognitive Neuroscience, pp. 283–348. ISSN: 0273-2297. DOI: `https://doi.org/10.1016/j.dr.2007.06.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0273229707000238`.

A
TOP-GD tool Source Code

The complete source code of the created TOP-GD tool during this thesis is annexed in this appendix. The tool was made in the MATLAB App Designer environment.

```
classdef TOPGD_App < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                      matlab.ui.Figure
        GeneralMenu                   matlab.ui.container.Menu
        SaveSetupMenu                 matlab.ui.container.Menu
        ResetAllMenu                  matlab.ui.container.Menu
        ImportMenu                    matlab.ui.container.Menu
        ImportGeometryMenu            matlab.ui.container.Menu
        ImportSetupMenu               matlab.ui.container.Menu
        RunButton                     matlab.ui.control.Button
        TextArea                      matlab.ui.control.TextArea
        TabGroup                      matlab.ui.container.TabGroup
        SetupTab                      matlab.ui.container.Tab
        MaterialsPanel                matlab.ui.container.Panel
        RemoveMaterialButton          matlab.ui.control.Button
        AddMaterialButton             matlab.ui.control.Button
        UITableMaterials              matlab.ui.control.Table
        PresetsDropDown               matlab.ui.control.DropDown
        PresetsDropDownLabel          matlab.ui.control.Label
        MaterialNameEditField         matlab.ui.control.EditField
        MaterialNameEditFieldLabel    matlab.ui.control.Label
        PoissonratioSpinner           matlab.ui.control.Spinner
        PoissonratioSpinner_2Label    matlab.ui.control.Label
        DensitySpinner                matlab.ui.control.Spinner
        Densitykgm3SpinnerLabel       matlab.ui.control.Label
        YoungsModulusSpinner          matlab.ui.control.Spinner
        YoungsModulusMPaSpinnerLabel  matlab.ui.control.Label
        FixedconstraintPanel          matlab.ui.container.Panel
        RemoveConstraintButton        matlab.ui.control.Button
        ConstraintsListBox            matlab.ui.control.ListBox
        ConstraintsListBoxLabel       matlab.ui.control.Label
        AddConstraintButton           matlab.ui.control.Button
        ZdirectionCheckBox            matlab.ui.control.CheckBox
        YdirectionCheckBox            matlab.ui.control.CheckBox
        XdirectionCheckBox            matlab.ui.control.CheckBox
        AppliedforceNPanel            matlab.ui.container.Panel
        ImportExcelButton             matlab.ui.control.Button
        LoadCasesSpinner              matlab.ui.control.Spinner
        LoadcasesLabel                matlab.ui.control.Label
        TabGroupLC                    matlab.ui.container.TabGroup
        LC_1Tab                       matlab.ui.container.Tab
        ForcesListBox                 matlab.ui.control.ListBox
        ForcesListBoxLabel            matlab.ui.control.Label
        ZEditField                    matlab.ui.control.NumericEditField
        ZEditFieldLabel               matlab.ui.control.Label
```

```
        YEditField                      matlab.ui.control.NumericEditField
        YEditFieldLabel                 matlab.ui.control.Label
        RemoveForceButton               matlab.ui.control.Button
        XEditField                      matlab.ui.control.NumericEditField
        XEditFieldLabel                 matlab.ui.control.Label
        AddForceButton                  matlab.ui.control.Button
        SolidRegionsPanel               matlab.ui.container.Panel
        RemoveSolidButton               matlab.ui.control.Button
        AddSolidButton                  matlab.ui.control.Button
        SolidsListBox                   matlab.ui.control.ListBox
        SolidsListBoxLabel              matlab.ui.control.Label
        VoidRegionsPanel                matlab.ui.container.Panel
        RemoveVoidButton                matlab.ui.control.Button
        AddVoidButton                   matlab.ui.control.Button
        VoidsListBox                    matlab.ui.control.ListBox
        VoidsListBoxLabel               matlab.ui.control.Label
        InputParametersPanel            matlab.ui.container.Panel
        StepFilterRadiusSpinner         matlab.ui.control.Spinner
        FilterradiusLabel_3             matlab.ui.control.Label
        StepVolFracSpinner              matlab.ui.control.Spinner
        VolumeFractionLabel_3           matlab.ui.control.Label
        FilterRadiusSpinner_2           matlab.ui.control.Spinner
        FilterradiusLabel_2             matlab.ui.control.Label
        VolFracSpinner_2                matlab.ui.control.Spinner
        VolumeFractionLabel_2           matlab.ui.control.Label
        FilterRadiusSpinner             matlab.ui.control.Spinner
        FilterradiusLabel               matlab.ui.control.Label
        MaxIterationsperrunSpinner      matlab.ui.control.Spinner
        MaxIterationsperrunSpinnerLabel  matlab.ui.control.Label
        VolFracSpinner                  matlab.ui.control.Spinner
        VolumeFractionLabel             matlab.ui.control.Label
        NodesPanel                      matlab.ui.container.Panel
        MeshrefinementSlider            matlab.ui.control.Slider
        MeshrefinementSliderLabel       matlab.ui.control.Label
        VoxelizeButton                  matlab.ui.control.Button
        ExtractNodesButton              matlab.ui.control.Button
        ShowNodesButton                 matlab.ui.control.Button
        UIAxesNodes                     matlab.ui.control.UIAxes
        DensityPlotsTab                 matlab.ui.container.Tab
        ComplianceMassGraphTab          matlab.ui.container.Tab
        ExportSelectedSolutionasSTLButton  matlab.ui.control.Button
        ShowDesignDataButton            matlab.ui.control.Button
        UITableSelection                matlab.ui.control.Table
        UIAxesSelection_2               matlab.ui.control.UIAxes
        UIAxesSelection                 matlab.ui.control.UIAxes
        UIAxes                          matlab.ui.control.UIAxes
        DataTableOverview               matlab.ui.container.Tab
        ExportSelectedSolutionasSTLButton_2  matlab.ui.control.Button
        UITableData                     matlab.ui.control.Table
        UIAxesTableSelection_2          matlab.ui.control.UIAxes
        UIAxesTableSelection            matlab.ui.control.UIAxes
    end

    properties (Access = public)
        Volume
        Materials
        stlfile
        nelx
        nely
        nelz
        nel
        sz
        maxit
        nodes
        nodeNrs
        elemNrs
        Snodes
        actnod
        Constr
        Forces
        Voids
```

```
        Solids
        geom
        grid   %plots
        extracted
        selected
        con
        fcs
        sol
        arrow
        LCtabs
        Data
        Result
        DataTable
        T
        FT
        MatCheck
        FRCheck
        VFCheck
        axess
        axesL
        LocTab
        LocNodPlot
        LocPan
        DTab
        PosFTab
        maxmass
        maxcomp
        resplot
    end

    methods (Access = public)

        function ResetAll(app)
            delete(app.UIAxesNodes.Children)
            delete(app.DensityPlotsTab.Children)
            delete(app.UIAxes.Children)
            delete(app.UIAxesSelection.Children)
            delete(app.UIAxesSelection_2.Children)
            delete(app.UIAxesTableSelection.Children)
            delete(app.UIAxesTableSelection_2.Children)
            app.UITableMaterials.Data = [];
            app.UITableSelection.Data = [];
            app.UITableData.Data = [];
            app.TextArea.Value = '';
            app.VoidsListBox.Items = {};
            app.ConstraintsListBox.Items = {};
            app.ForcesListBox.Items = {};
            app.SolidsListBox.Items = {};
            if app.LoadCasesSpinner.Value > 1
                for i = 2:length(app.TabGroupLC.Children)
                    delete(app.TabGroupLC.Children(end))
                end
            end
            app.LoadCasesSpinner.Value = 1;
            app.XdirectionCheckBox.Value = 0;
            app.YdirectionCheckBox.Value = 0;
            app.ZdirectionCheckBox.Value = 0;
            app.XEditField.Value = 0;
            app.YEditField.Value = 0;
            app.ZEditField.Value = 0;
            app.MaxIterationsperrunSpinner.Value = 50;
            app.VolFracSpinner.Value = 0.15;
            app.VolFracSpinner_2.Value = 0.35;
            app.StepVolFracSpinner.Value = 0.1;
            app.FilterRadiusSpinner.Value = 1.7;
            app.FilterRadiusSpinner_2.Value = 2;
            app.StepFilterRadiusSpinner.Value = 0.3;
            app.YoungsModulusSpinner.Value = 206000;
            app.DensitySpinner.Value = 7850;
            app.PoissonratioSpinner.Value = 0.3;
            app.MeshrefinementSlider.Value = 10;
```

```
            app.Volume = [];
            app.stlfile = [];
            app.nelx = app.MeshrefinementSlider.Value;
            app.nely = [];
            app.nelz = [];
            app.nel = [];
            app.sz = [];
            app.geom = [];
            app.nodes = [];
            app.nodeNrs = [];
            app.elemNrs = [];
            app.Snodes = [];
            startupFcn(app)
        end

        function [result, perf, rdcc, risovals, VonMisesMax, vonmises] = TOPGD_TO(app,nelx,
            nely,nelz,volfrac,penal,rmin,ft,ftBC,eta,beta,move,E0,nu,maxit,pasV,pasS,fixed,
            lcDof,Fl)
            % ---------------------------- PRE. 1) MATERIAL AND CONTINUATION PARAMETERS
            q = 0.5;                                                               %
                Stress Relaxation factor
            Emin = 1e-9;                                                           %
                Young modulus of "void"
            penalCnt = { 1, 1, 25, 0.25 };                                         %
                continuation scheme on penal
            betaCnt  = { 1, 1, 25,    2 };                                         %
                continuation scheme on beta
            if ftBC == 'N', bcF = 'symmetric'; else, bcF = 0; end                  %
                filter BC selector
            % ----------------------------------------- PRE. 2) DISCRETIZATION FEATURES
            nEl = nelx * nely * nelz;                                              %
                number of elements          #3D#
            NodeNrs = int32(app.nodeNrs);                                      % nodes
                numbering             #3D#
            cVec = reshape( 3 * NodeNrs( 1 : nely, 1 : nelz, 1 : nelx ) + 1, nEl, 1 ); %
                                            #3D#
            cMat = cVec+int32( [0,1,2,3*(nely+1)*(nelz+1)+[0,1,2,-3,-2,-1],-3,-2,-1,3*(nely
                +...
               1)+[0,1,2],3*(nely+1)*(nelz+2)+[0,1,2,-3,-2,-1],3*(nely+1)+[-3,-2,-1]]);%
                   connectivity matrix          #3D#
            nDof = ( 1 + nely ) * ( 1 + nelz ) * ( 1 + nelx ) * 3;                 %
                total number of DOFs        #3D#
            [ sI, sII ] = deal( [ ] );
            for j = 1 : 24
                sI = cat( 2, sI, j : 24 );
                sII = cat( 2, sII, repmat( j, 1, 24 - j + 1 ) );
            end
            [ iK , jK ] = deal( cMat( :,  sI )', cMat( :, sII )' );
            Iar = sort( [ iK( : ), jK( : ) ], 2, 'descend' ); clear iK jK          %
                reduced assembly indexing
            Ke = 1/(1+nu)/(2*nu-1)/144 *( [ -32;-6;-6;8;6;6;10;6;3;-4;-6;-3;-4;-3;-6;10;...
                3;6;8;3;3;4;-3;-3; -32;-6;-6;-4;-3;6;10;3;6;8;6;-3;-4;-6;-3;4;-3;3;8;3;...
                3;10;6;-32;-6;-3;-4;-3;-3;4;-3;-6;-4;6;6;8;6;3;10;3;3;8;3;6;10;-32;6;6;...
                -4;6;3;10;-6;-3;10;-3;-6;-4;3;6;4;3;3;8;-3;-3;-32;-6;-6;8;6;-6;10;3;3;4;...
                -3;3;-4;-6;-3;10;6;-3;8;3;-32;3;-6;-4;3;-3;4;-6;3;10;-6;6;8;-3;6;10;-3;...
                3;8;-32;-6;6;8;6;-6;8;3;-3;4;-3;3;-4;-3;6;10;3;-6;-32;6;-6;-4;3;3;8;-3;...
                3;10;-6;-3;-4;6;-3;4;3;-32;6;3;-4;-3;-3;8;-3;-6;10;-6;-6;8;-6;-3;10;-32;...
                6;-6;4;3;-3;8;-3;3;10;-3;6;-4;3;-6;-32;6;-3;10;-6;-3;8;-3;3;4;3;3;-4;6;...
                -32;3;-6;10;3;-3;8;6;-3;10;6;-6;8;-32;-6;6;8;6;-6;10;6;-3;-4;-6;3;-32;6;...
                -6;-4;3;6;10;-3;6;8;-6;-32;6;3;-4;3;3;4;3;6;-4;-32;6;-6;-4;6;-3;10;-6;3;...
                -32;6;-6;8;-6;-6;10;-3;-32;-3;6;-4;-3;3;4;-32;-6;-6;8;6;6;-32;-6;-6;-4;...
                -3;-32;-6;-3;-4;-32;6;6;-32;-6;-32]+nu*[ 48;0;0;0;-24;-24;-12;0;-12;0;...
                24;0;0;0;24;-12;-12;0;-12;0;0;-12;12;12;48;0;24;0;0;0;-12;-12;-24;0;-24;...
                0;0;24;12;-12;12;0;-12;0;-12;-12;0;48;24;0;0;12;12;-12;0;24;0;-24;-24;0;...
                0;-12;-12;0;0;-12;-12;0;-12;48;0;0;0;-24;0;-12;0;12;-12;12;0;0;0;-24;...
                -12;-12;-12;-12;0;0;48;0;24;0;-24;0;-12;-12;-12;-12;12;0;0;24;12;-12;0;...
                0;-12;0;48;0;24;0;-12;12;-12;0;-12;-12;24;-24;0;12;0;-12;0;0;-12;48;0;0;...
                0;-24;24;-12;0;0;-12;12;-12;0;0;-24;-12;-12;0;48;0;24;0;0;0;-12;0;-12;...
                -12;0;0;0;-24;12;-12;-12;48;-24;0;0;0;0;-12;12;0;-12;24;24;0;0;12;-12;...
                48;0;0;-12;-12;12;-12;0;0;-12;12;0;0;0;24;48;0;12;-12;0;0;-12;0;-12;-12;...
                -12;0;0;-24;48;-12;0;-12;0;0;-12;0;12;-12;-24;24;0;48;0;0;0;-24;24;-12;...
```

```
        0;12;0;24;0;48;0;24;0;0;0;-12;12;-24;0;24;48;-24;0;0;-12;-12;-12;0;-24;...
        0;48;0;0;0;-24;0;-12;0;-12;48;0;24;0;24;0;-12;12;48;0;-24;0;12;-12;-12;...
        48;0;0;0;-24;-24;48;0;24;0;0;48;24;0;0;48;0;0;48;0;48 ] );          % elemental stiffness matrix  #3D#
    Ke0( tril( ones( 24 ) ) == 1 ) = Ke';
    Ke0 = reshape( Ke0, 24, 24 );
    Ke0 = Ke0 + Ke0' - diag( diag( Ke0 ) );                                 % recover full matrix

    D = 1./((1+nu)*(1-2*nu))*[1-nu nu nu 0 0 0; nu 1-nu nu 0 0 0;...        % elastic matrix formulation
        nu nu 1-nu 0 0 0; 0 0 0 (1-2*nu)/2 0 0; 0 0 0 0 (1-2*nu)/2 0;...
        0 0 0 0 0 (1-2*nu)/2];

    B_1=[-0.044658,0,0,0.044658,0,0,0.16667,0                               % strain matrix formulation
    0,-0.044658,0,0,-0.16667,0,0,0.16667
    0,0,-0.044658,0,0,-0.16667,0,0
    -0.044658,-0.044658,0,-0.16667,0.044658,0,0.16667,0.16667
    0,-0.044658,-0.044658,0,-0.16667,-0.16667,0,-0.62201
    -0.044658,0,-0.044658,-0.16667,0,0.044658,-0.62201,0];
    B_2=[0,-0.16667,0,0,-0.16667,0,0,0.16667
    0,0,0.044658,0,0,-0.16667,0,0
    -0.62201,0,0,-0.16667,0,0,0.044658,0
    0,0.044658,-0.16667,0,-0.16667,-0.16667,0,-0.62201
    0.16667,0,-0.16667,0.044658,0,0.044658,-0.16667,0
    0.16667,-0.16667,0,-0.16667,0.044658,0,-0.16667,0.16667];
    B_3=[0,0,0.62201,0,0,-0.62201,0,0
    -0.62201,0,0,0.62201,0,0,0.16667,0
    0,0.16667,0,0,0.62201,0,0,0.16667
    0.16667,0,0.62201,0.62201,0,0.16667,-0.62201,0
    0.16667,-0.62201,0,0.62201,0.62201,0,0.16667,0.16667
    0,0.16667,0.62201,0,0.62201,0.16667,0,-0.62201];

    B=[B_1,B_2,B_3];

    % ----------------------------- PRE. 3) LOADS, SUPPORTS AND PASSIVE DOMAINS
    F = [];
    for i=1:size(Fl,2)
        FF = fsparse(cell2mat(lcDof(i)), 1, cell2mat(Fl(i)), [nDof, 1] );
        F = [F FF];                                                         % Define Loads
    end
    free = setdiff( 1 : nDof, fixed );                                      % set of free DOFs
    act = setdiff( ( 1 : nEl )', union( pasS, pasV ) );                     % set of active d.v.
    % --------------------------------------- PRE. 4) DEFINE IMPLICIT FUNCTIONS
    prj = @(v,eta,beta) (tanh(beta*eta)+tanh(beta*(v(:)-eta)))./...
        (tanh(beta*eta)+tanh(beta*(1-eta)));                                % projection
    deta = @(v,eta,beta) - beta * csch( beta ) .* sech( beta * ( v( : ) - eta ) ).^2 .* ...
        sinh( v( : ) * beta ) .* sinh( ( 1 - v( : ) ) * beta );             % projection eta-derivative
    dprj = @(v,eta,beta) beta*(1-tanh(beta*(v-eta)).^2)./(tanh(beta*eta)+tanh(beta*(1-eta)));% proj. x-derivative
    cnt = @(v,vCnt,l) v+(l>=vCnt{1}).*(v<vCnt{2}).*(mod(l,vCnt{3})==0).*vCnt{4};
    % ----------------------------------------------------- PRE. 5) PREPARE FILTER
    [dy,dz,dx]=meshgrid(-ceil(rmin)+1:ceil(rmin)-1,...
        -ceil(rmin)+1:ceil(rmin)-1,-ceil(rmin)+1:ceil(rmin)-1 );
    h = max( 0, rmin - sqrt( dx.^2 + dy.^2 + dz.^2 ) );                     % conv. kernel                #3D#
    Hs = imfilter( ones( nely, nelz, nelx ), h, bcF );                      % matrix of weights (filter)  #3D#
    dHs = Hs;
    % ------------------------ PRE. 6) ALLOCATE AND INITIALIZE OTHER PARAMETERS
    [ x, dsK, dV ] = deal( zeros( nEl, 1 ) );                               % initialize vectors
    dV( act, 1 ) = 1/nEl/volfrac;                                           % derivative of volume
```

```
            x( act ) = ( volfrac*( nEl - length(pasV) ) - length(pasS) )/length( act );% volume fraction on active set
            x( pasS ) = 1;                                                           % set x = 1 on pasS set
            [ xPhys, xOld, ch, loop, U ] = deal( x, 1, 1, 0, zeros( nDof, size(Fl,2) ) ); % old x, x change, it. counter, U
            % ================================================ START OPTIMIZATION LOOP
            figure
            h1 = axes;
            set(h1,'xdir','reverse')
            while ch > 1e-6 && loop < maxit
              loop = loop + 1;                                                       % update iter. counter
              % ----------- RL. 1) COMPUTE PHYSICAL DENSITY FIELD (AND ETA IF PROJECT.)
              xTilde = imfilter( reshape( x, nely, nelz, nelx ), h, bcF ) ./ Hs;      % filtered field            #3D#
              xPhys( act ) = xTilde( act );                                          % reshape to column vector
              if ft > 1                                                              % compute optimal eta* with Newton
                  f = ( mean( prj( xPhys, eta, beta ) ) - volfrac )  * (ft == 3);    % function (volume)
                  while abs( f ) > 1e-6                                              % Newton process for finding opt. eta
                      eta = eta - f / mean( deta( xPhys, eta, beta ) );
                      f = mean( prj( xPhys, eta, beta ) ) - volfrac;
                  end
                  dHs = Hs ./ reshape( dprj( xPhys, eta, beta ), nely, nelz, nelx );  % sensitivity modification    #3D#
                  xPhys = prj( xPhys, eta, beta );                                   % projected (physical) field
              end
              ch = norm( xPhys - xOld ) ./ nEl;
              xOld = xPhys;
              % ------------------------- RL. 2) SETUP AND SOLVE EQUILIBRIUM EQUATIONS
              sK = ( Emin + xPhys.^penal * ( E0 - Emin ) );
              dsK( act ) = -penal * ( E0 - Emin ) * xPhys( act ) .^ ( penal - 1 );
              sK = reshape( Ke( : ) * sK', length( Ke ) * nEl, 1 );
              K = fsparse( Iar( :, 1 ), Iar( :, 2 ), sK, [ nDof, nDof ] );
              L = chol( K( free, free ), 'lower' );
              U( free , : ) = L' \ ( L \ F( free , : ) );                            % f/b substitution
              % ----------------------------------------- RL. 3) COMPUTE SENSITIVITIES
              C = 0;
              dc = 0;
              for i = 1:size(Fl,2)
                  Ui = U(:,i);
                  Fi = F(:,i);
                  C = C + Fi'*Ui;
                  dc = dc + dsK .* sum( ( Ui( cMat ) * Ke0 ) .* Ui( cMat ), 2 );     % derivative of compliance
              end
              dc = imfilter( reshape( dc, nely, nelz, nelx ) ./ dHs, h, bcF );        % filter objective sens.     #3D#
              dV0 = imfilter( reshape( dV, nely, nelz, nelx ) ./ dHs, h, bcF );       % filter compliance sens.    #3D#
              % ----------------- RL. 4) UPDATE DESIGN VARIABLES AND APPLY CONTINUATION
              xT = x( act );
              [ xU, xL ] = deal( xT + move, xT - move );                             % current upper and lower bound
              ocP = xT .* sqrt( - dc( act ) ./ dV0( act ) );                         % constant part in resizing rule
              l = [ 0, mean( ocP ) / volfrac ];                                      % initial estimate for LM
              while ( l( 2 ) - l( 1 ) ) / ( l( 2 ) + l( 1 ) ) > 1e-4                 % OC resizing rule
                  lmid = 0.5 * ( l( 1 ) + l( 2 ) );
                  x( act ) = max( max( min( min( ocP / lmid, xU ), 1 ), xL ), 0 );
                  if mean( x ) > volfrac, l( 1 ) = lmid; else, l( 2 ) = lmid; end
              end
```

```
            [penal,beta] = deal(cnt(penal,penalCnt,loop), cnt(beta,betaCnt,loop));   %
                apply conitnuation on parameters
            % -------------------------- RL. 5) PRINT CURRENT RESULTS AND PLOT DESIGN

            V=mean(xPhys(:));
            fprintf( 'It.:%5i C:%6.5e V:%7.3f ch.:%0.2e penal:%7.2f beta:%7.1f eta:%7.2f lm
                :%0.2e \n', ...
                loop, C, V, ch, penal, beta, eta, lmid );
            isovals = shiftdim( reshape( xPhys, nely, nelz, nelx ), 2 );
            isovals = smooth3( isovals, 'box', 1 );
            dcc = smooth3(shiftdim(dc,2));
            sur = isosurface(isovals, .5);
            cap = isocaps(isovals, .5);
            p = patch(sur);
            cp = patch(cap);
            isonormals(isovals,p)
            isonormals(isovals,cp)
            isocolors(dcc,p)
            isocolors(dcc,cp)
            tur = turbo;
            tur = flipud(tur);
            colormap(tur)
            p.FaceColor = 'interp';
            cp.FaceColor = 'interp';
            drawnow; view( [ 145, 25 ] ); axis equal tight off;

            if loop == maxit || ch <= 1e-6
                result.faces = [sur.faces; cap.faces+length(sur.vertices(:,1))];
                result.vertices = [sur.vertices; cap.vertices];
                rdcc = dcc;
                risovals = isovals;
                rp = patch(result);
                isonormals(isovals,rp)
                isocolors(dcc,rp)
                colormap(tur)
                rp.FaceColor = 'interp';
                perf.C = C;
                perf.V = V;

                MISES=zeros(nEl,size(Fl,2)); %von Mises stress vector
                misesmax=zeros(nEl,1);
                for j = 1:size(Fl,2)
                    Uj = U(:,j);
                    for i=1:nEl
                        temp=xPhys(i)^q*(D*B*Uj(cMat(i,:)))';
                        MISES(i,j)=sqrt(0.5*((temp(1)-temp(2))^2+(temp(1)-temp(3))^2....
                        +(temp(2)-temp(3))^2+6*sum(temp(4:6).^2)));
                        if abs(MISES(i,j)) > abs(misesmax(i))
                          misesmax(i)=MISES(i,j);
                        end
                    end
                end
                VonMisesMax = max(MISES,[],'all');
                vonmises = shiftdim(reshape(misesmax,nely,nelz,nelx),2);
            else
                cla();
            end
          end
      end
  end

  methods (Access = private)

      function FcslistboxValueChanged(app,~,~)
          delete(app.selected)
          if ~isempty(app.ConstraintsListBox.Items)
              app.ConstraintsListBox.Value = {};
          end
          if ~isempty(app.VoidsListBox.Items)
              app.VoidsListBox.Value = {};
          end
```

```
            if ~isempty(app.SolidsListBox.Items)
                app.SolidsListBox.Value = {};
            end

            LC = find(strcmp(string(app.TabGroupLC.SelectedTab.Title),cat(1,app.LCtabs.Title)
                ));

            allforces = cat(1,app.Forces.lc);
            jndex = find(allforces==LC);
            lcforces = app.Forces(jndex);

            if LC == 1
                index = find(ismember(app.ForcesListBox.Items, app.ForcesListBox.Value));
            else
                index = find(ismember(app.LCtabs(LC).lbx.Items, app.LCtabs(LC).lbx.Value));
            end

            app.selected = plot3(app.UIAxesNodes,app.nodes(lcforces(index).nodes,1),app.nodes
                (lcforces(index).nodes,2),app.nodes(lcforces(index).nodes,3),'m*');
        end

        function ImportMyGeometry(app, filename)
           data = stlread(filename);
           delete(app.geom)
           gray = [.6 .6 .6];
           app.geom = trisurf(data,'Parent',app.UIAxesNodes,'FaceAlpha',0.5,'edgecolor',gray,
               'facecolor',gray);
           axis(app.UIAxesNodes,'equal')
           hold(app.UIAxesNodes,'on')
           model = createpde;
           importGeometry(model,filename);
           mesh = generateMesh(model);
           app.Volume = volume(mesh); %in mm^3, *10^-9 voor m^3
        end

        function [result] = GetResultFromSelectedDataPoint(app)
            cla(app.UIAxesSelection)
            cla(app.UIAxesSelection_2)
            tur = turbo;
            tur = flipud(tur);
            app.TextArea.Value = "";
            for k = 1:length(app.Materials)
                Y = get(app.resplot(k), 'YData');
                brush = get(app.resplot(k), 'BrushData');
                if ~isempty(Y(logical(brush)))
                    by = Y(logical(brush));
                    by = round(by*app.maxcomp);
                end
            end
            if length(by)==1
                [I1,~,I3] = ind2sub(size(app.Data(:,2,:)),find(ismember(round(app.Data(:,2,:)
                    ),by)));
                result.faces = app.Result(I1).faces;
                result.vertices = app.Result(I1).vertices;
                dcc = app.Result(I1).dcc;
                isovals = app.Result(I1).isovals;
                vonmises = app.Result(I1).vonmises;

                xlabel(app.UIAxesSelection, 'Y')
                ylabel(app.UIAxesSelection, 'X')
                zlabel(app.UIAxesSelection, 'Z')
                set(app.UIAxesSelection,'XTickLabel',[])
                set(app.UIAxesSelection,'YTickLabel',[])
                set(app.UIAxesSelection,'ZTickLabel',[])
                set(app.UIAxesSelection,'xdir','reverse');
                title(app.UIAxesSelection,'Compliance Sensitivity (J)')

                rp = patch(app.UIAxesSelection, result);
                isonormals(isovals,rp)
                isocolors(dcc,rp)
                rp.FaceColor = 'interp';
```

```
                colormap(app.UIAxesSelection,tur)
                colorbar(app.UIAxesSelection)
                view(app.UIAxesSelection, [ 145, 25 ]);
                axis(app.UIAxesSelection, "equal");

                xlabel(app.UIAxesSelection_2, 'Y')
                ylabel(app.UIAxesSelection_2, 'X')
                zlabel(app.UIAxesSelection_2, 'Z')
                set(app.UIAxesSelection_2,'XTickLabel',[])
                set(app.UIAxesSelection_2,'YTickLabel',[])
                set(app.UIAxesSelection_2,'ZTickLabel',[])
                set(app.UIAxesSelection_2,'xdir','reverse');
                title(app.UIAxesSelection_2,'Von Mises Stress (MPa)')

                rps = patch(app.UIAxesSelection_2, result);
                isonormals(isovals,rps)
                isocolors(vonmises,rps)
                rps.FaceColor = 'interp';
                colormap(app.UIAxesSelection_2,'turbo')
                colorbar(app.UIAxesSelection_2)
                view(app.UIAxesSelection_2, [ 145, 25 ]);
                axis(app.UIAxesSelection_2, "equal");

                tabresult=struct();
                tabresult.comp = app.Data(I1,2,I3);
                tabresult.mass = app.Data(I1,4,I3);
                tabresult.stress = app.Data(I1,5,I3);
                tabresult.VF = app.Data(I1,1,I3);
                tabresult.fr = app.Data(I1,3,I3);
                tabresult.mat = app.Materials(I3).name;

                app.UITableSelection.Data = struct2table(tabresult);
            else
                app.TextArea.Value = "Please select a single data point";
                result=0;
            end
        end
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.maxit = app.MaxIterationsperrunSpinner.Value;
            app.Constr = [];
            app.Forces = [];
            app.Voids = [];
            app.Solids = [];
            app.actnod = [];
            app.LCtabs = [];
            app.Materials = [];
            tab1 = struct();
            tab1.Tab = app.LC_1Tab;
            tab1.Title = app.LC_1Tab.Title;
            tab1.lbx = app.ForcesListBox;
            tab1.lbl = app.ForcesListBoxLabel;
            app.LCtabs = tab1;
            app.axess = [];
            app.DataTable = [];
            app.UITableData.SelectionType = 'row';
            app.UITableData.ColumnSortable = true;
            close all
        end

        % Value changed function: MeshrefinementSlider
        function MeshrefinementSliderValueChanged(app, event)
            app.nelx = round(app.MeshrefinementSlider.Value);
            app.MeshrefinementSlider.Value = app.nelx;
        end
```

```

        % Button pushed function: ShowNodesButton
        function ShowNodesButtonPushed(app, event)
            app.Voids = [];
            app.actnod = [];
            app.Constr = [];
            app.Forces = [];
            app.Voids = [];
            app.Solids = [];
            app.VoidsListBox.Items = {};
            app.ConstraintsListBox.Items = {};
            app.SolidsListBox.Items = {};
            for i=1:app.LoadCasesSpinner.Value
                app.LCtabs(i).lbx.Items = {};
            end

            [stlcoords] = READ_stl(app.stlfile);
            xco = squeeze( stlcoords(:,1,:) )';
            yco = squeeze( stlcoords(:,2,:) )';
            zco = squeeze( stlcoords(:,3,:) )';

            app.nelx = round(app.MeshrefinementSlider.Value);

            app.sz = (max(xco,[],'all')-min(xco,[],'all'))/app.nelx;
            app.nely = round((max(yco,[],'all')-min(yco,[],'all'))/app.sz);
            app.nelz = round((max(zco,[],'all')-min(zco,[],'all'))/app.sz);
            app.nel = app.nelx*app.nely*app.nelz;
            app.nodeNrs = reshape(1:(1+app.nelx)*(1+app.nely)*(1+app.nelz),1+app.nely,1+app.
                nelz,1+app.nelx);
            app.elemNrs = reshape(1:(app.nelx)*(app.nely)*(app.nelz),app.nely,app.nelz,app.
                nelx);

            app.nodes = zeros((app.nelx+1)*(app.nely+1)*(app.nelz+1),3);
            n=1;
            for i=min(xco,[],'all'):app.sz:max(xco,[],'all')
                for k=(min(zco,[],'all')):app.sz:((min(zco,[],'all'))+(app.sz*app.nelz))
                    for j=(min(yco,[],'all')):app.sz:(min(yco,[],'all')+(app.sz*app.nely))
                        app.nodes(n,1)=i;
                        app.nodes(n,2)=j;
                        app.nodes(n,3)=k;
                        n=n+1;
                    end
                end
            end

            if ~isempty(app.grid)
                delete(app.grid)
            end

            app.actnod = (1:length(app.nodes))';
            app.grid=plot3(app.UIAxesNodes,app.nodes(:,1),app.nodes(:,2),app.nodes(:,3),'bx')
                ;
        end

        % Button pushed function: VoxelizeButton
        function VoxelizeButtonPushed(app, event)
            if ismember('Voxelization',app.VoidsListBox.Items) == 0
                app.actnod=[];
                void=struct();
                [OUTPUTgrid] = VOXELISE(app.nelx,app.nely,app.nelz,app.stlfile,'xyz');
                void.elem = zeros(1,(app.nel-sum(OUTPUTgrid,'all')));
                void.nodes = [];
                m = 0;
                n = 0;
                for i=1:app.nelx
                    for k=1:app.nelz
                        for j=1:app.nely
                            n = n+1;
                            [y,z,x] = ind2sub(size(app.elemNrs),find(app.elemNrs == n));
                            if OUTPUTgrid(i,j,k) == 0
                                m = m+1;
```

```
                                void.elem(1,m) = n;
                                void.nodes = unique([void.nodes; app.nodeNrs(y,z,x); app.nodeNrs(y+1,z,x); app.nodeNrs(y,z+1,x); app.nodeNrs(y+1,z+1,x); app.nodeNrs(y,z,x+1); app.nodeNrs(y+1,z,x+1); app.nodeNrs(y,z+1,x+1); app.nodeNrs(y+1,z+1,x+1)]);
                            else
                                app.actnod = unique([app.actnod; app.nodeNrs(y,z,x); app.nodeNrs(y+1,z,x); app.nodeNrs(y,z+1,x); app.nodeNrs(y+1,z+1,x); app.nodeNrs(y,z,x+1); app.nodeNrs(y+1,z,x+1); app.nodeNrs(y,z+1,x+1); app.nodeNrs(y+1,z+1,x+1)]);
                            end
                        end
                    end
                end

                void.nodes = setdiff(void.nodes,app.actnod);

                if isempty(void.nodes)
                    app.TextArea.Value = 'No elements to exclude for this geometry...';
                else
                    void.name='Voxelization';
                    app.VoidsListBox.Items{end+1} = void.name;

                    if isempty(app.Voids)
                            app.Voids=void;
                    else
                        app.Voids=[app.Voids; void];
                    end

                    delete(app.grid)
                    app.grid = plot3(app.UIAxesNodes,app.nodes(app.actnod,1),app.nodes(app.actnod,2),app.nodes(app.actnod,3),'bx');
                end
            else
                app.TextArea.Value = 'Already voxelized... If mesh refinement has changed, first push Show Nodes button again';
            end
        end

        % Button pushed function: ExtractNodesButton
        function ExtractNodesButtonPushed(app, event)
            if ~isempty(app.nodes)
                if ~isempty(app.extracted)
                    delete(app.extracted)
                end

                X = get(app.grid, 'XData');
                Y = get(app.grid, 'YData');
                Z = get(app.grid, 'ZData');
                brush = get(app.grid, 'BrushData');
                bx = X(logical(brush));
                by = Y(logical(brush));
                bz = Z(logical(brush));

                app.Snodes=zeros(length(bx),1);

                for i=1:length(bx)
                    app.Snodes(i,1) = find(ismember(app.nodes,[bx(i) by(i) bz(i)],"rows"));
                end
                app.extracted = plot3(app.UIAxesNodes,app.nodes(app.Snodes,1),app.nodes(app.Snodes,2),app.nodes(app.Snodes,3),'y*');
            else
                app.TextArea.Value = 'First Show Nodes...';
            end
        end

        % Button pushed function: AddConstraintButton
        function AddConstraintButtonPushed(app, event)
            if ~isempty(app.Snodes)
                if ~(app.XdirectionCheckBox.Value==0 && app.YdirectionCheckBox.Value == 0 && app.ZdirectionCheckBox.Value == 0)
```

```
                    app.TextArea.Value = '';
                    delete(app.extracted)
                    delete(app.con)
                    constr=struct();
                    constr.nodes=app.Snodes(:,1);

                    cn = 1;
                    while ismember(strcat('Constraint_',num2str(cn)), app.ConstraintsListBox.
                        Items)==1
                        cn=cn+1;
                    end

                    constr.name=strcat('Constraint_',num2str(cn));
                    app.ConstraintsListBox.Items{end+1} = constr.name;
                    constr.xyz = [0 0 0];

                    if app.XdirectionCheckBox.Value == 1
                        constr.xyz(1) = 1;
                    end

                    if app.YdirectionCheckBox.Value == 1
                        constr.xyz(2) = 1;
                    end

                    if app.ZdirectionCheckBox.Value == 1
                        constr.xyz(3) = 1;
                    end

                    if isempty(app.Constr)
                        app.Constr=constr;
                    else
                        app.Constr=[app.Constr; constr];
                    end

                    allconstr = cat(1,app.Constr.nodes);

                    app.con = plot3(app.UIAxesNodes,app.nodes(allconstr,1),app.nodes(
                        allconstr,2),app.nodes(allconstr,3),'r*');
                    app.Snodes=[];
                else
                    app.TextArea.Value = 'Select at least 1 fixed direction...';
                end
            else
                app.TextArea.Value = 'First extract nodes...';
            end
        end

        % Value changed function: ConstraintsListBox
        function ConstraintsListBoxValueChanged(app, event)
            index = find(ismember(app.ConstraintsListBox.Items, app.ConstraintsListBox.Value)
                );
            delete(app.selected)

            if ~isempty(app.ForcesListBox.Items)
                app.ForcesListBox.Value = {};
            end
            if ~isempty(app.VoidsListBox.Items)
                app.VoidsListBox.Value = {};
            end
            if ~isempty(app.SolidsListBox.Items)
                app.SolidsListBox.Value = {};
            end

            app.selected = plot3(app.UIAxesNodes,app.nodes(app.Constr(index).nodes,1),app.
                nodes(app.Constr(index).nodes,2),app.nodes(app.Constr(index).nodes,3),'m*');
        end

        % Button pushed function: RemoveConstraintButton
        function RemoveConstraintButtonPushed(app, event)
            [~,idx] = ismember(app.ConstraintsListBox.Value,app.ConstraintsListBox.Items);
            app.ConstraintsListBox.Items(idx) = [];
```

```
            app.Constr(idx)=[];
            delete(app.con)
            delete(app.selected)

            allconstr = cat(1,app.Constr.nodes);
            app.con = plot3(app.UIAxesNodes,app.nodes(allconstr,1),app.nodes(allconstr,2),app
                .nodes(allconstr,3),'r*');
        end

        % Value changed function: LoadCasesSpinner
        function LoadCasesSpinnerValueChanged(app, event)
            value = app.LoadCasesSpinner.Value;

            if value > length(app.LCtabs)
                for i = (length(app.TabGroupLC.Children)+1):value
                        tab = struct();
                        tab.Tab = uitab(app.TabGroupLC,'Title',['LC_' num2str(i)],'
                            AutoResizeChildren','off','SizeChangedFcn',@LC_1TabSizeChanged);
                        tab.Title = strcat('LC_', num2str(i));
                        tab.lbx=uilistbox(tab.Tab,'Position',[66,8,(app.PosFTab.w-81),(app.
                            PosFTab.h-14)],'Items',{},'ValueChangedFcn',@app.
                            FcslistboxValueChanged);
                        tab.lbl=uilabel(tab.Tab,'Text','Forces','Position',[9,82,42,22],'
                            HorizontalAlignment','right');
                        app.LCtabs = [app.LCtabs; tab];
                end
            elseif value < length(app.LCtabs)
                for i = value+1:length(app.TabGroupLC.Children)
                    delete(app.TabGroupLC.Children(end))
                    app.LCtabs(end) = [];
                    if ~isempty(app.Forces)
                        allforces = cat(1,app.Forces.lc);
                        index = find(allforces==i);
                        app.Forces(index) = [];
                    end
                end
            end
            TabGroupLCSelectionChanged(app,event)
        end

        % Value changed function: ForcesListBox
        function ForcesListBoxValueChanged(app, event)
            delete(app.selected)

            if ~isempty(app.ConstraintsListBox.Items)
                app.ConstraintsListBox.Value = {};
            end
            if ~isempty(app.VoidsListBox.Items)
                app.VoidsListBox.Value = {};
            end
            if ~isempty(app.SolidsListBox.Items)
                app.SolidsListBox.Value = {};
            end

            LC = find(strcmp(string(app.TabGroupLC.SelectedTab.Title),cat(1,app.LCtabs.Title)
                ));

            allforces = cat(1,app.Forces.lc);
            jndex = find(allforces==LC);
            lcforces = app.Forces(jndex);

            if LC == 1
                index = find(ismember(app.ForcesListBox.Items, app.ForcesListBox.Value));
            else
                index = find(ismember(app.LCtabs(LC).lbx.Items, app.LCtabs(LC).lbx.Value));
            end

            app.selected = plot3(app.UIAxesNodes,app.nodes(lcforces(index).nodes,1),app.nodes
                (lcforces(index).nodes,2),app.nodes(lcforces(index).nodes,3),'m*');
        end

```

```
        % Button pushed function: AddForceButton
        function AddForceButtonPushed(app, event)
            if ~isempty(app.Snodes)
                if app.XEditField.Value == 0 && app.YEditField.Value == 0 && app.ZEditField.
                    Value == 0
                    app.TextArea.Value = 'Magnitude of Force should not be 0';
                else

                    app.TextArea.Value = '';

                    force=struct();
                    force.nodes=app.Snodes(:,1);
                    force.lc = find(strcmp(string(app.TabGroupLC.SelectedTab.Title),cat(1,app
                        .LCtabs.Title)));

                    cn = 1;
                    while ismember(strcat('LC_',num2str(force.lc),'_Force_',num2str(cn)), app
                        .LCtabs(force.lc).lbx.Items)==1
                        cn=cn+1;
                    end

                    force.name=strcat('LC_',num2str(force.lc),'_Force_',num2str(cn));
                    app.LCtabs(force.lc).lbx.Items{end+1} = force.name;

                    lcDofx=[];
                    lcDofy=[];
                    lcDofz=[];

                    if ~app.XEditField.Value == 0
                        lcDofx=(3*force.nodes)-2;
                        force.U = app.XEditField.Value*ones(length(force.nodes),1);
                    else
                        force.U = zeros(length(force.nodes),1);
                    end

                    if ~app.YEditField.Value == 0
                        lcDofy=(3*force.nodes)-1;
                        force.V = app.YEditField.Value*ones(length(force.nodes),1);
                    else
                        force.V = zeros(length(force.nodes),1);
                    end

                    if ~app.ZEditField.Value == 0
                        lcDofz=3*force.nodes;
                        force.W = app.ZEditField.Value*ones(length(force.nodes),1);
                    else
                        force.W = zeros(length(force.nodes),1);
                    end

                    force.lcDof = [lcDofx;lcDofy;lcDofz];
                    force.Fl = nonzeros([force.U/length(force.U); force.V/length(force.V);
                        force.W/length(force.W)]);

                    if isempty(app.Forces)
                        app.Forces=force;
                    else
                        app.Forces=[app.Forces; force];
                    end

                    TabGroupLCSelectionChanged(app, event)

                    app.Snodes=[];

                end
            else
                app.TextArea.Value = 'First extract nodes...';
            end
        end

        % Button pushed function: RemoveForceButton
        function RemoveForceButtonPushed(app, event)
```

```
            LC = find(strcmp(string(app.TabGroupLC.SelectedTab.Title),cat(1,app.LCtabs.Title)
                ));

            allforces = cat(1,app.Forces.lc);
            jndex = find(allforces==LC);
            lcforces = app.Forces(jndex);

            if LC == 1
                index = find(ismember(app.ForcesListBox.Items, app.ForcesListBox.Value));
            else
                index = find(ismember(app.LCtabs(LC).lbx.Items, app.LCtabs(LC).lbx.Value));
            end

            app.LCtabs(LC).lbx.Items(index) = [];
            idx = find(strcmp(string(lcforces(index).name), cat(1,app.Forces.name)));
            app.Forces(idx) = [];

            TabGroupLCSelectionChanged(app, event)
        end

        % Button pushed function: AddVoidButton
        function AddVoidButtonPushed(app, event)
            if ~isempty(app.Snodes)
                app.TextArea.Value = '';
                delete(app.extracted)
                void=struct();
                void.nodes=app.Snodes(:,1);

                cn = 1;
                while ismember(strcat('Void_',num2str(cn)), app.VoidsListBox.Items)==1
                    cn=cn+1;
                end

                void.name=strcat('Void_',num2str(cn));
                app.VoidsListBox.Items{end+1} = void.name;
                void.elem=[];

                for x=1:app.nelx
                    for z=1:app.nelz
                        for y=1:app.nely
                            if ismember(app.nodeNrs(y,z,x),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z,x),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z+1,x),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z+1,x),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z,x+1),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z,x+1),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z+1,x+1),void.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z+1,x+1),void.nodes) == 1
                                void.elem = [void.elem app.elemNrs(y,z,x)];
                            end
                        end
                    end
                end

                if isempty(app.Voids)
                    app.Voids=void;
                else
                    app.Voids=[app.Voids; void];
                end

                allvoids = unique(cat(1,app.Voids.nodes));
                app.actnod = setdiff(app.actnod,allvoids);

                delete(app.grid)
                app.grid = plot3(app.UIAxesNodes,app.nodes(app.actnod,1),app.nodes(app.actnod
                    ,2),app.nodes(app.actnod,3),'bx');

                app.Snodes=[];

            else
                app.TextArea.Value = 'First extract nodes...';
```

```
            end
        end

        % Value changed function: VoidsListBox
        function VoidsListBoxValueChanged(app, event)
            index = find(ismember(app.VoidsListBox.Items, app.VoidsListBox.Value));

            delete(app.selected)

            if ~isempty(app.ConstraintsListBox.Items)
                app.ConstraintsListBox.Value = {};
            end
            if ~isempty(app.ForcesListBox.Items)
                app.ForcesListBox.Value = {};
            end
            if ~isempty(app.SolidsListBox.Items)
                app.SolidsListBox.Value = {};
            end

            app.selected = plot3(app.UIAxesNodes,app.nodes(app.Voids(index).nodes,1),app.
                nodes(app.Voids(index).nodes,2),app.nodes(app.Voids(index).nodes,3),'x');
            app.selected.Color = [.9 .9 .9];
        end

        % Button pushed function: RemoveVoidButton
        function RemoveVoidButtonPushed(app, event)
            [~,idx] = ismember(app.VoidsListBox.Value,app.VoidsListBox.Items);
            app.VoidsListBox.Items(idx) = [];
            app.actnod = [app.actnod; app.Voids(idx).nodes];
            app.Voids(idx)=[];

            delete(app.grid)
            delete(app.selected)

            app.grid = plot3(app.UIAxesNodes,app.nodes(app.actnod,1),app.nodes(app.actnod,2),
                app.nodes(app.actnod,3),'bx');
        end

        % Button pushed function: AddSolidButton
        function AddSolidButtonPushed(app, event)
            if ~isempty(app.Snodes)
                app.TextArea.Value = '';
                delete(app.extracted)
                delete(app.sol)
                solid=struct();
                solid.nodes=app.Snodes(:,1);

                cn = 1;

                while ismember(strcat('Solid_',num2str(cn)), app.SolidsListBox.Items)==1
                    cn=cn+1;
                end

                solid.name=strcat('Solid_',num2str(cn));
                app.SolidsListBox.Items{end+1} = solid.name;
                solid.elem=[];

                for x=1:app.nelx
                    for z=1:app.nelz
                        for y=1:app.nely
                            if ismember(app.nodeNrs(y,z,x),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z,x),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z+1,x),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z+1,x),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z,x+1),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z,x+1),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y,z+1,x+1),solid.nodes) == 1 ...
                            && ismember(app.nodeNrs(y+1,z+1,x+1),solid.nodes) == 1
                                solid.elem = [solid.elem app.elemNrs(y,z,x)];
                            end
                        end
```

```
                    end
                end

                if isempty(app.Solids)
                    app.Solids=solid;
                else
                    app.Solids=[app.Solids; solid];
                end

                allsolids = cat(1,app.Solids.nodes);
                app.sol = plot3(app.UIAxesNodes,app.nodes(allsolids,1),app.nodes(allsolids,2)
                    ,app.nodes(allsolids,3),'k*');

                app.Snodes=[];
            else
                app.TextArea.Value = 'First extract nodes...';
            end
        end

        % Value changed function: SolidsListBox
        function SolidsListBoxValueChanged(app, event)
            index = find(ismember(app.SolidsListBox.Items, app.SolidsListBox.Value));
            delete(app.selected)

            if ~isempty(app.ConstraintsListBox.Items)
                app.ConstraintsListBox.Value = {};
            end
            if ~isempty(app.ForcesListBox.Items)
                app.ForcesListBox.Value = {};
            end
            if ~isempty(app.VoidsListBox.Items)
                app.VoidsListBox.Value = {};
            end

            app.selected = plot3(app.UIAxesNodes,app.nodes(app.Solids(index).nodes,1),app.
                nodes(app.Solids(index).nodes,2),app.nodes(app.Solids(index).nodes,3),'m*');
        end

        % Button pushed function: RemoveSolidButton
        function RemoveSolidButtonPushed(app, event)
            [~,idx] = ismember(app.SolidsListBox.Value,app.SolidsListBox.Items);
            app.SolidsListBox.Items(idx) = [];
            app.Solids(idx)=[];

            allsolids = cat(1,app.Solids.nodes);

            delete(app.sol)
            delete(app.selected)

            app.sol = plot3(app.UIAxesNodes,app.nodes(allsolids,1),app.nodes(allsolids,2),app
                .nodes(allsolids,3),'k*');
        end

        % Value changed function: VolFracSpinner
        function VolFracSpinnerValueChanged(app, event)
            value = app.VolFracSpinner.Value;
        end

        % Value changed function: MaxIterationsperrunSpinner
        function MaxIterationsperrunSpinnerValueChanged(app, event)
            app.maxit = app.MaxIterationsperrunSpinner.Value;
        end

        % Value changed function: FilterRadiusSpinner
        function FilterRadiusSpinnerValueChanged(app, event)
            value = app.FilterRadiusSpinner.Value;
        end

        % Value changed function: YoungsModulusSpinner
        function YoungsModulusSpinnerValueChanged(app, event)
            value = app.YoungsModulusSpinner.Value;
```

```
        end

        % Value changed function: PoissonratioSpinner
        function PoissonratioSpinnerValueChanged(app, event)
            value = app.PoissonratioSpinner.Value;
        end

        % Menu selected function: ResetAllMenu
        function ResetAllMenuSelected(app, event)
            ResetAll(app)
        end

        % Button pushed function: AddMaterialButton
        function AddMaterialButtonPushed(app, event)
            material=struct();
            material.name = app.MaterialNameEditField.Value;
            material.E = app.YoungsModulusSpinner.Value;
            material.rho = app.DensitySpinner.Value;
            material.nu = app.PoissonratioSpinner.Value;
            app.Materials = [app.Materials; material];
            app.UITableMaterials.Data = struct2table(app.Materials);
        end

        % Button pushed function: RemoveMaterialButton
        function RemoveMaterialButtonPushed(app, event)
             idx = app.UITableMaterials.Selection(1);
             app.Materials(idx) = [];
             app.UITableMaterials.Data = struct2table(app.Materials);
        end

        % Value changed function: PresetsDropDown
        function PresetsDropDownValueChanged(app, event)
            value = app.PresetsDropDown.Value;
            if value == "Structural steel S235JR"
                app.MaterialNameEditField.Value = "Structural steel S235JR";
                app.YoungsModulusSpinner.Value = 206000;
                app.DensitySpinner.Value = 7850;
                app.PoissonratioSpinner.Value = 0.3;
            elseif value == "Aluminum AlSi12"
                app.MaterialNameEditField.Value = "Aluminum AlSi12";
                app.YoungsModulusSpinner.Value = 72000;
                app.DensitySpinner.Value = 2650;
                app.PoissonratioSpinner.Value = 0.27;
            elseif value == "Titanium Alloy"
                app.MaterialNameEditField.Value = "Titanium Alloy";
                app.YoungsModulusSpinner.Value = 110000;
                app.DensitySpinner.Value = 4430;
                app.PoissonratioSpinner.Value = 0.34;
            end
        end

        % Value changed function: TextArea
        function TextAreaValueChanged(app, event)
            value = app.TextArea.Value;
        end

        % Value changed function: XEditField
        function XEditFieldValueChanged(app, event)
            value = app.XEditField.Value;
        end

        % Value changed function: XdirectionCheckBox
        function XdirectionCheckBoxValueChanged(app, event)
            value = app.XdirectionCheckBox.Value;
        end

        % Value changed function: YdirectionCheckBox
        function YdirectionCheckBoxValueChanged(app, event)
            value = app.YdirectionCheckBox.Value;
        end

```

```
        % Value changed function: ZdirectionCheckBox
        function ZdirectionCheckBoxValueChanged(app, event)
            value = app.ZdirectionCheckBox.Value;
        end

        % Value changed function: DensitySpinner
        function DensitySpinnerValueChanged(app, event)
            value = app.DensitySpinner.Value;
        end

        % Selection changed function: UITableMaterials
        function UITableMaterialsSelectionChanged(app, event)
            selection = app.UITableMaterials.Selection;
        end

        % Button pushed function: ShowDesignDataButton
        function ShowDesignDataButtonPushed(app, event)
            [~] = GetResultFromSelectedDataPoint(app);
        end

        % Selection changed function: UITableData
        function UITableDataSelectionChanged(app, event)
            idx = app.UITableData.Selection;

            cla(app.UIAxesTableSelection)
            cla(app.UIAxesTableSelection_2)
            tur = turbo;
            tur = flipud(tur);

            result.faces = app.Result(idx).faces;
            result.vertices = app.Result(idx).vertices;
            dcc = app.Result(idx).dcc;
            isovals = app.Result(idx).isovals;
            vonmises = app.Result(idx).vonmises;

            xlabel(app.UIAxesTableSelection, 'Y')
            ylabel(app.UIAxesTableSelection, 'X')
            zlabel(app.UIAxesTableSelection, 'Z')
            set(app.UIAxesTableSelection,'XTickLabel',[])
            set(app.UIAxesTableSelection,'YTickLabel',[])
            set(app.UIAxesTableSelection,'ZTickLabel',[])

            set(app.UIAxesTableSelection,'xdir','reverse');
            title(app.UIAxesTableSelection,'Compliance Sensitivity (J)')

            rp = patch(app.UIAxesTableSelection, result);
            isonormals(isovals,rp)
            isocolors(dcc,rp)
            rp.FaceColor = 'interp';
            colormap(app.UIAxesTableSelection,tur)
            colorbar(app.UIAxesTableSelection)
            view(app.UIAxesTableSelection, [ 145, 25 ]);
            axis(app.UIAxesTableSelection, "equal");

            xlabel(app.UIAxesTableSelection_2, 'Y')
            ylabel(app.UIAxesTableSelection_2, 'X')
            zlabel(app.UIAxesTableSelection_2, 'Z')
            set(app.UIAxesTableSelection_2,'XTickLabel',[])
            set(app.UIAxesTableSelection_2,'YTickLabel',[])
            set(app.UIAxesTableSelection_2,'ZTickLabel',[])
            set(app.UIAxesTableSelection_2,'xdir','reverse');
            title(app.UIAxesTableSelection_2,'Von Mises Stress (MPa)')

            rps = patch(app.UIAxesTableSelection_2, result);
            isonormals(isovals,rps)
            isocolors(vonmises,rps)
            rps.FaceColor = 'interp';
            colormap(app.UIAxesTableSelection_2,'turbo')
            colorbar(app.UIAxesTableSelection_2)
            view(app.UIAxesTableSelection_2, [ 145, 25 ]);
            axis(app.UIAxesTableSelection_2, "equal");
```

```
        end

        % Menu selected function: ImportGeometryMenu
        function ImportGeometryMenuSelected(app, event)
           ResetAll(app)
           [filename, ~]=uigetfile('*.stl');
           figure(app.UIFigure)
           if ~isequal(filename, 0)  % User did not press Cancel:
               app.stlfile = filename;
               ImportMyGeometry(app,app.stlfile)
           end
        end

        % Menu selected function: SaveSetupMenu
        function SaveSetupMenuSelected(app, event)
           [file, folder] = uiputfile('*.mat');
           if ~isequal(file, 0)  % User did not press Cancel
                master = struct();
                master.geom = app.stlfile;
                master.clb = app.ConstraintsListBox.Items;
                master.vlb = app.VoidsListBox.Items;
                master.slb = app.SolidsListBox.Items;

                master.maxit = app.MaxIterationsperrunSpinner.Value;
                master.volfracmin = app.VolFracSpinner.Value;
                master.volfracmax = app.VolFracSpinner_2.Value;
                master.volfracstep = app.StepVolFracSpinner.Value;
                master.filrmin = app.FilterRadiusSpinner.Value;
                master.filrmax = app.FilterRadiusSpinner_2.Value;
                master.filrstep = app.StepFilterRadiusSpinner.Value;

                master.materials = app.Materials;
                master.lc = app.LoadCasesSpinner.Value;

                master.nelx = app.nelx;
                master.nely = app.nely;
                master.nelz = app.nelz;
                master.sz = app.sz;

                master.constr = app.Constr;
                master.forces = app.Forces;
                master.voids = app.Voids;
                master.solids = app.Solids;
                master.nodes = app.nodes;
                master.actnod = app.actnod;

                master.LCtabs = app.LCtabs;

                save(fullfile(folder, file), 'master')
                app.TextArea.Value = 'Setup file saved';
           end
        end

        % Menu selected function: ImportSetupMenu
        function ImportSetupMenuSelected(app, event)
            ResetAll(app)
            [filename, ~]=uigetfile('*.mat');
            figure(app.UIFigure)
            if ~isequal(filename, 0)
                warning('off','MATLAB:appdesigner:appdesigner:LoadObjWarning')
                load(filename,'master');
                figure(app.UIFigure)
                app.stlfile = master.geom;
                app.MeshrefinementSlider.Value = master.nelx;
                app.ConstraintsListBox.Items = master.clb;
                app.VoidsListBox.Items = master.vlb;
                app.SolidsListBox.Items = master.slb;

                app.MaxIterationsperrunSpinner.Value = master.maxit;
                app.maxit = master.maxit;
                app.VolFracSpinner.Value = master.volfracmin;
```

```
                app.VolFracSpinner_2.Value = master.volfracmax;
                app.StepVolFracSpinner.Value = master.volfracstep;
                app.FilterRadiusSpinner.Value = master.filrmin;
                app.FilterRadiusSpinner_2.Value = master.filrmax;
                app.StepFilterRadiusSpinner.Value = master.filrstep;

                app.Materials = master.materials;
                app.UITableMaterials.Data = struct2table(app.Materials);

                app.LoadCasesSpinner.Value = master.lc;

                LoadCasesSpinnerValueChanged(app,event)

                for i=1:app.LoadCasesSpinner.Value
                    app.LCtabs(i).lbx.Items = master.LCtabs(i).lbx.Items;
                end

                app.nelx = master.nelx;
                app.nely = master.nely;
                app.nelz = master.nelz;
                app.sz = master.sz;

                app.Constr = master.constr;
                app.Forces = master.forces;
                app.Voids = master.voids;
                app.Solids = master.solids;
                app.nodes = master.nodes;
                app.actnod = master.actnod;

                ImportMyGeometry(app,app.stlfile)

                app.nel = app.nelx*app.nely*app.nelz;
                app.nodeNrs = reshape(1:(1+app.nelx)*(1+app.nely)*(1+app.nelz),1+app.nely,1+
                    app.nelz,1+app.nelx);
                app.elemNrs = reshape(1:(app.nelx)*(app.nely)*(app.nelz),app.nely,app.nelz,
                    app.nelx);

                if isempty(app.actnod)
                    app.actnod = (1:length(app.nodes))';
                end
                app.grid = plot3(app.UIAxesNodes,app.nodes(app.actnod,1),app.nodes(app.actnod
                    ,2),app.nodes(app.actnod,3),'bx');

                allconstr = cat(1,app.Constr.nodes);
                app.con = plot3(app.UIAxesNodes,app.nodes(allconstr,1),app.nodes(allconstr,2)
                    ,app.nodes(allconstr,3),'r*');

                if ~isempty(app.Solids)
                    allsolids = cat(1,app.Solids.nodes);
                    app.sol = plot3(app.UIAxesNodes,app.nodes(allsolids,1),app.nodes(
                        allsolids,2),app.nodes(allsolids,3),'k*');
                end
                TabGroupLCSelectionChanged(app, event)
            end
        end

        % Button pushed function: ExportSelectedSolutionasSTLButton
        function ExportSelectedSolutionasSTLButtonPushed(app, event)
            [result] = GetResultFromSelectedDataPoint(app);
            [file, folder] = uiputfile('*.stl');
            if ~isequal(file, 0)  % User did not press Cancel:
                stlwrite(fullfile(folder, file), result)
                app.TextArea.Value = strcat('.STL file saved: ',file);
            end
        end

        % Button pushed function: ExportSelectedSolutionasSTLButton_2
        function ExportSelectedSolutionasSTLButton_2Pushed(app, event)
            idx = app.UITableData.Selection;
            result.faces = app.Result(idx).faces;
            result.vertices = app.Result(idx).vertices;
```

```
            [file, folder] = uiputfile('*.stl');
            if ~isequal(file, 0)  % User did not press Cancel:
                stlwrite(fullfile(folder, file), result)
                app.TextArea.Value = strcat('.STL file saved: ',file);
            end
        end

        % Size changed function: DensityPlotsTab
        function DensityPlotsTabSizeChanged(app, event)
            position = app.DensityPlotsTab.Position;
            app.DTab.w=position(3);
            app.DTab.h=position(4);
        end

        % Size changed function: LC_1Tab
        function LC_1TabSizeChanged(app, event)
            position = app.LC_1Tab.Position;
            app.PosFTab.w=position(3);
            app.PosFTab.h=position(4);
            for i=1:length(app.LCtabs)
                app.LCtabs(i).lbx.Position = [66,8,(app.PosFTab.w-81),(app.PosFTab.h-14)];
            end
        end

        % Selection change function: TabGroupLC
        function TabGroupLCSelectionChanged(app, event)
            delete(app.extracted)
            delete(app.fcs)
            delete(app.arrow)
            delete(app.selected)

            LC = find(strcmp(string(app.TabGroupLC.SelectedTab.Title),cat(1,app.LCtabs.Title)
                ));
            if ~isempty(app.Forces)
                allforces = cat(1,app.Forces.lc);
                index = find(allforces==LC);
                if ~isempty(index)
                    allforces = cat(1,app.Forces(index).nodes);
                    allforcesU = cat(1,app.Forces(index).U)/10;
                    allforcesV = cat(1,app.Forces(index).V)/10;
                    allforcesW = cat(1,app.Forces(index).W)/10;

                    app.fcs = plot3(app.UIAxesNodes,app.nodes(allforces,1),app.nodes(
                        allforces,2),app.nodes(allforces,3),'g*');

                    app.arrow = quiver3(app.UIAxesNodes,app.nodes(allforces,1),app.nodes(
                        allforces,2),app.nodes(allforces,3),allforcesU,allforcesV,allforcesW,
                        'g','LineWidth',2);
                end
            end
        end

        % Button pushed function: RunButton
        function RunButtonPushed(app, event)
            if ~isempty(app.Constr)
                if ~isempty(app.Forces)
                    if ~isempty(app.Materials)
                        %% Cleaning
                        delete(app.DensityPlotsTab.Children)
                        cla(app.UIAxes)
                        cla(app.UIAxesSelection)
                        cla(app.UIAxesSelection_2)
                        cla(app.UIAxesTableSelection)
                        cla(app.UIAxesTableSelection_2)
                        app.UITableSelection.Data = [];
                        app.UITableData.Data = [];
                        app.axess = [];
                        app.axesL = [];
                        app.Data = [];
                        app.Result = [];
                        app.DataTable = [];
```

```

if ~isempty(app.LocTab)
    delete(app.LocTab)
end

%% General Parameters
penal=3;
ft=1;
ftBC='N';
eta=0.5;
beta=1;
move=0.2;
tur = turbo;
tur = flipud(tur);

%% Set Passive Voids & Solids
if ~isempty(app.Voids)
    pasV = unique(cat(2,app.Voids.elem));
else
    pasV = [];
end

if ~isempty(app.Solids)
    pasS = unique(cat(2,app.Solids.elem));
else
    pasS = [];
end

%% Assemble fixed DoF for Constraints
for i=1:length(app.Constr)
    if app.Constr(i).xyz(1) == 1
        fixedx=(3*app.Constr(i).nodes)-2;
    end

    if app.Constr(i).xyz(2) == 1
        fixedy=(3*app.Constr(i).nodes)-1;
    end

    if app.Constr(i).xyz(3) == 1
        fixedz=3*app.Constr(i).nodes;
    end
    app.Constr(i).fixed = sort([fixedx;fixedy;fixedz]);
end
fixed = cat(1,app.Constr.fixed);

%% Assemble lcDof and F for Forces
lcDof = {};
F = {};

for j = 1:app.LoadCasesSpinner.Value
    index = find(cat(1,app.Forces.lc)==j);
    allforces = app.Forces(index);
    lcDof(:,j) = {cat(1,allforces.lcDof)};
    F(:,j) = {cat(1,allforces.Fl)};
end

%% Start Run
app.TextArea.Value="Trying out different settings...";
app.TabGroup.SelectedTab = app.DensityPlotsTab;

i = 1;
x = [1 ((app.DTab.w-20)/3)+5 ((app.DTab.w-20)*2/3)+10];
y = 1;
xi = 1;

for volfr = app.VolFracSpinner.Value:app.StepVolFracSpinner.Value:app.VolFracSpinner_2.Value
    for frmin = app.FilterRadiusSpinner.Value:app.StepFilterRadiusSpinner.Value:app.FilterRadiusSpinner_2.Value
        for k = 1:length(app.Materials)
```

```
                        [result, perf, dcc, isovals, MaxVonMises, vonmises] =
                            TOPGD_TO(app, app.nelx,app.nely,app.nelz,volfr,penal,
                            frmin,ft,ftBC,eta,beta,move,...
                            app.Materials(k).E,app.Materials(k).nu,app.maxit,pasV
                                ,pasS,fixed,lcDof,F);
                        app.Data(i,:,k)=[perf.V perf.C frmin perf.V*app.Volume*(
                            app.Materials(k).rho*10^-6) MaxVonMises];
                        app.Result(i).faces=result.faces;
                        app.Result(i).vertices=result.vertices;
                        app.Result(i).dcc = dcc;
                        app.Result(i).isovals = isovals;
                        app.Result(i).vonmises = vonmises;

                        run.comp = perf.C;
                        run.mass = perf.V*app.Volume*(app.Materials(k).rho*10^-6)
                            ;
                        run.stress = MaxVonMises;
                        run.VF = volfr;
                        run.fr = frmin;
                        run.mat = app.Materials(k).name;
                        app.DataTable = [app.DataTable; run];

                        ax = uiaxes(app.DensityPlotsTab,'Position',[x(xi),y,((app
                            .DTab.w-20)/3),((app.DTab.w-20)/4)]);
                        xlabel(ax, 'Y')
                        ylabel(ax, 'X')
                        zlabel(ax, 'Z')
                        set(ax,'XTickLabel',[])
                        set(ax,'YTickLabel',[])
                        set(ax,'ZTickLabel',[])
                        app.axess = [app.axess; ax];
                        rp = patch(ax, result);
                        set(ax,'xdir','reverse');
                        isonormals(isovals,rp)
                        isocolors(dcc,rp)
                        rp.FaceColor = 'interp';
                        colormap(ax,tur)
                        view(ax, [ 145, 25 ]);
                        axis(ax, "equal");
                        titletext{1} = [strcat("Comp.: ",num2str(perf.C)," J","
                            Mass: ",num2str(perf.V*app.Volume*(app.Materials(k).
                            rho*10^-6))," g")];
                        titletext{2} = [strcat("Material: ",app.Materials(k).name
                            )];
                        titletext{3} = [strcat("Vol. frac.: ",num2str(volfr), "
                            Filter rad.: ",num2str(frmin))];
                        title(ax,titletext);

                        i=i+1;
                        if xi<3
                            xi = xi+1;
                        elseif xi == 3
                            xi = 1;
                            y = y + ((app.DTab.w-20)/4)+5;
                        end
                    end
                end
            end

            app.TabGroup.SelectedTab = app.ComplianceMassGraphTab;
            hold(app.UIAxes,'on');
            app.maxmass = max(nonzeros(app.Data(:,4,:)),[],'all');
            app.maxcomp = max(nonzeros(app.Data(:,2,:)),[],'all');
            for k = 1:length(app.Materials)
                app.resplot(k) = plot(app.UIAxes,(nonzeros(app.Data(:,4,k))/app.
                    maxmass),(nonzeros(app.Data(:,2,k))/app.maxcomp),'+');
            end
            xlabel(app.UIAxes,'Mass (Normalized)');
            ylabel(app.UIAxes, 'Compliance (Normalized)');
            matrls = struct2cell(app.Materials);
            legend(app.UIAxes, matrls(1,:));
```

```

                app.T = struct2table(app.DataTable);
                app.UITableData.Data = app.T;
                close all
                app.TextArea.Value="Results Plotted";
            else
                app.TextArea.Value = 'No Material defined';
            end
        else
            app.TextArea.Value = 'No Force defined';
        end
    else
        app.TextArea.Value = 'No Constraint defined';
    end
end

% Button pushed function: ImportExcelButton
function ImportExcelButtonPushed(app, event)
     [filename, ~]=uigetfile('*.xlsx');
     figure(app.UIFigure)
     if ~isequal(filename, 0)
         app.FT = readtable(filename);
         app.LoadCasesSpinner.Value = max(app.FT.LoadCase);
         LoadCasesSpinnerValueChanged(app,event)
         for i = 1:size(app.FT,1)
            force.lc = app.FT.LoadCase(i);

            force.name=app.FT.Name{i};
            app.LCtabs(force.lc).lbx.Items{end+1} = force.name;

            Fcoor = [app.FT.LocX(i) app.FT.LocY(i) app.FT.LocZ(i)];

            k = dsearchn(app.nodes(app.actnod,:),Fcoor);

            ndsx = app.actnod;
            ndsy = app.actnod;
            ndsz = app.actnod;

            if app.FT.VariationX(i) == 0
               ndsx = find(app.nodes(app.actnod,1)==app.nodes(app.actnod(k),1));
            end
            if app.FT.VariationY(i) == 0
                ndsy = find(app.nodes(app.actnod,2)==app.nodes(app.actnod(k),2));
            end
            if app.FT.VariationZ(i) == 0
                ndsz = find(app.nodes(app.actnod,3)==app.nodes(app.actnod(k),3));
            end
            nds = intersect(intersect(ndsx,ndsy),ndsz);

            if app.FT.VariationX(i) == 1 && app.FT.VariationY(i) == 1 && app.FT.
                VariationZ(i) == 1
                ndss = app.nodes(app.actnod,:);
            else
                ndss=app.nodes(app.actnod(nds),:);
            end

            ptCloud = pointCloud(ndss);
            r = app.FT.Radius(i);

            [idx, ~] = findNeighborsInRadius(ptCloud,Fcoor,r);
            [~,force.nodes]=ismember(ndss(idx,:),app.nodes,'rows');

            lcDofx=[];
            lcDofy=[];
            lcDofz=[];
            if ~app.FT.MagX(i) == 0
                lcDofx=(3*force.nodes)-2;
                force.U = app.FT.MagX(i)*ones(length(force.nodes),1);
            else
                force.U = zeros(length(force.nodes),1);
            end
```

```
                if ~app.FT.MagY(i) == 0
                    lcDofy=(3*force.nodes)-1;
                    force.V = app.FT.MagY(i)*ones(length(force.nodes),1);
                else
                    force.V = zeros(length(force.nodes),1);
                end

                if ~app.FT.MagZ(i) == 0
                    lcDofz=3*force.nodes;
                    force.W = app.FT.MagZ(i)*ones(length(force.nodes),1);
                else
                    force.W = zeros(length(force.nodes),1);
                end

                force.lcDof = [lcDofx;lcDofy;lcDofz];
                force.Fl = nonzeros([force.U/length(force.U); force.V/length(force.V);
                    force.W/length(force.W)]);

                if isempty(app.Forces)
                    app.Forces=force;
                else
                    app.Forces=[app.Forces; force];
                end
             end
             TabGroupLCSelectionChanged(app, event)
         end
     end
 end

 % Component initialization
 methods (Access = private)

     % Create UIFigure and components
     function createComponents(app)

         % Create UIFigure and hide until all components are created
         app.UIFigure = uifigure('Visible', 'off');
         app.UIFigure.Position = [100 100 1221 707];
         app.UIFigure.Name = 'MATLAB App';

         % Create GeneralMenu
         app.GeneralMenu = uimenu(app.UIFigure);
         app.GeneralMenu.Text = 'General';

         % Create SaveSetupMenu
         app.SaveSetupMenu = uimenu(app.GeneralMenu);
         app.SaveSetupMenu.MenuSelectedFcn = createCallbackFcn(app, @SaveSetupMenuSelected
             , true);
         app.SaveSetupMenu.Text = 'Save Setup';

         % Create ResetAllMenu
         app.ResetAllMenu = uimenu(app.GeneralMenu);
         app.ResetAllMenu.MenuSelectedFcn = createCallbackFcn(app, @ResetAllMenuSelected,
             true);
         app.ResetAllMenu.Text = 'Reset All';

         % Create ImportMenu
         app.ImportMenu = uimenu(app.UIFigure);
         app.ImportMenu.Text = 'Import';

         % Create ImportGeometryMenu
         app.ImportGeometryMenu = uimenu(app.ImportMenu);
         app.ImportGeometryMenu.MenuSelectedFcn = createCallbackFcn(app,
             @ImportGeometryMenuSelected, true);
         app.ImportGeometryMenu.Text = 'Import Geometry';

         % Create ImportSetupMenu
         app.ImportSetupMenu = uimenu(app.ImportMenu);
         app.ImportSetupMenu.MenuSelectedFcn = createCallbackFcn(app,
             @ImportSetupMenuSelected, true);
```

```
            app.ImportSetupMenu.Text = 'Import Setup';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [0 68 1220 639];

            % Create SetupTab
            app.SetupTab = uitab(app.TabGroup);
            app.SetupTab.Title = 'Setup';

            % Create UIAxesNodes
            app.UIAxesNodes = uiaxes(app.SetupTab);
            title(app.UIAxesNodes, 'Setup Node View')
            xlabel(app.UIAxesNodes, 'X')
            ylabel(app.UIAxesNodes, 'Y')
            zlabel(app.UIAxesNodes, 'Z')
            app.UIAxesNodes.Position = [638 222 558 381];

            % Create NodesPanel
            app.NodesPanel = uipanel(app.SetupTab);
            app.NodesPanel.Title = 'Nodes';
            app.NodesPanel.Position = [295 401 327 148];

            % Create ShowNodesButton
            app.ShowNodesButton = uibutton(app.NodesPanel, 'push');
            app.ShowNodesButton.ButtonPushedFcn = createCallbackFcn(app, @ShowNodesButtonPushed, true);
            app.ShowNodesButton.Position = [17 21 83 23];
            app.ShowNodesButton.Text = 'Show Nodes';

            % Create ExtractNodesButton
            app.ExtractNodesButton = uibutton(app.NodesPanel, 'push');
            app.ExtractNodesButton.ButtonPushedFcn = createCallbackFcn(app, @ExtractNodesButtonPushed, true);
            app.ExtractNodesButton.FontWeight = 'bold';
            app.ExtractNodesButton.Position = [220 21 96 23];
            app.ExtractNodesButton.Text = 'Extract Nodes';

            % Create VoxelizeButton
            app.VoxelizeButton = uibutton(app.NodesPanel, 'push');
            app.VoxelizeButton.ButtonPushedFcn = createCallbackFcn(app, @VoxelizeButtonPushed, true);
            app.VoxelizeButton.Position = [117 21 83 23];
            app.VoxelizeButton.Text = 'Voxelize ';

            % Create MeshrefinementSliderLabel
            app.MeshrefinementSliderLabel = uilabel(app.NodesPanel);
            app.MeshrefinementSliderLabel.HorizontalAlignment = 'right';
            app.MeshrefinementSliderLabel.Position = [12 86 94 22];
            app.MeshrefinementSliderLabel.Text = 'Mesh refinement';

            % Create MeshrefinementSlider
            app.MeshrefinementSlider = uislider(app.NodesPanel);
            app.MeshrefinementSlider.Limits = [5 50];
            app.MeshrefinementSlider.MajorTicks = [5 10 15 20 25 30 35 40 45 50];
            app.MeshrefinementSlider.ValueChangedFcn = createCallbackFcn(app, @MeshrefinementSliderValueChanged, true);
            app.MeshrefinementSlider.Position = [127 95 180 3];
            app.MeshrefinementSlider.Value = 10;

            % Create InputParametersPanel
            app.InputParametersPanel = uipanel(app.SetupTab);
            app.InputParametersPanel.Title = 'Input Parameters';
            app.InputParametersPanel.Position = [14 401 263 197];

            % Create VolumeFractionLabel
            app.VolumeFractionLabel = uilabel(app.InputParametersPanel);
            app.VolumeFractionLabel.HorizontalAlignment = 'right';
            app.VolumeFractionLabel.Position = [8 124 115 22];
            app.VolumeFractionLabel.Text = 'Min Volume Fraction';

```

```
            % Create VolFracSpinner
            app.VolFracSpinner = uispinner(app.InputParametersPanel);
            app.VolFracSpinner.Step = 0.1;
            app.VolFracSpinner.Limits = [0.01 1];
            app.VolFracSpinner.ValueChangedFcn = createCallbackFcn(app,
                @VolFracSpinnerValueChanged, true);
            app.VolFracSpinner.Position = [148 124 97 22];
            app.VolFracSpinner.Value = 0.15;

            % Create MaxIterationsperrunSpinnerLabel
            app.MaxIterationsperrunSpinnerLabel = uilabel(app.InputParametersPanel);
            app.MaxIterationsperrunSpinnerLabel.HorizontalAlignment = 'right';
            app.MaxIterationsperrunSpinnerLabel.Position = [8 145 125 22];
            app.MaxIterationsperrunSpinnerLabel.Text = 'Max. Iterations per run';

            % Create MaxIterationsperrunSpinner
            app.MaxIterationsperrunSpinner = uispinner(app.InputParametersPanel);
            app.MaxIterationsperrunSpinner.Limits = [1 Inf];
            app.MaxIterationsperrunSpinner.ValueDisplayFormat = '%.0f';
            app.MaxIterationsperrunSpinner.ValueChangedFcn = createCallbackFcn(app,
                @MaxIterationsperrunSpinnerValueChanged, true);
            app.MaxIterationsperrunSpinner.Position = [148 145 97 22];
            app.MaxIterationsperrunSpinner.Value = 50;

            % Create FilterradiusLabel
            app.FilterradiusLabel = uilabel(app.InputParametersPanel);
            app.FilterradiusLabel.HorizontalAlignment = 'right';
            app.FilterradiusLabel.Position = [9 63 90 22];
            app.FilterradiusLabel.Text = 'Min Filter radius';

            % Create FilterRadiusSpinner
            app.FilterRadiusSpinner = uispinner(app.InputParametersPanel);
            app.FilterRadiusSpinner.Step = 0.1;
            app.FilterRadiusSpinner.Limits = [0.1 Inf];
            app.FilterRadiusSpinner.ValueChangedFcn = createCallbackFcn(app,
                @FilterRadiusSpinnerValueChanged, true);
            app.FilterRadiusSpinner.Position = [148 63 97 22];
            app.FilterRadiusSpinner.Value = 1.7;

            % Create VolumeFractionLabel_2
            app.VolumeFractionLabel_2 = uilabel(app.InputParametersPanel);
            app.VolumeFractionLabel_2.HorizontalAlignment = 'right';
            app.VolumeFractionLabel_2.Position = [8 104 118 22];
            app.VolumeFractionLabel_2.Text = 'Max Volume Fraction';

            % Create VolFracSpinner_2
            app.VolFracSpinner_2 = uispinner(app.InputParametersPanel);
            app.VolFracSpinner_2.Step = 0.1;
            app.VolFracSpinner_2.Limits = [0.01 1];
            app.VolFracSpinner_2.Position = [148 104 97 22];
            app.VolFracSpinner_2.Value = 0.35;

            % Create FilterradiusLabel_2
            app.FilterradiusLabel_2 = uilabel(app.InputParametersPanel);
            app.FilterradiusLabel_2.HorizontalAlignment = 'right';
            app.FilterradiusLabel_2.Position = [9 42 94 22];
            app.FilterradiusLabel_2.Text = 'Max Filter radius';

            % Create FilterRadiusSpinner_2
            app.FilterRadiusSpinner_2 = uispinner(app.InputParametersPanel);
            app.FilterRadiusSpinner_2.Step = 0.1;
            app.FilterRadiusSpinner_2.Limits = [0.1 Inf];
            app.FilterRadiusSpinner_2.Position = [148 42 97 22];
            app.FilterRadiusSpinner_2.Value = 2;

            % Create VolumeFractionLabel_3
            app.VolumeFractionLabel_3 = uilabel(app.InputParametersPanel);
            app.VolumeFractionLabel_3.HorizontalAlignment = 'right';
            app.VolumeFractionLabel_3.Position = [7 83 120 22];
            app.VolumeFractionLabel_3.Text = 'Step Volume Fraction';

```

```
            % Create StepVolFracSpinner
            app.StepVolFracSpinner = uispinner(app.InputParametersPanel);
            app.StepVolFracSpinner.Step = 0.1;
            app.StepVolFracSpinner.Limits = [0.01 1];
            app.StepVolFracSpinner.Position = [148 83 97 22];
            app.StepVolFracSpinner.Value = 0.1;

            % Create FilterradiusLabel_3
            app.FilterradiusLabel_3 = uilabel(app.InputParametersPanel);
            app.FilterradiusLabel_3.HorizontalAlignment = 'right';
            app.FilterradiusLabel_3.Position = [7 21 96 22];
            app.FilterradiusLabel_3.Text = 'Step Filter radius';

            % Create StepFilterRadiusSpinner
            app.StepFilterRadiusSpinner = uispinner(app.InputParametersPanel);
            app.StepFilterRadiusSpinner.Step = 0.1;
            app.StepFilterRadiusSpinner.Limits = [0.1 Inf];
            app.StepFilterRadiusSpinner.Position = [148 21 97 22];
            app.StepFilterRadiusSpinner.Value = 0.3;

            % Create VoidRegionsPanel
            app.VoidRegionsPanel = uipanel(app.SetupTab);
            app.VoidRegionsPanel.Title = 'Void Regions';
            app.VoidRegionsPanel.Position = [412 216 210 175];

            % Create VoidsListBoxLabel
            app.VoidsListBoxLabel = uilabel(app.VoidRegionsPanel);
            app.VoidsListBoxLabel.HorizontalAlignment = 'right';
            app.VoidsListBoxLabel.Position = [15 122 62 22];
            app.VoidsListBoxLabel.Text = 'Voids';

            % Create VoidsListBox
            app.VoidsListBox = uilistbox(app.VoidRegionsPanel);
            app.VoidsListBox.Items = {};
            app.VoidsListBox.ValueChangedFcn = createCallbackFcn(app,
                @VoidsListBoxValueChanged, true);
            app.VoidsListBox.Position = [81 57 116 89];
            app.VoidsListBox.Value = {};

            % Create AddVoidButton
            app.AddVoidButton = uibutton(app.VoidRegionsPanel, 'push');
            app.AddVoidButton.ButtonPushedFcn = createCallbackFcn(app, @AddVoidButtonPushed,
                true);
            app.AddVoidButton.Position = [12 10 86 23];
            app.AddVoidButton.Text = 'Add Void';

            % Create RemoveVoidButton
            app.RemoveVoidButton = uibutton(app.VoidRegionsPanel, 'push');
            app.RemoveVoidButton.ButtonPushedFcn = createCallbackFcn(app,
                @RemoveVoidButtonPushed, true);
            app.RemoveVoidButton.Position = [111 10 87 23];
            app.RemoveVoidButton.Text = 'Remove Void';

            % Create SolidRegionsPanel
            app.SolidRegionsPanel = uipanel(app.SetupTab);
            app.SolidRegionsPanel.Title = 'Solid Regions';
            app.SolidRegionsPanel.Position = [412 6 210 202];

            % Create SolidsListBoxLabel
            app.SolidsListBoxLabel = uilabel(app.SolidRegionsPanel);
            app.SolidsListBoxLabel.HorizontalAlignment = 'right';
            app.SolidsListBoxLabel.Position = [15 149 62 22];
            app.SolidsListBoxLabel.Text = 'Solids';

            % Create SolidsListBox
            app.SolidsListBox = uilistbox(app.SolidRegionsPanel);
            app.SolidsListBox.Items = {};
            app.SolidsListBox.ValueChangedFcn = createCallbackFcn(app,
                @SolidsListBoxValueChanged, true);
            app.SolidsListBox.Position = [81 57 116 116];
            app.SolidsListBox.Value = {};
```

```
            % Create AddSolidButton
            app.AddSolidButton = uibutton(app.SolidRegionsPanel, 'push');
            app.AddSolidButton.ButtonPushedFcn = createCallbackFcn(app, @AddSolidButtonPushed, true);
            app.AddSolidButton.Position = [12 10 86 23];
            app.AddSolidButton.Text = 'Add Solid';

            % Create RemoveSolidButton
            app.RemoveSolidButton = uibutton(app.SolidRegionsPanel, 'push');
            app.RemoveSolidButton.ButtonPushedFcn = createCallbackFcn(app, @RemoveSolidButtonPushed, true);
            app.RemoveSolidButton.Position = [110 10 90 23];
            app.RemoveSolidButton.Text = 'Remove Solid';

            % Create AppliedforceNPanel
            app.AppliedforceNPanel = uipanel(app.SetupTab);
            app.AppliedforceNPanel.Title = 'Applied force (N)';
            app.AppliedforceNPanel.Position = [14 5 381 203];

            % Create AddForceButton
            app.AddForceButton = uibutton(app.AppliedforceNPanel, 'push');
            app.AddForceButton.ButtonPushedFcn = createCallbackFcn(app, @AddForceButtonPushed, true);
            app.AddForceButton.Position = [25 11 100 23];
            app.AddForceButton.Text = 'Add Force';

            % Create XEditFieldLabel
            app.XEditFieldLabel = uilabel(app.AppliedforceNPanel);
            app.XEditFieldLabel.HorizontalAlignment = 'right';
            app.XEditFieldLabel.Position = [5 154 13 22];
            app.XEditFieldLabel.Text = 'X';

            % Create XEditField
            app.XEditField = uieditfield(app.AppliedforceNPanel, 'numeric');
            app.XEditField.ValueChangedFcn = createCallbackFcn(app, @XEditFieldValueChanged, true);
            app.XEditField.Position = [37 154 40 22];

            % Create RemoveForceButton
            app.RemoveForceButton = uibutton(app.AppliedforceNPanel, 'push');
            app.RemoveForceButton.ButtonPushedFcn = createCallbackFcn(app, @RemoveForceButtonPushed, true);
            app.RemoveForceButton.Position = [149 11 100 23];
            app.RemoveForceButton.Text = 'Remove Force';

            % Create YEditFieldLabel
            app.YEditFieldLabel = uilabel(app.AppliedforceNPanel);
            app.YEditFieldLabel.HorizontalAlignment = 'right';
            app.YEditFieldLabel.Position = [5 133 13 22];
            app.YEditFieldLabel.Text = 'Y';

            % Create YEditField
            app.YEditField = uieditfield(app.AppliedforceNPanel, 'numeric');
            app.YEditField.Position = [37 133 40 22];

            % Create ZEditFieldLabel
            app.ZEditFieldLabel = uilabel(app.AppliedforceNPanel);
            app.ZEditFieldLabel.HorizontalAlignment = 'right';
            app.ZEditFieldLabel.Position = [5 112 13 22];
            app.ZEditFieldLabel.Text = 'Z';

            % Create ZEditField
            app.ZEditField = uieditfield(app.AppliedforceNPanel, 'numeric');
            app.ZEditField.Position = [37 112 40 22];

            % Create TabGroupLC
            app.TabGroupLC = uitabgroup(app.AppliedforceNPanel);
            app.TabGroupLC.SelectionChangedFcn = createCallbackFcn(app, @TabGroupLCSelectionChanged, true);
            app.TabGroupLC.Position = [152 38 223 138];
```

```
            % Create LC_1Tab
            app.LC_1Tab = uitab(app.TabGroupLC);
            app.LC_1Tab.AutoResizeChildren = 'off';
            app.LC_1Tab.SizeChangedFcn = createCallbackFcn(app, @LC_1TabSizeChanged, true);
            app.LC_1Tab.Title = 'LC_1';

            % Create ForcesListBoxLabel
            app.ForcesListBoxLabel = uilabel(app.LC_1Tab);
            app.ForcesListBoxLabel.HorizontalAlignment = 'right';
            app.ForcesListBoxLabel.Position = [9 82 42 22];
            app.ForcesListBoxLabel.Text = 'Forces';

            % Create ForcesListBox
            app.ForcesListBox = uilistbox(app.LC_1Tab);
            app.ForcesListBox.Items = {};
            app.ForcesListBox.ValueChangedFcn = createCallbackFcn(app,
                @ForcesListBoxValueChanged, true);
            app.ForcesListBox.Position = [66 8 148 100];
            app.ForcesListBox.Value = {};

            % Create LoadcasesLabel
            app.LoadcasesLabel = uilabel(app.AppliedforceNPanel);
            app.LoadcasesLabel.HorizontalAlignment = 'right';
            app.LoadcasesLabel.Position = [6 56 69 22];
            app.LoadcasesLabel.Text = 'Load Cases';

            % Create LoadCasesSpinner
            app.LoadCasesSpinner = uispinner(app.AppliedforceNPanel);
            app.LoadCasesSpinner.Limits = [1 Inf];
            app.LoadCasesSpinner.RoundFractionalValues = 'on';
            app.LoadCasesSpinner.ValueChangedFcn = createCallbackFcn(app,
                @LoadCasesSpinnerValueChanged, true);
            app.LoadCasesSpinner.Position = [82 56 53 22];
            app.LoadCasesSpinner.Value = 1;

            % Create ImportExcelButton
            app.ImportExcelButton = uibutton(app.AppliedforceNPanel, 'push');
            app.ImportExcelButton.ButtonPushedFcn = createCallbackFcn(app,
                @ImportExcelButtonPushed, true);
            app.ImportExcelButton.Position = [267 11 100 23];
            app.ImportExcelButton.Text = 'Import Excel';

            % Create FixedconstraintPanel
            app.FixedconstraintPanel = uipanel(app.SetupTab);
            app.FixedconstraintPanel.Title = 'Fixed constraint';
            app.FixedconstraintPanel.Position = [14 216 381 175];

            % Create XdirectionCheckBox
            app.XdirectionCheckBox = uicheckbox(app.FixedconstraintPanel);
            app.XdirectionCheckBox.ValueChangedFcn = createCallbackFcn(app,
                @XdirectionCheckBoxValueChanged, true);
            app.XdirectionCheckBox.Text = 'X direction';
            app.XdirectionCheckBox.Position = [5 121 79 22];

            % Create YdirectionCheckBox
            app.YdirectionCheckBox = uicheckbox(app.FixedconstraintPanel);
            app.YdirectionCheckBox.ValueChangedFcn = createCallbackFcn(app,
                @YdirectionCheckBoxValueChanged, true);
            app.YdirectionCheckBox.Text = 'Y direction';
            app.YdirectionCheckBox.Position = [5 99 78 22];

            % Create ZdirectionCheckBox
            app.ZdirectionCheckBox = uicheckbox(app.FixedconstraintPanel);
            app.ZdirectionCheckBox.ValueChangedFcn = createCallbackFcn(app,
                @ZdirectionCheckBoxValueChanged, true);
            app.ZdirectionCheckBox.Text = 'Z direction';
            app.ZdirectionCheckBox.Position = [5 77 78 22];

            % Create AddConstraintButton
            app.AddConstraintButton = uibutton(app.FixedconstraintPanel, 'push');
```

```
            app.AddConstraintButton.ButtonPushedFcn = createCallbackFcn(app,
                @AddConstraintButtonPushed, true);
            app.AddConstraintButton.Position = [24 10 100 23];
            app.AddConstraintButton.Text = 'Add Constraint';

            % Create ConstraintsListBoxLabel
            app.ConstraintsListBoxLabel = uilabel(app.FixedconstraintPanel);
            app.ConstraintsListBoxLabel.HorizontalAlignment = 'right';
            app.ConstraintsListBoxLabel.Position = [152 122 62 22];
            app.ConstraintsListBoxLabel.Text = 'Constraints';

            % Create ConstraintsListBox
            app.ConstraintsListBox = uilistbox(app.FixedconstraintPanel);
            app.ConstraintsListBox.Items = {};
            app.ConstraintsListBox.ValueChangedFcn = createCallbackFcn(app,
                @ConstraintsListBoxValueChanged, true);
            app.ConstraintsListBox.Position = [218 57 148 89];
            app.ConstraintsListBox.Value = {};

            % Create RemoveConstraintButton
            app.RemoveConstraintButton = uibutton(app.FixedconstraintPanel, 'push');
            app.RemoveConstraintButton.ButtonPushedFcn = createCallbackFcn(app,
                @RemoveConstraintButtonPushed, true);
            app.RemoveConstraintButton.Position = [172 10 118 23];
            app.RemoveConstraintButton.Text = 'Remove Constraint';

            % Create MaterialsPanel
            app.MaterialsPanel = uipanel(app.SetupTab);
            app.MaterialsPanel.Title = 'Materials';
            app.MaterialsPanel.Position = [638 6 558 212];

            % Create YoungsModulusMPaSpinnerLabel
            app.YoungsModulusMPaSpinnerLabel = uilabel(app.MaterialsPanel);
            app.YoungsModulusMPaSpinnerLabel.HorizontalAlignment = 'right';
            app.YoungsModulusMPaSpinnerLabel.Position = [37 138 131 22];
            app.YoungsModulusMPaSpinnerLabel.Text = 'Young''s Modulus (MPa)';

            % Create YoungsModulusSpinner
            app.YoungsModulusSpinner = uispinner(app.MaterialsPanel);
            app.YoungsModulusSpinner.Limits = [0.1 Inf];
            app.YoungsModulusSpinner.ValueChangedFcn = createCallbackFcn(app,
                @YoungsModulusSpinnerValueChanged, true);
            app.YoungsModulusSpinner.Position = [177 138 97 22];
            app.YoungsModulusSpinner.Value = 206000;

            % Create Densitykgm3SpinnerLabel
            app.Densitykgm3SpinnerLabel = uilabel(app.MaterialsPanel);
            app.Densitykgm3SpinnerLabel.HorizontalAlignment = 'right';
            app.Densitykgm3SpinnerLabel.Position = [37 117 95 22];
            app.Densitykgm3SpinnerLabel.Text = 'Density (kg/m^3)';

            % Create DensitySpinner
            app.DensitySpinner = uispinner(app.MaterialsPanel);
            app.DensitySpinner.Limits = [0.1 Inf];
            app.DensitySpinner.ValueChangedFcn = createCallbackFcn(app,
                @DensitySpinnerValueChanged, true);
            app.DensitySpinner.Position = [177 117 97 22];
            app.DensitySpinner.Value = 7850;

            % Create PoissonratioSpinner_2Label
            app.PoissonratioSpinner_2Label = uilabel(app.MaterialsPanel);
            app.PoissonratioSpinner_2Label.HorizontalAlignment = 'right';
            app.PoissonratioSpinner_2Label.Position = [37 96 74 22];
            app.PoissonratioSpinner_2Label.Text = 'Poisson ratio';

            % Create PoissonratioSpinner
            app.PoissonratioSpinner = uispinner(app.MaterialsPanel);
            app.PoissonratioSpinner.Limits = [0 1];
            app.PoissonratioSpinner.ValueChangedFcn = createCallbackFcn(app,
                @PoissonratioSpinnerValueChanged, true);
            app.PoissonratioSpinner.Position = [177 96 97 22];
```

```
            app.PoissonratioSpinner.Value = 0.3;

            % Create MaterialNameEditFieldLabel
            app.MaterialNameEditFieldLabel = uilabel(app.MaterialsPanel);
            app.MaterialNameEditFieldLabel.HorizontalAlignment = 'right';
            app.MaterialNameEditFieldLabel.Position = [38 162 83 22];
            app.MaterialNameEditFieldLabel.Text = 'Material Name';

            % Create MaterialNameEditField
            app.MaterialNameEditField = uieditfield(app.MaterialsPanel, 'text');
            app.MaterialNameEditField.Position = [136 162 139 22];
            app.MaterialNameEditField.Value = 'Structural steel S235JR';

            % Create PresetsDropDownLabel
            app.PresetsDropDownLabel = uilabel(app.MaterialsPanel);
            app.PresetsDropDownLabel.HorizontalAlignment = 'right';
            app.PresetsDropDownLabel.Position = [298 162 46 22];
            app.PresetsDropDownLabel.Text = 'Presets';

            % Create PresetsDropDown
            app.PresetsDropDown = uidropdown(app.MaterialsPanel);
            app.PresetsDropDown.Items = {'Structural steel S235JR', 'Aluminum AlSi12', '
                Titanium Alloy'};
            app.PresetsDropDown.ValueChangedFcn = createCallbackFcn(app,
                @PresetsDropDownValueChanged, true);
            app.PresetsDropDown.Position = [359 162 122 22];
            app.PresetsDropDown.Value = 'Structural steel S235JR';

            % Create UITableMaterials
            app.UITableMaterials = uitable(app.MaterialsPanel);
            app.UITableMaterials.ColumnName = {'Material'; 'Young''s Modulus (MPa)'; 'Density
                 (kg/m^3)'; 'Poisson ratio'};
            app.UITableMaterials.RowName = {};
            app.UITableMaterials.SelectionChangedFcn = createCallbackFcn(app,
                @UITableMaterialsSelectionChanged, true);
            app.UITableMaterials.Position = [24 6 512 80];

            % Create AddMaterialButton
            app.AddMaterialButton = uibutton(app.MaterialsPanel, 'push');
            app.AddMaterialButton.ButtonPushedFcn = createCallbackFcn(app,
                @AddMaterialButtonPushed, true);
            app.AddMaterialButton.Position = [319 104 100 23];
            app.AddMaterialButton.Text = 'Add Material';

            % Create RemoveMaterialButton
            app.RemoveMaterialButton = uibutton(app.MaterialsPanel, 'push');
            app.RemoveMaterialButton.ButtonPushedFcn = createCallbackFcn(app,
                @RemoveMaterialButtonPushed, true);
            app.RemoveMaterialButton.Position = [431 104 106 23];
            app.RemoveMaterialButton.Text = 'Remove Material';

            % Create DensityPlotsTab
            app.DensityPlotsTab = uitab(app.TabGroup);
            app.DensityPlotsTab.AutoResizeChildren = 'off';
            app.DensityPlotsTab.SizeChangedFcn = createCallbackFcn(app,
                @DensityPlotsTabSizeChanged, true);
            app.DensityPlotsTab.Title = 'Density Plots';
            app.DensityPlotsTab.Scrollable = 'on';

            % Create ComplianceMassGraphTab
            app.ComplianceMassGraphTab = uitab(app.TabGroup);
            app.ComplianceMassGraphTab.Title = 'Compliance-Mass Graph';

            % Create UIAxes
            app.UIAxes = uiaxes(app.ComplianceMassGraphTab);
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.Position = [2 232 597 381];
```

```
            % Create UIAxesSelection
            app.UIAxesSelection = uiaxes(app.ComplianceMassGraphTab);
            xlabel(app.UIAxesSelection, 'X')
            ylabel(app.UIAxesSelection, 'Y')
            zlabel(app.UIAxesSelection, 'Z')
            app.UIAxesSelection.XTickLabel = '';
            app.UIAxesSelection.YTickLabel = '';
            app.UIAxesSelection.Position = [719 311 492 302];

            % Create UIAxesSelection_2
            app.UIAxesSelection_2 = uiaxes(app.ComplianceMassGraphTab);
            xlabel(app.UIAxesSelection_2, 'X')
            ylabel(app.UIAxesSelection_2, 'Y')
            zlabel(app.UIAxesSelection_2, 'Z')
            app.UIAxesSelection_2.XTickLabel = '';
            app.UIAxesSelection_2.YTickLabel = '';
            app.UIAxesSelection_2.Position = [719 4 492 302];

            % Create UITableSelection
            app.UITableSelection = uitable(app.ComplianceMassGraphTab);
            app.UITableSelection.ColumnName = {'Compliance (J)'; 'Mass (g)'; 'Max VonMises
                Stress (MPa)'; 'Volume Fraction'; 'Filter Radius'; 'Material'};
            app.UITableSelection.RowName = {};
            app.UITableSelection.Position = [64 58 653 81];

            % Create ShowDesignDataButton
            app.ShowDesignDataButton = uibutton(app.ComplianceMassGraphTab, 'push');
            app.ShowDesignDataButton.ButtonPushedFcn = createCallbackFcn(app,
                @ShowDesignDataButtonPushed, true);
            app.ShowDesignDataButton.Position = [65 187 114 23];
            app.ShowDesignDataButton.Text = 'Show Design Data';

            % Create ExportSelectedSolutionasSTLButton
            app.ExportSelectedSolutionasSTLButton = uibutton(app.ComplianceMassGraphTab, '
                push');
            app.ExportSelectedSolutionasSTLButton.ButtonPushedFcn = createCallbackFcn(app,
                @ExportSelectedSolutionasSTLButtonPushed, true);
            app.ExportSelectedSolutionasSTLButton.Position = [196 187 192 23];
            app.ExportSelectedSolutionasSTLButton.Text = 'Export Selected Solution as .STL';

            % Create DataTableOverview
            app.DataTableOverview = uitab(app.TabGroup);
            app.DataTableOverview.Title = 'Data Table Overview';

            % Create UIAxesTableSelection
            app.UIAxesTableSelection = uiaxes(app.DataTableOverview);
            title(app.UIAxesTableSelection, 'Title')
            xlabel(app.UIAxesTableSelection, 'X')
            ylabel(app.UIAxesTableSelection, 'Y')
            zlabel(app.UIAxesTableSelection, 'Z')
            app.UIAxesTableSelection.XTickLabel = '';
            app.UIAxesTableSelection.YTickLabel = '';
            app.UIAxesTableSelection.Position = [724 312 492 301];

            % Create UIAxesTableSelection_2
            app.UIAxesTableSelection_2 = uiaxes(app.DataTableOverview);
            title(app.UIAxesTableSelection_2, 'Title')
            xlabel(app.UIAxesTableSelection_2, 'X')
            ylabel(app.UIAxesTableSelection_2, 'Y')
            zlabel(app.UIAxesTableSelection_2, 'Z')
            app.UIAxesTableSelection_2.XTickLabel = '';
            app.UIAxesTableSelection_2.YTickLabel = '';
            app.UIAxesTableSelection_2.Position = [724 5 492 301];

            % Create UITableData
            app.UITableData = uitable(app.DataTableOverview);
            app.UITableData.ColumnName = {'Compliance (J)'; 'Mass (g)'; 'Max VonMises Stress
                (MPa)'; 'Volume Fraction'; 'Filter Radius'; 'Material'};
            app.UITableData.RowName = {};
```

```
            app.UITableData.SelectionChangedFcn = createCallbackFcn(app,
                @UITableDataSelectionChanged, true);
            app.UITableData.Position = [8 35 710 572];

            % Create ExportSelectedSolutionasSTLButton_2
            app.ExportSelectedSolutionasSTLButton_2 = uibutton(app.DataTableOverview, 'push')
                ;
            app.ExportSelectedSolutionasSTLButton_2.ButtonPushedFcn = createCallbackFcn(app,
                @ExportSelectedSolutionasSTLButton_2Pushed, true);
            app.ExportSelectedSolutionasSTLButton_2.Position = [524 5 192 23];
            app.ExportSelectedSolutionasSTLButton_2.Text = 'Export Selected Solution as .STL'
                ;

            % Create TextArea
            app.TextArea = uitextarea(app.UIFigure);
            app.TextArea.ValueChangedFcn = createCallbackFcn(app, @TextAreaValueChanged, true
                );
            app.TextArea.FontWeight = 'bold';
            app.TextArea.FontColor = [1 0 0];
            app.TextArea.Position = [243 14 648 40];

            % Create RunButton
            app.RunButton = uibutton(app.UIFigure, 'push');
            app.RunButton.ButtonPushedFcn = createCallbackFcn(app, @RunButtonPushed, true);
            app.RunButton.FontWeight = 'bold';
            app.RunButton.Position = [96 24 100 23];
            app.RunButton.Text = 'Run';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = TOPGD_Apd

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

B

Experiment Design Assignments

In this Appendix, the three Design Assignments as they ware given to the participants during the experiment are annexed.

B.1. Assignment A

You will be designing a structural arm. The .STL file of your design space looks like shown in Figure B.1: The rectangular box between the rings is 600 by 300mm.

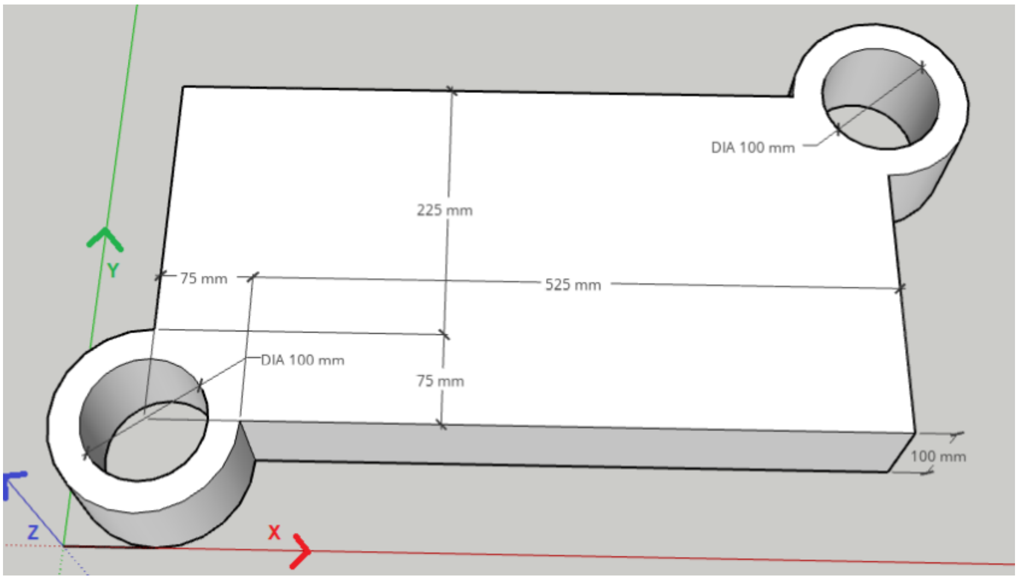


Figure B.1: Assignment A: Design space

The structural arm should have:

- a fixed constraint in all directions defined at the surface of the upper right mounting hole (red)
- a horizontal force applied to the surface of the lower left mounting hole (green), of 1000 N in the positive Y-direction
- and solid areas (black) defined around both mounting holes as shown in Figure B.2.

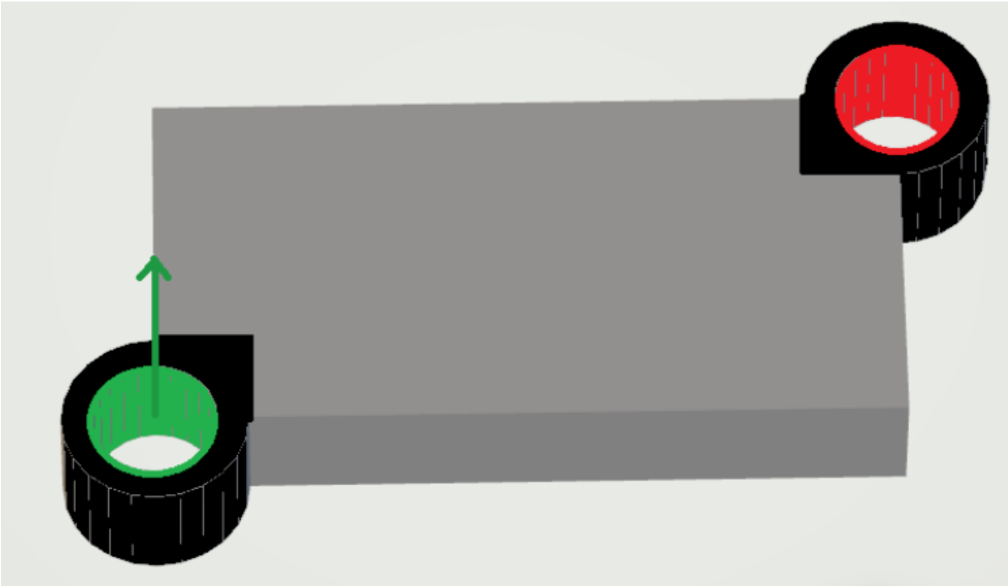


Figure B.2: Assignment A: Fixed Constraint, Solid Areas and Force Surface

B.2. Assignment B

You will be designing a bracket. The .STL file of your design space looks like shown in Figure B.3:

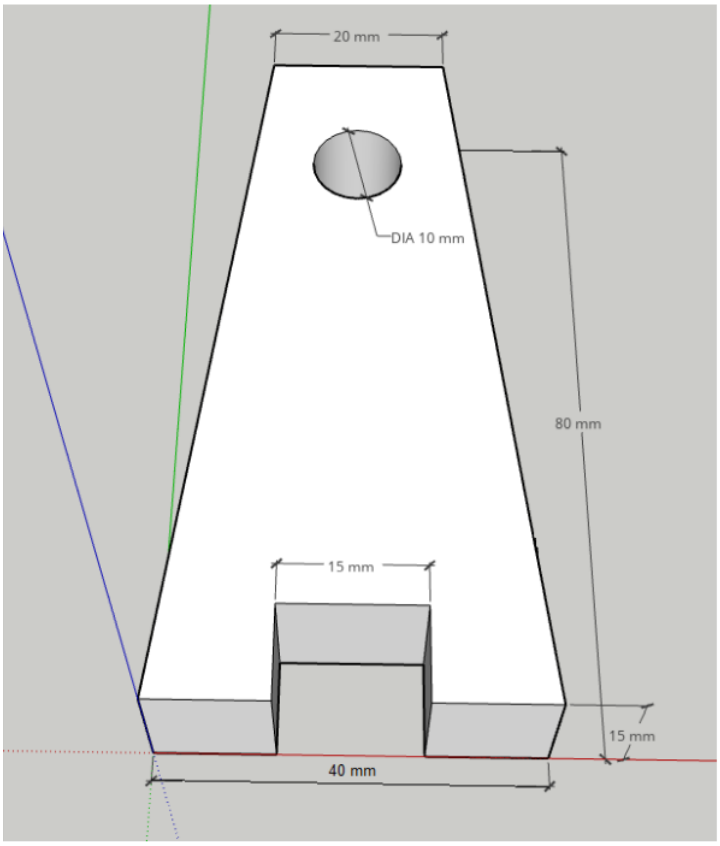


Figure B.3: Assignment B: Design space

The bracket should have a fixed constraint in all directions defined at the surfaces of the bottom legs (red), and solid areas (black) defined around the hole as shown in Figure B.4:

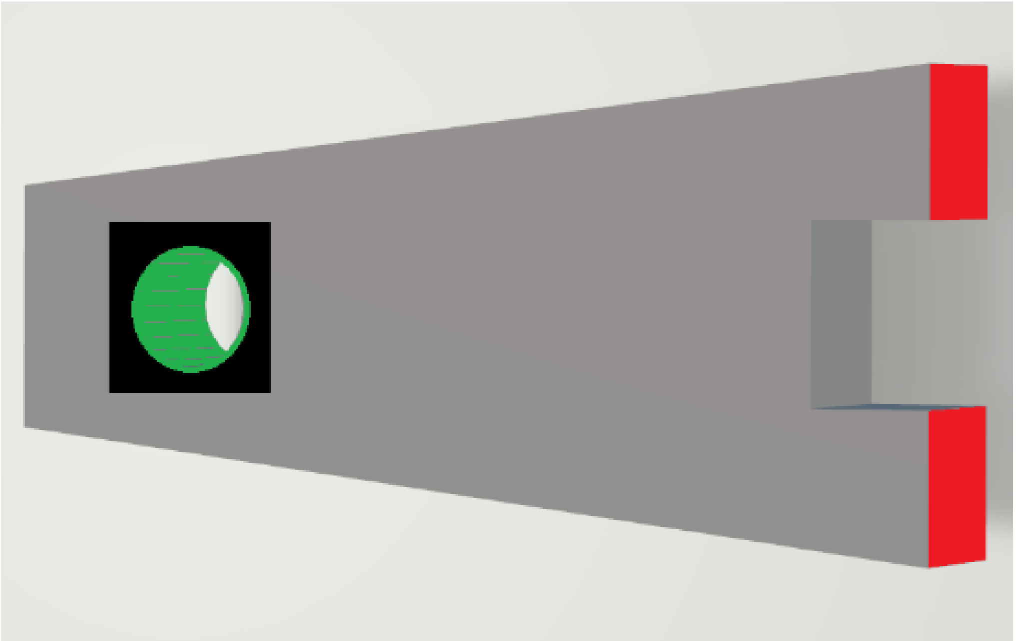


Figure B.4: Assignment B: Fixed Constraint and Solid Areas

There are 2 load cases for this bracket design problem, shown in Figure B.5.

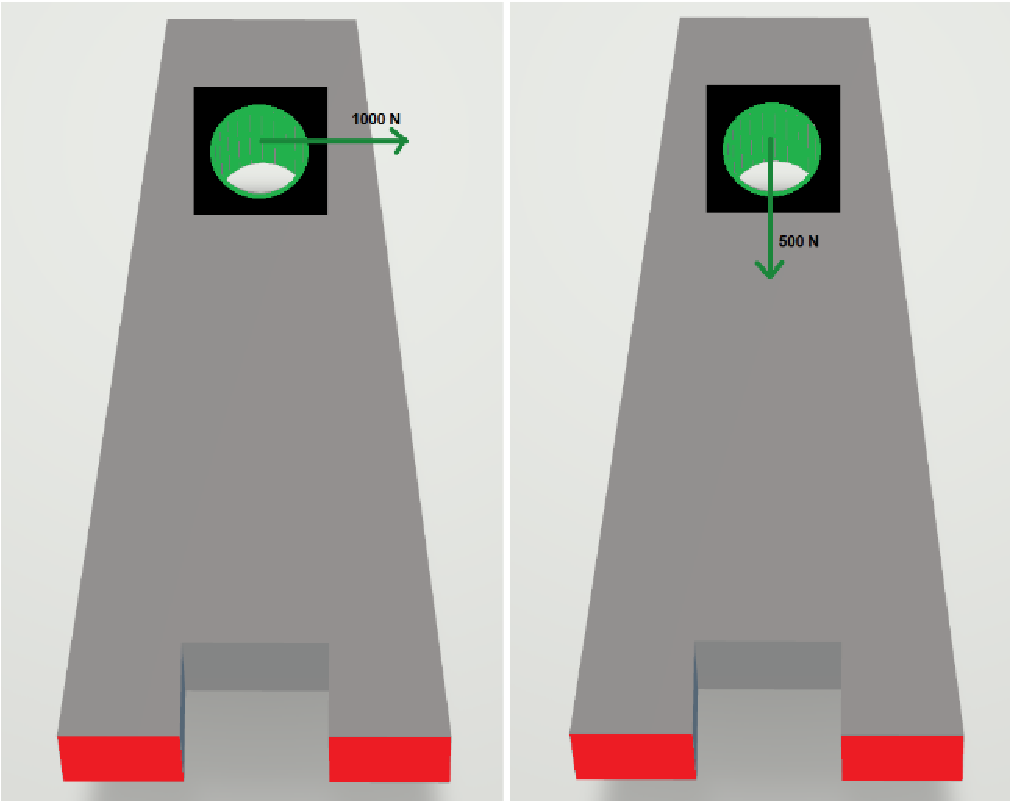


Figure B.5: Assignment B: Load Cases

Load case 1: A distributed load on the surface of the hole in Figure B.5, of 1000 N sideways

Load case 2: A distributed load on the surface of the hole in Figure B.5, of 500 N towards the bottom legs

B.3. Assignment C

You will be designing a kitchen step. The .STL file of your design space looks like shown in Figure B.6:

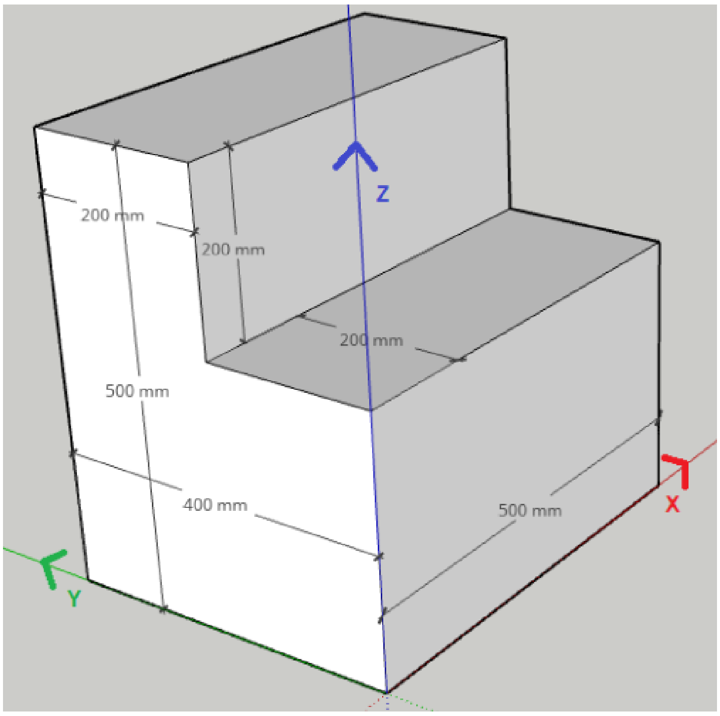


Figure B.6: Assignment C: Design space

The kitchen step should have a fixed constraint in all directions defined at the bottom surface (red), and solid areas (black) defined for the two steps as shown in Figure B.7.

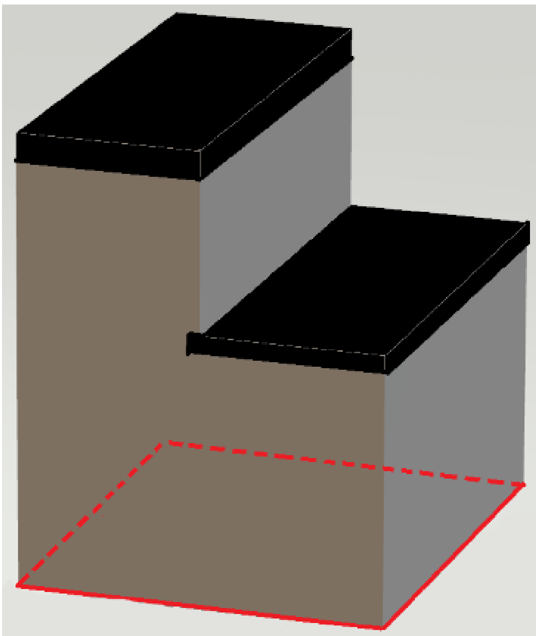


Figure B.7: Assignment C: Fixed Constraint and Solid Areas

There are 3 load cases for this kitchen step design problem.

Load case 1: Standing on the first step, shown in Figure B.8 below. There is one distributed load of 800N downwards attached to the green surface highlighted in the figure.

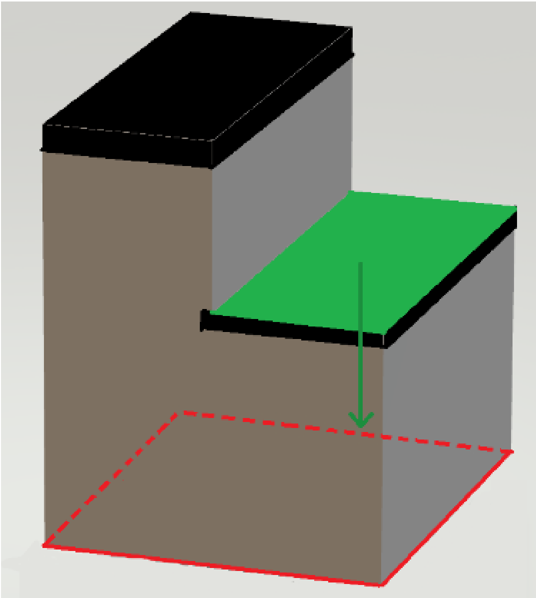


Figure B.8: Assignment C: Load case 1, standing on the first step

Load case 2: Standing on the second step, shown in Figure B.9 below. There is one distributed load of 800N downwards attached to the green surface highlighted in the figure.

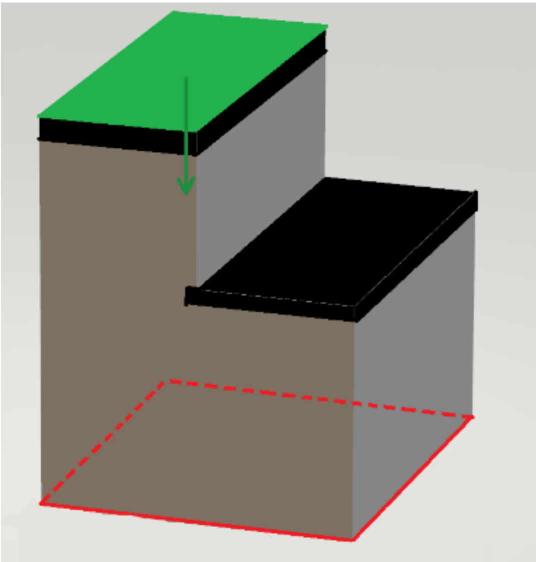


Figure B.9: Assignment C: Load case 2, standing on the second step

Load case 3: Sitting on the step, shown in Figure B.10 below. There are two distributed loads attached to the green surfaces highlighted in the figure. F1 is a load downwards of 600 N, and F2 is a horizontal load backwards of 200 N.

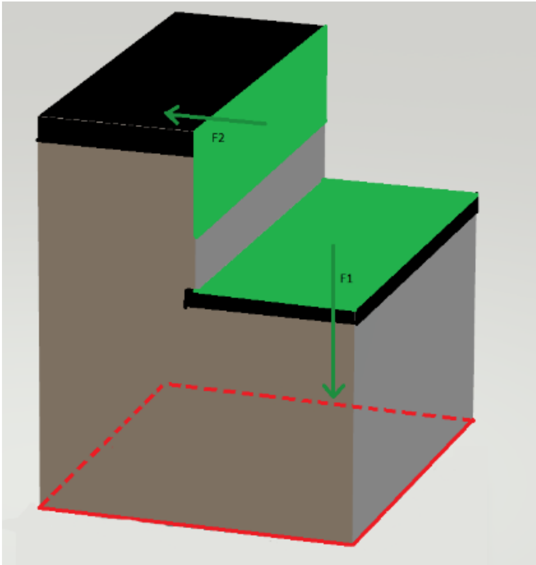


Figure B.10: Assignment C: Load case 3, sitting on the step

Possible Material parameters could be those of Oakwood:
Young's Modulus: 11 GPa (with the grain) & Density: 600 kg/m3

or HDPE:
Young's Modulus: 800 MPa & Density: 970 kg/m3

C

Experiment Survey

Starting from the next page, the Google Form is annexed that was used to guide the participants through the experiment, including all introduction videos and survey questions asked.

Topology Optimization Experiment

1. What is your Engineering Backround? (e.g. Mechanical Engineering, Structural Engineering)

2. What is your background knowledge on Topology Optimization?

Markeer slechts één ovaal.

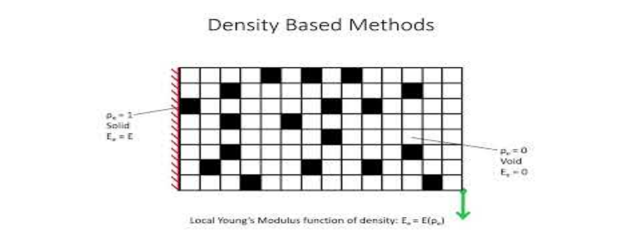
I have never heard of it

1 ⬭
2 ⬭
3 ⬭
4 ⬭
5 ⬭

I am experienced with it

Video Introduction to Topology Optimization

[redacted]



[redacted] http://youtube.com/watch?v=hmw3SqCsua0

3. Do you understand the basics of topology optimization after seeing the explanation video?

   Markeer slechts één ovaal.

   - ( ) Yes
   - ( ) No
   - ( ) Anders: ______________________

Manual Designing

Assignment 1

4. How hard do you think the first design problem was? (Not the manual designing process, but the problem itself)

Markeer slechts één ovaal.

Very Easy

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Very Hard

5. How confident are you that you have found the optimal solution for this first design problem?

Markeer slechts één ovaal.

Not Confident	
1	◯
2	◯
3	◯
4	◯
5	◯
Very Confident	

6. Do you understand what are structurally the most important areas of this part?

Markeer slechts één ovaal.

	No idea
1	◯
2	◯
3	◯
4	◯
5	◯
	Yes, clearly

7. How would you rate the manual design process?

Markeer slechts één ovaal.

	Very Easy
1	◯
2	◯
3	◯
4	◯
5	◯
	Very Hard

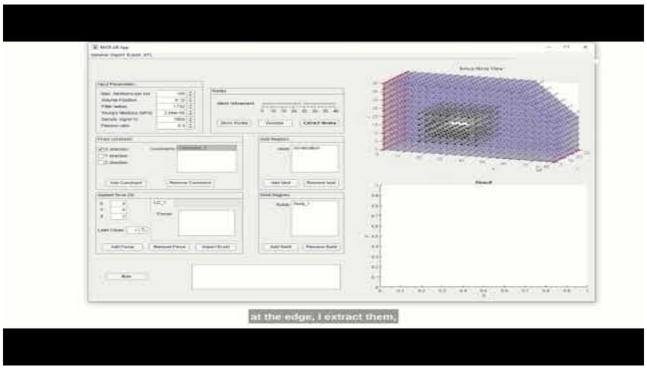
8. What are the positive aspects of manually designing?

9. What are the negative aspects of manually designing, or what could be improved?

Basic Topology Optimization App

Assignment 2

Video Tutorial Basic TO App



http://youtube.com/watch?v=FpK0-JoCjkI

10. Do you think the tutorial of this app, previous to the experiment, was clear?

Markeer slechts één ovaal.

	Not Clear
1	⬭
2	⬭
3	⬭
4	⬭
5	⬭
	Very Clear

11. How hard do you think this design problem was? (Not the design process with the app, but the problem itself)

Markeer slechts één ovaal.

	Very Easy
1	◯
2	◯
3	◯
4	◯
5	◯
	Very Hard

12. How confident are you that you have found the optimal solution for this design problem?

Markeer slechts één ovaal.

	Not Confident
1	◯
2	◯
3	◯
4	◯
5	◯
	Very Confident

13. Do you understand what are structurally the most important areas of this part?

Markeer slechts één ovaal.

	No idea
1	◯
2	◯
3	◯
4	◯
5	◯
	Yes, clearly

14. What was the overall experience of using this app in the design process?

Markeer slechts één ovaal.

	Bad Experience
1	◯
2	◯
3	◯
4	◯
5	◯
	Good Experience

15. Does the use of this app improve your understanding of the design problems compared to manually designing?

Markeer slechts één ovaal.

	Not at all
1	◯
2	◯
3	◯
4	◯
5	◯
	Yes a lot

16. How would you rate the Basic App on user-friendliness?

Markeer slechts één ovaal.

Hard to use

1 ◯
2 ◯
3 ◯
4 ◯
5 ◯

Easy to use

17. What are positive aspects of the app, or designing with the app?

____
____
____
____
____

18. What are negative aspects of the app, or things that need improvement?

____
____
____
____
____

19. How would you rate the overview generated by the app?

Markeer slechts één ovaal.

Poor overview
1 ◯
2 ◯
3 ◯
4 ◯
5 ◯
Clear overview

20. Would you consider the use of this app in the design process as an improvement?

Markeer slechts één ovaal.

No improvement

1 ◯

2 ◯

3 ◯

4 ◯

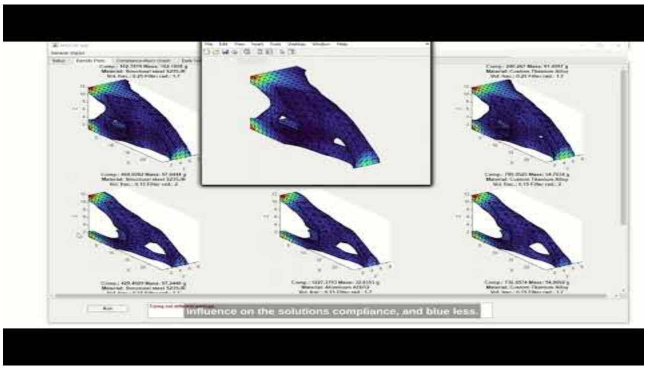
5 ◯

Big improvement

Advanced Topology Optimization App

Assignment 3

Video Tutorial Advanced TO App



http://youtube.com/watch?v=UauV1bRjx8M

21. Do you think the tutorial of this app, previous to the experiment, was clear?

Markeer slechts één ovaal.

Not Clear	
1	◯
2	◯
3	◯
4	◯
5	◯
Very Clear	

22. How hard do you think this design problem was? (Not the designing process with the app, but the problem itself)

Markeer slechts één ovaal.

Very Easy

1 ⬭

2 ⬭

3 ⬭

4 ⬭

5 ⬭

Very Hard

23. How confident are you that you have found the optimal solution for this design problem?

Markeer slechts één ovaal.

	Not Confident
1	◯
2	◯
3	◯
4	◯
5	◯
	Very Confident

24. Do you understand what are structurally the most important areas of this part?

Markeer slechts één ovaal.

No idea
1 ⬭
2 ⬭
3 ⬭
4 ⬭
5 ⬭
Yes, clearly

25. What was the overall experience of using this app in the design process?

Markeer slechts één ovaal.

Bad Experience
1 ⬭
2 ⬭
3 ⬭
4 ⬭
5 ⬭
Good Experience

26. How would you rate this app on user-friendliness?

Markeer slechts één ovaal.

	Hard to use
1	◯
2	◯
3	◯
4	◯
5	◯
	Easy to use

27. Does the use of this app improve your understanding of the design problems compared to manually designing?

Markeer slechts één ovaal.

Not at all

1 ⬭
2 ⬭
3 ⬭
4 ⬭
5 ⬭

Yes a lot

28. Does the use of this app improve your understanding of the design problems compared to using the previous basic app for designing?

Markeer slechts één ovaal.

Not at all
1
2
3
4
5
Yes a lot

29. Does the use of this app improve your understanding of Topology Optimization and its settings compared to using the previous basic app for designing?

Markeer slechts één ovaal.

Not at all

1 ○

2 ○

3 ○

4 ○

5 ○

Yes a lot

30. How would you rate the overview generated by the advanced app, compared to the basic app?

Markeer slechts één ovaal.

Poor overview

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Clear overview

31. What are positive aspects of the app, or designing with the app?

32. What are negative aspects of the app, or things that need improvement?

33. Would you consider the use of this app in the design process as an improvement?

Markeer slechts één ovaal.

No improvement

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Big improvement

D
Gaze Density Heat Maps

In this appendix, the heat map of the average gaze location data collected during the eye-tracking experiment and shown in Figure 5.12, Figure 5.13 and Figure 5.14 have been separated into multiple heat maps representing the 3 participants and showing any differences of the gaze density plots between them.

D.1. Basic TO tool heat maps

The gaze density data of each participant has been plotted over the Basic TO GUI in the figures below.

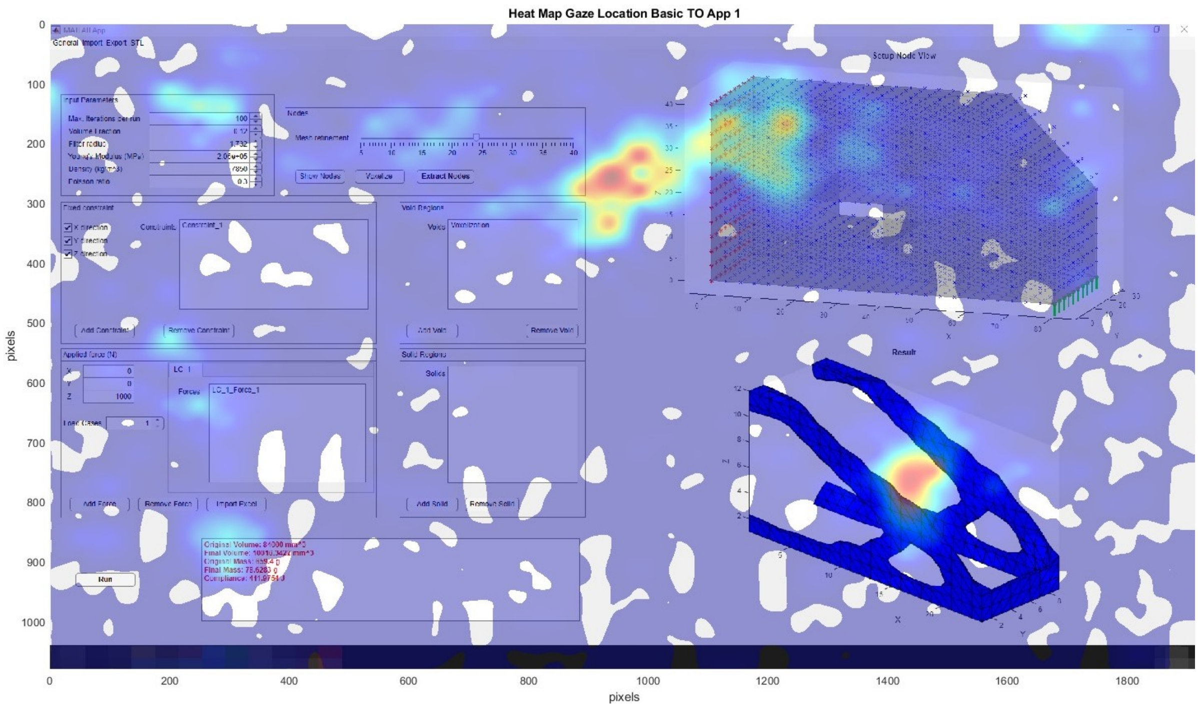


Figure D.1: Heat Map Gaze Density Basic TO tool Participant 1

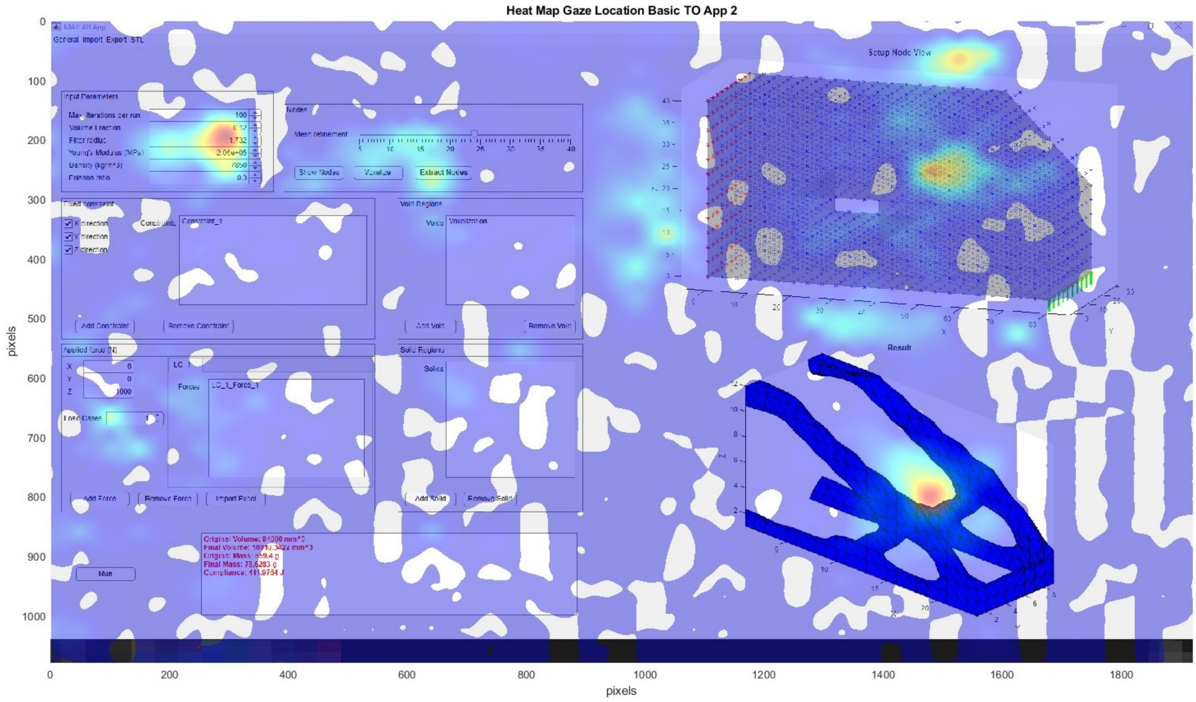


Figure D.2: Heat Map Gaze Density Basic TO tool Participant 2

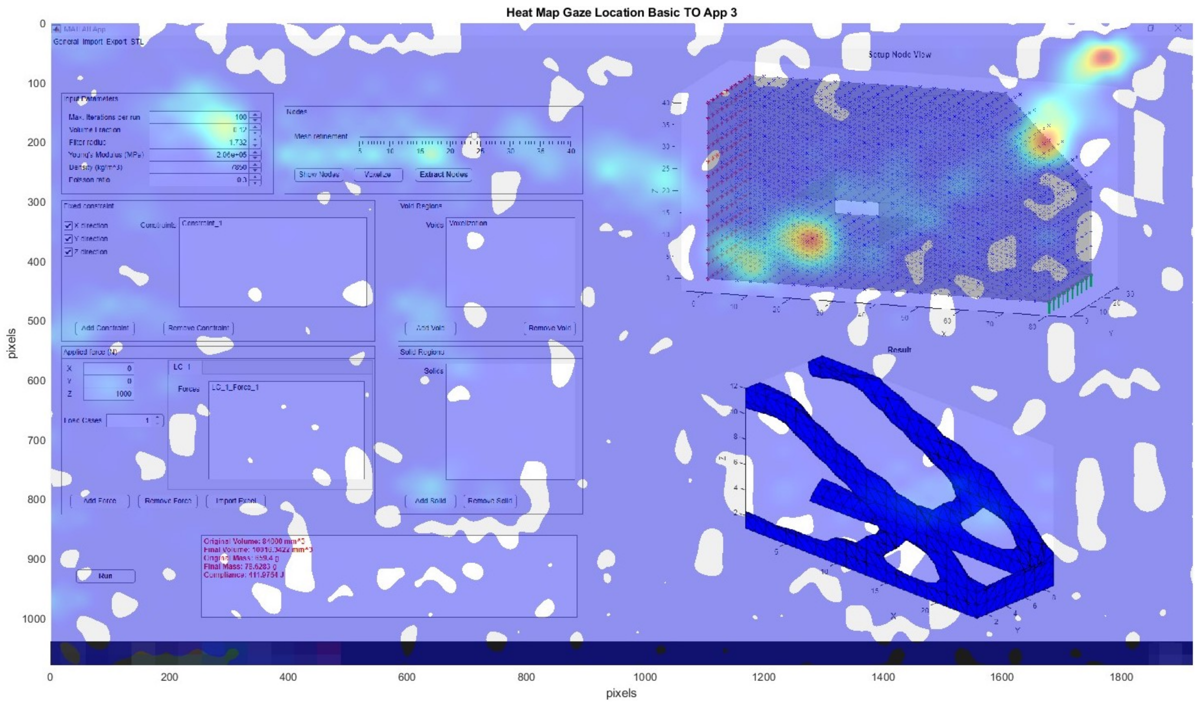


Figure D.3: Heat Map Gaze Density Basic TO tool Participant 3

D.2. TOP-GD tool heat maps

The gaze density data of each participant has been plotted over the Setup tab (Figure D.4, Figure D.5 and Figure D.6) and the Compliance-Mass Graph tab (Figure D.7, Figure D.8 and Figure D.7) of the TOP-GD GUI below.

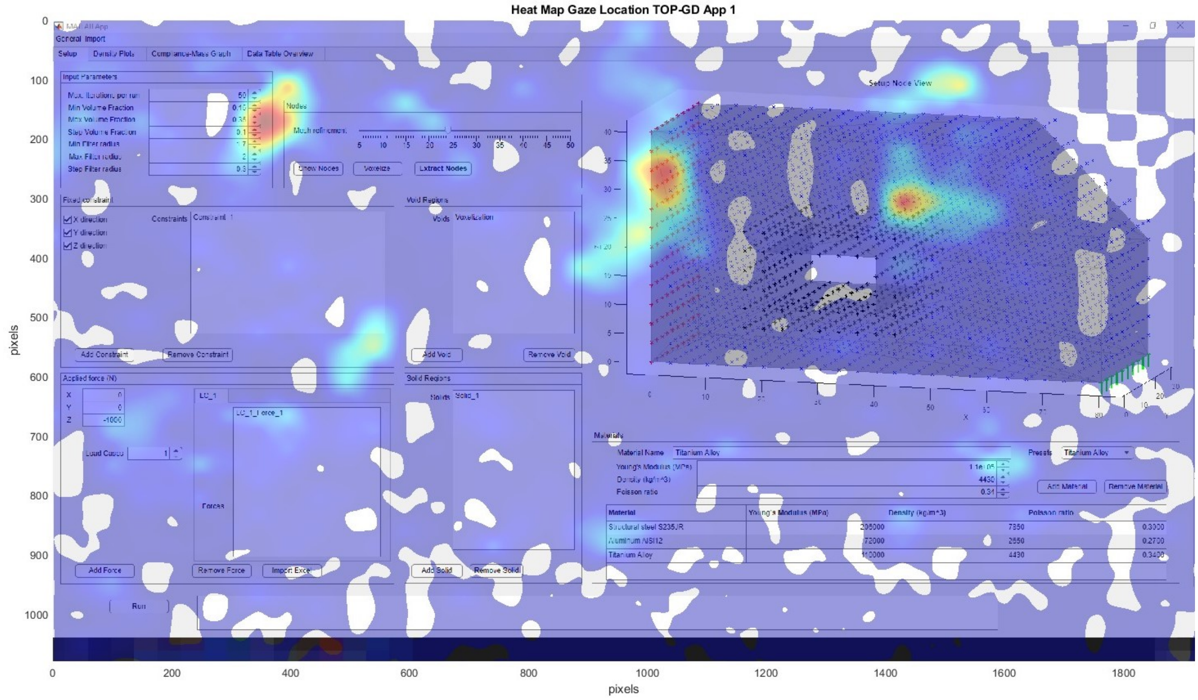


Figure D.4: Heat Map Gaze Density TOP-GD tool Setup tab Participant 1

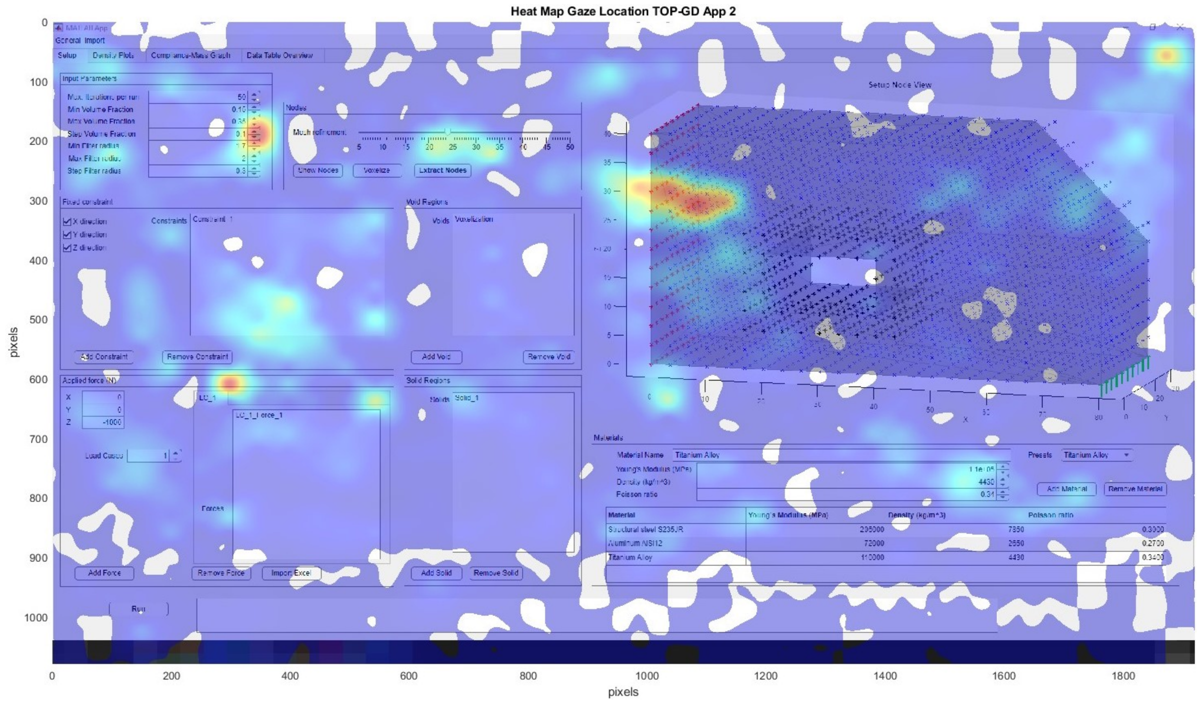


Figure D.5: Heat Map Gaze Density TOP-GD tool Setup tab Participant 2

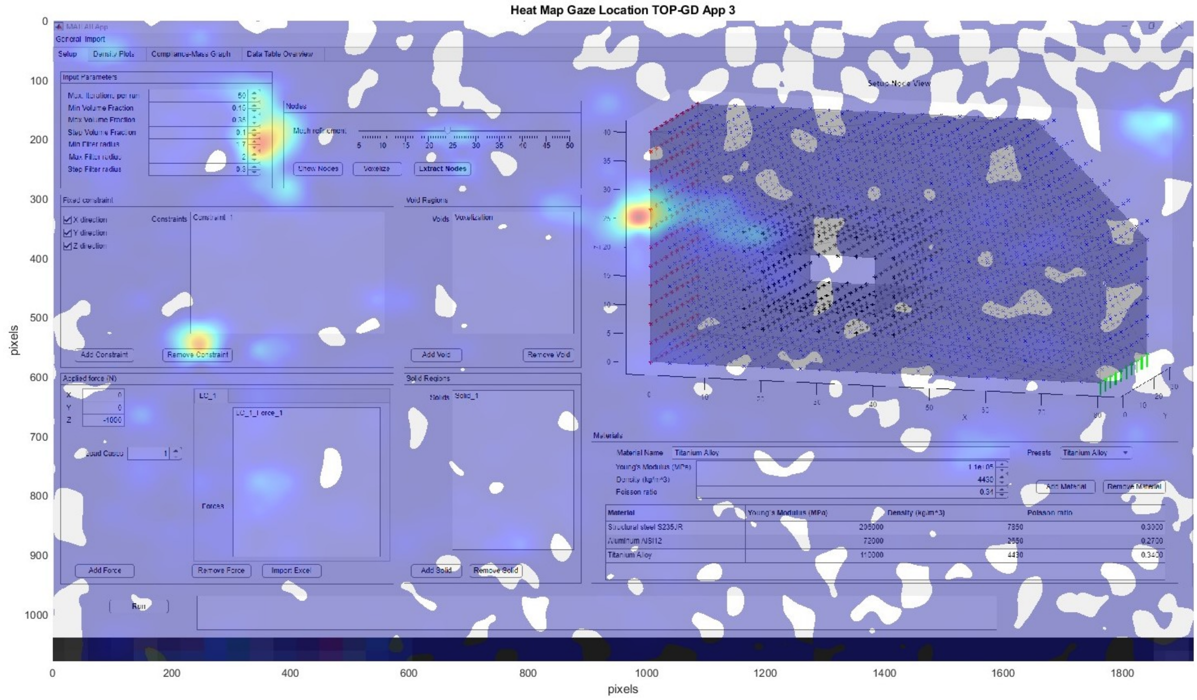


Figure D.6: Heat Map Gaze Density TOP-GD tool Setup tab Participant 3

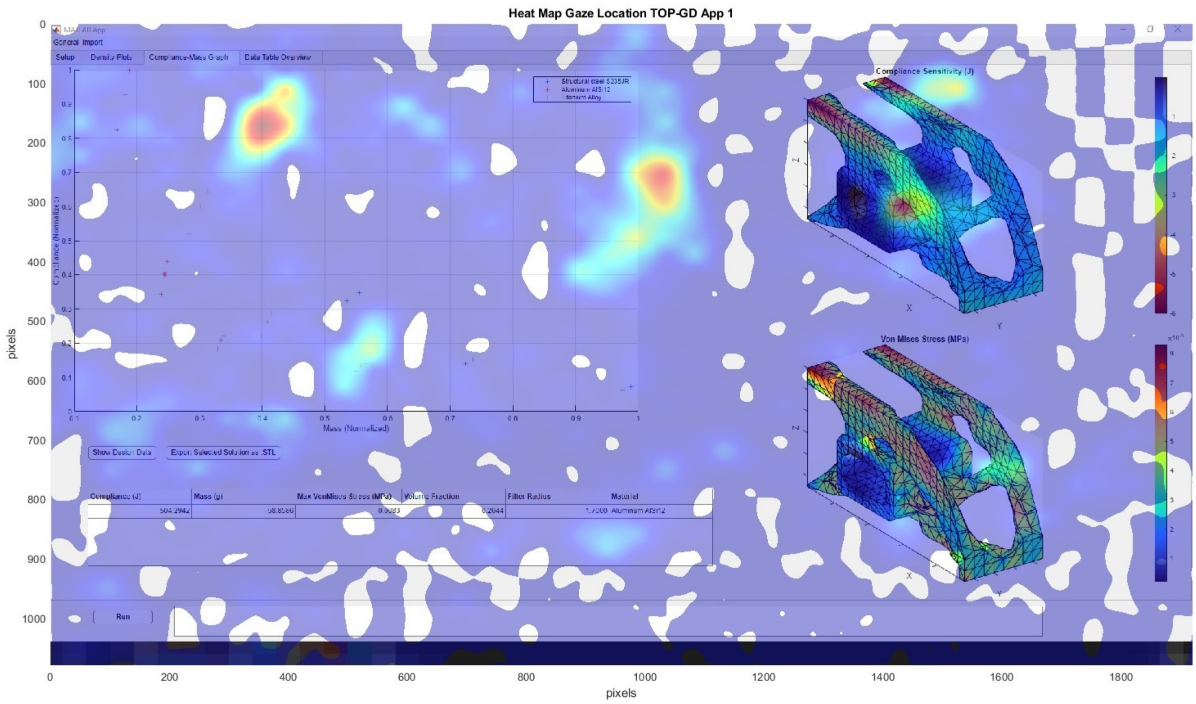


Figure D.7: Heat Map Gaze Density TOP-GD tool Compliance-Mass Graph tab Participant 1

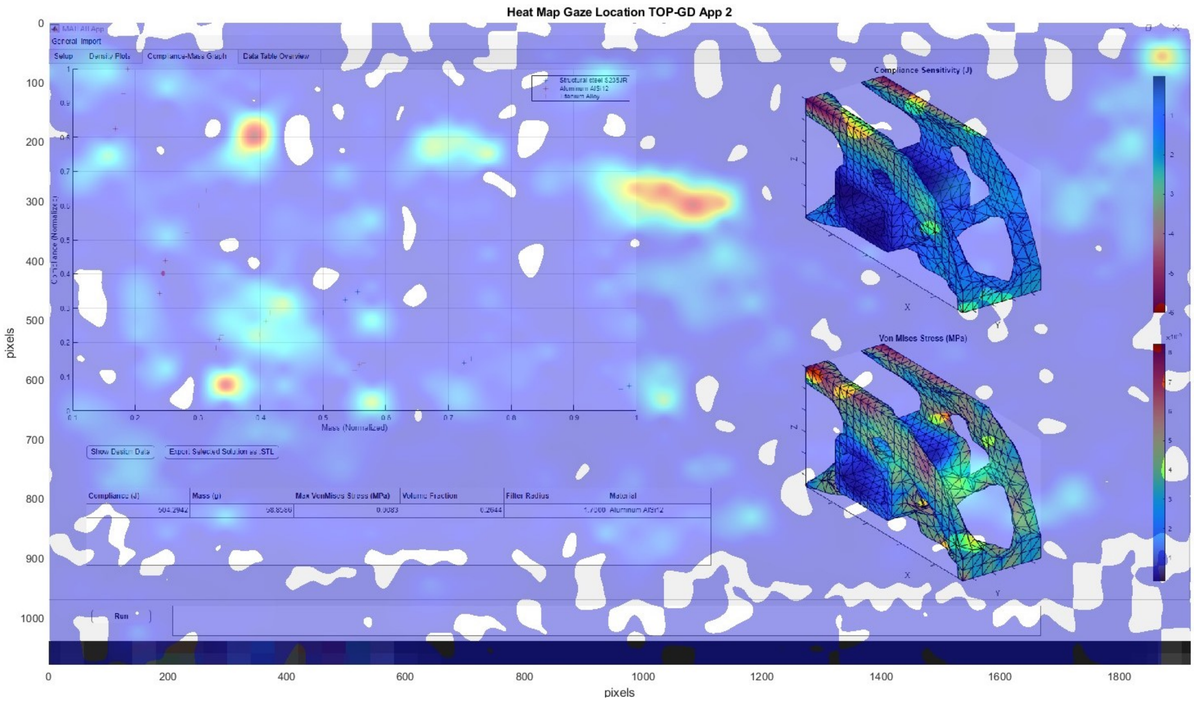


Figure D.8: Heat Map Gaze Density TOP-GD tool Compliance-Mass Graph tab Participant 2

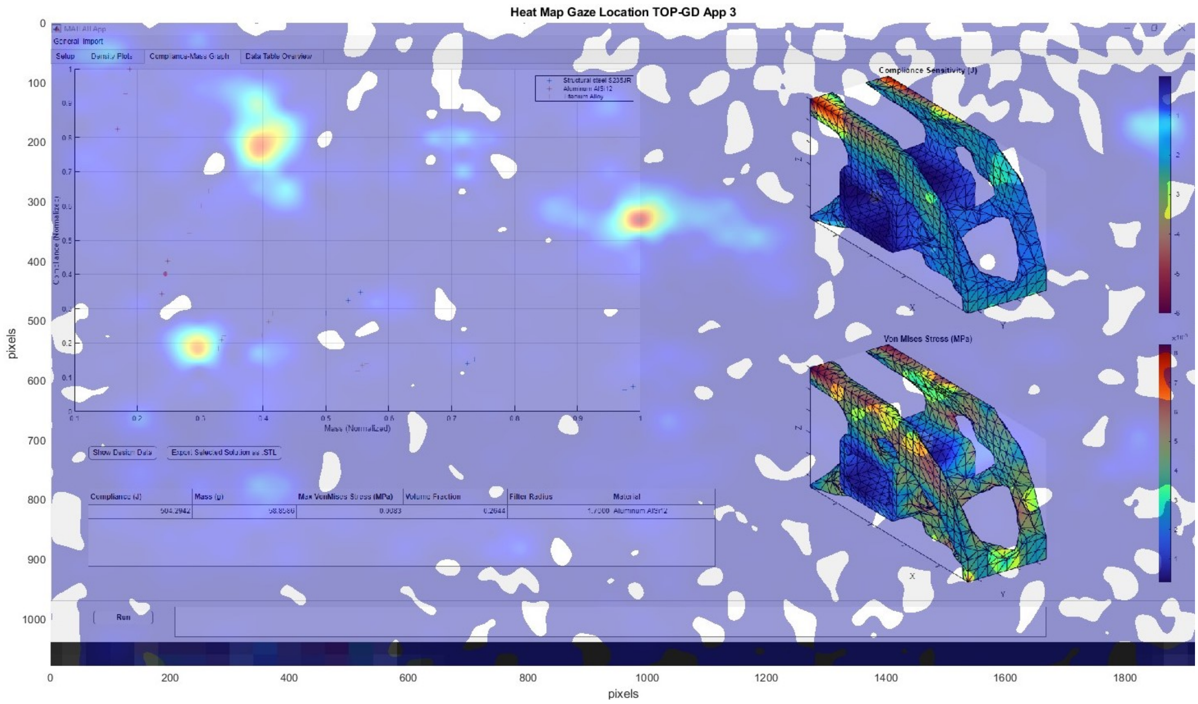


Figure D.9: Heat Map Gaze Density TOP-GD tool Compliance-Mass Graph tab Participant 3