

Jørgen Petlund  
Morten Jørgensen  
Bartosz Jakubowski

# **Sensitivity Analysis of Topology Optimization Programs Using the SIMP, Optimality Criteria and Level-Set Method**

**May 2019**

**NTNU**

Norwegian University of Science and Technology  
Faculty of Engineering  
Department of Mechanical and Industrial Engineering

**Bachelor's thesis**

**2019**





Jørgen Petlund  
Morten Jørgensen  
Bartosz Jakubowski

# **Sensitivity Analysis of Topology Optimization Programs Using the SIMP, Optimality Criteria and Level-Set Method**

Bachelor's thesis  
May 2019

**NTNU**  
Norwegian University of Science and Technology  
Faculty of Engineering  
Department of Mechanical and Industrial Engineering



Norwegian University of  
Science and Technology



---

### *Dedication*

The thesis is written as a product of a bachelor's degree carried out at the Department of Mechanical engineering at the Norwegian University of Science and Technology during the spring of 2019. The thesis serves as a comparison between practical differences in the results obtained from four unique topology optimization algorithms. The results may in the future contribute in the research towards a Ph.D by Evangelos Tyflopoulos.

The work has given us a good general knowledge about TO, and a better understanding of the practical differences between the most used algorithms used in a static load bearing structure problems. We have worked with a wide variety of software and operating systems, including windows, macOS, and a linux based supercomputer provided by NTNU. The knowledge gained from our work this spring, has given us a comprehensive knowledge in the ever more popular field of TO and additive manufacturing.

We would like to thank our supervisor Evangelos Tyflopoulos for guiding us along the way and constructing the thesis task. We would also like to tank John Floan, chef engineer in the IT-development section, and Freddy Barstad, senior adviser in the IT- strategy and -steering section at NTNU, for their companionship, useful expertise and help with acquiring the necessary licenses, and installing the needed software on NTNU's supercomputer.

---





---

# Preface

This thesis is written as a product of the bachelor's thesis work carried out at the Department of Mechanical Engineering and Production at the Norwegian University of Science and Technology during the spring of 2019. This thesis serves as a contribution to Ph.D dissertation of Evangelos Tyflopoulos, as well as aiming at increasing knowledge of topology optimization in general.

The work has been rewarding, and it has given us the opportunity to study the complex field of optimizing structures using numerical algorithms. Working with a variety of computer systems and software has given us a greater understanding of the mentioned technology, and its possibilities.

We will also thank Prof. Evangelos Tyflopoulos, John Floan and Freddy Barstad for their contribution to the thesis. We would not be able to do it without their expertise and good will.

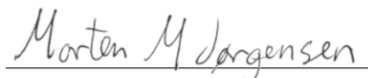
Trondheim, May 20, 2019



Jørgen Petlund



Bartosz Jakubowski



Morten Mosvold Jørgensen

---



---

# Summary

This thesis is written as an introduction to topology optimization with its theory and methods, as well as a comparison between four programs, two open source (TopOpt and OpenLSTO) and two commercially available ones (Abaqus and Ansys). The programs are chosen to represent the three mainly used algorithms there are: SIMP, Optimality Criteria and Level Set Method, all of which will be explained in more detail.

The way the programs capabilities will be tested is by setting up a simple box of 4x4x1 proportions with varying fixtures and loads. The methods and the approach taken will be discussed, and the results analyzed in the light of our findings.

The technology discussed is an early glimpse into what the manufacturing process may look like for nearly all crucial, load bearing parts in the near future. With the recent advances in computer science, topology optimization, and additive manufacturing, this future feels very real, and will require engineers in the mentioned field to adapt.

---

# Sammendrag

Denne oppgaven er skrevet som en introduksjon til topologioptimalisering, dets teori og metoder, og en sammenligning av fire programmer, to Open Source (TopOpt and OpenLSTO) og to kommersielt tilgjengelige (Abaqus and Ansys). Programmene er valg slik at de representerer de tre hovedtypene algoritmer brukt: SIMP, Optimality Criteria og Level Set Method. Alle disse vil være omtalt i mer detalj.

Måten programmene blir testet på, er ved å sette opp en enkel boks med  $4 \times 4 \times 1$  proporsjoner, med varierende fastsetninger og belastninger. Metodene og tilnærmingen som tas vil bli diskutert, og resultatene vil bli analysert i forhold til våre funn.

Teknologien som omtales her, er et tidlig innblikk i hvordan produksjonsprosessen til nesten alle viktige, bærende deler kan se ut i nær fremtid. Med tanke på nyeste fremskritt innen datavitenskap, topologioptimalisering og additivproduksjon, føles denne fremtiden veldig ekte, og vil kreve at ingeniører i det nevnte feltet tilpasser seg.

# Table of Contents

<b>Preface</b>	<b>3</b>
<b>Summary</b>	<b>i</b>
<b>Sammendrag</b>	<b>ii</b>
<b>Table of Contents</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>x</b>
<b>Abbreviations</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Target audience . . . . .	1
1.3 Task . . . . .	2
1.4 Changes . . . . .	3
<b>2 Literature Review</b>	<b>5</b>
2.1 Current state of TO . . . . .	6
2.2 What is TO (general) . . . . .	6
2.2.1 Applications . . . . .	7

---

<b>3</b>	<b>Basic Theory</b>	<b>9</b>
3.1	Topology definition . . . . .	10
3.2	Categories of optimization . . . . .	12
3.3	Methods/ solvers/ algorithms . . . . .	13
3.3.1	SIMP . . . . .	13
3.3.2	Optimality Criteria . . . . .	15
3.3.3	Level set . . . . .	17
3.3.4	Other methods . . . . .	20
<b>4</b>	<b>Method</b>	<b>21</b>
4.1	Input variables . . . . .	22
4.2	Materials and limitations . . . . .	24
4.3	Experiment . . . . .	24
4.4	Process description of Abaqus setup . . . . .	26
4.4.1	Software . . . . .	26
4.4.2	Model, meshing and static analysis . . . . .	26
4.4.3	Topology optimization setup . . . . .	27
4.4.4	Post process visualisation . . . . .	28
4.5	Process description of Ansys setup . . . . .	30
4.5.1	Model, meshing and static analysis . . . . .	30
4.5.2	Topology optimization setup . . . . .	34
4.5.3	Post process visualisation . . . . .	35
4.6	Process description of TopOpt setup . . . . .	36
4.6.1	Software: Large scale topology optimization code using PETSc . . . . .	36
4.6.2	Optimization problem, mesh, and general settings: TopOpt.cc	36
4.7	Process description of OpenLSTO setup . . . . .	40
4.7.1	Software: Open Source Level Set Topology Optimization .	40
4.7.2	Model, meshing and static analysis . . . . .	40
4.7.3	Topology optimization setup . . . . .	44
4.8	Post process visualisation in ParaView . . . . .	46
<b>5</b>	<b>Assessment One</b>	<b>47</b>
5.1	Results for cases C1.1-1.6 . . . . .	48

---

---

5.1.1	Compliance . . . . .	48
5.1.2	Topology . . . . .	49
5.1.3	Computation time . . . . .	55
5.2	Discussion . . . . .	56
<b>6</b>	<b>Assessment Two</b>	<b>59</b>
6.1	Results Cases C2.1-2.5 . . . . .	60
6.1.1	Main findings . . . . .	60
6.1.2	Explanation of categorization of dependency . . . . .	61
6.1.3	Presentation of results . . . . .	61
6.1.4	The special case of OpenLSTO . . . . .	66
<b>7</b>	<b>Assessment Three</b>	<b>67</b>
7.1	Results C3.1-3.5 . . . . .	68
7.1.1	Presentation of results . . . . .	68
<b>8</b>	<b>Conclusion</b>	<b>73</b>
8.1	A subjective evaluation of the programs . . . . .	74
8.2	Further work . . . . .	75
	<b>Bibliography</b>	<b>77</b>
	<b>Appendix</b>	<b>81</b>
8.3	Appendix B1: Abaqus results . . . . .	82
8.3.1	Listing of convergence plots for Abaqus . . . . .	82
8.3.2	Listing of strain, mesh and model plots for Abaqus . . . . .	83
8.4	Appendix B2: Ansys results . . . . .	86
8.4.1	Listing of convergence plots for Ansys . . . . .	86
8.4.2	Listing of model plots for Ansys . . . . .	88
8.5	Appendix B3: TopOpt results . . . . .	93
8.5.1	Listing of convergence plots for TopOpt . . . . .	93
8.5.2	Listing of model plots for TopOpt . . . . .	96
8.6	Appendix B4: OpenLSTO results . . . . .	101
8.6.1	Listing of convergence plots for OpenLSTO . . . . .	101
8.7	Appendix C: . . . . .	107

---

---

8.7.1	Ansys, APDL input file . . . . .	107
-------	----------------------------------	-----



# List of Tables

4.1	Default parameteres . . . . .	23
4.2	Assessment one . . . . .	25
4.3	Ansys material properties . . . . .	32
5.1	Variances between the programs for cases C1.1-1.4 . . . . .	50
5.2	Load case variation C1.1-1.4 Results . . . . .	52
5.3	Average time used per iteration in seconds . . . . .	55
6.1	Mesh discretization dependency results . . . . .	62
6.2	Average time used per iteration in seconds. . . . .	64
6.3	Convergence status, and the iteration at which convergence occurs	65

---

# List of Figures

3.1	Open set visualisation . . . . .	10
3.2	Two figures of the same topological class . . . . .	11
3.3	Different types of optimization . . . . .	12
4.1	Load case variation model setup . . . . .	29
4.2	Element SOLID95 . . . . .	31
5.1	Compliance results for all cases across all programs . . . . .	49
5.2	Load case variation: case 1.5 . . . . .	53
5.3	Load case variation: case 1.5 front view . . . . .	53
5.4	Load case variation: case 1.6 . . . . .	54
5.5	Load case variation: case 1.6 Hollow . . . . .	54
5.6	Total time usage in seconds . . . . .	55
6.1	Compliance cases C2.1-2.5 . . . . .	63
6.2	Total time usage in seconds . . . . .	64
6.3	TopOpt result from the special case 2.5 . . . . .	65
6.4	Delauney smoothing through ParaView of Case 2.1 Ansys model . . . . .	66
6.5	OpenLSTO: new feature developing in case 1.1 . . . . .	66
7.1	Compliance assessment three . . . . .	69
7.2	Total time usage benchmarked against case C2.3 . . . . .	69
7.3	Assessment three: C3.1 . . . . .	70

---

7.4	Assessment three: C3.2 . . . . .	70
7.5	Assessment three: C3.3 . . . . .	71
7.6	Assessment three: C3.4 . . . . .	71
7.7	Assessment three: C3.5 . . . . .	72

---

# Abbreviations

TO	=	Topology Optimization
FE	=	Finite Element
FEM	=	Finite Element Method
ISE	=	Isotropic Solid/ Empty
ASE	=	Anisotropic Solid/ Empty
SIMP	=	Solid Isotropic Material with Penalization
OC	=	Optimality Criteria
BC	=	Boundary Condition
DoF	=	Degree of Freedom

---

# Chapter 1

## Introduction

### 1.1 Background

”Anyone can build a bridge that stands, but it takes an engineer to build a bridge that barely stands”

-Unknown

Our counselor Evangelos Tyfopoulos presented the subject in the fall of 2018. Our bachelor thesis group found each other through our shared enthusiasm for the field of study. The powerful applications of TO appealed to all of us, and we were certain that it’s popularity would increase with the advancements in additive manufacturing and the ever increasing computer power.

### 1.2 Target audience

The thesis is suited for anyone with novice to advanced knowledge about TO, who wants to better understand the practical similarities and differences between the four algorithms chosen in the thesis.

## 1.3 Task

### Sensitivity analysis of Topology optimization

Topology optimization contributes in solving the basic engineering problem by finding the limited used material. Structural optimization reduces the material usage, shortens the design cycle and enhances the product quality. SO can be implemented according to size, shape, and topology. Topology optimization is usually referred to as general shape optimization (Bendsøe 1989). Most of the techniques optimize either the topology or both the size and the shape. On figure.

If TO is integrated into the traditional finite element analysis, the procedure can be divided to 8 steps as it is shown in Figure 1. This figure illustrates the geometry shift of a structure from its original geometry to topology geometry. In the beginning, FEA is implemented. It is possible to be used geometric modifications in order to simplify the initial problem. This stage is challenging to be computerized because it involves applying experience and judgement in a qualitative manner. However, the most crucial step at FEA is the definition of the problem statement and its equivalent mathematical model with all the required parameters (material properties, loads and restraints). The optimum results occur through the discretization (meshing) of the model and with a repetitive convergence method. The topology optimization method offers a new optimized design geometry with a notable mass reduction (or increment) which can be used as a new starting point for the FEA. Finally, the new FEA results validate or evaluate the success of the TO approach (Tyflopoulos *et al.* 2018).

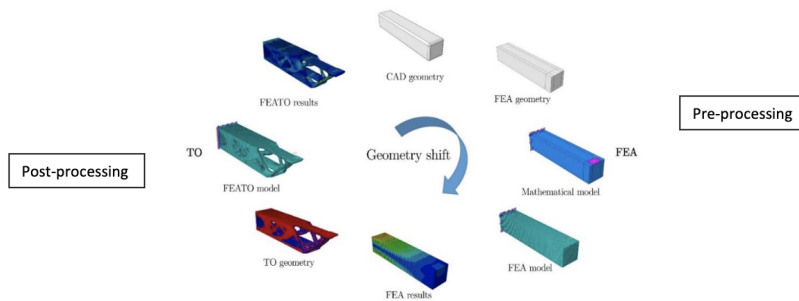


Figure 1: The geometry shift model of a cantilever beam with Abaqus (Tyflopoulos et al. 2018).

This bachelor thesis is focused on the pre-processing of topology optimization methodology. The pre-processing, as it is illustrated on Figure 1, consists of the choice of the CAD and FEA geometry, the definition of the mathematical model and the FEA of this model (CAD geometry-FEA results). A case study of a simple model will be used in order to support the theory and tie the academic text to a realistic application of topology optimization. In addition, a sensitivity analysis will be conducted in order to identify how the designer's choices can affect the topological optimized results.



### Tasks

- Make a literature research about the current state of the art of topology optimization and the implemented approaches.
- Make a research about the most used commercial programs in topology optimization.
- Try to quantify the potential loss by pre-set parameters in topology optimization:
  - Choose the **three** most used commercial software and implement a sensitivity analysis of the optimized results with respect to designer's choices (boundary conditions, constraints and so on).
  - The used software must use different topology optimization algorithms (do not use for example both Autodesk Fusion and Inventor because both of them are using the same topology optimization algorithm).
- Pick a simple model, as a case study, to implement the topology optimization methodology and collect data.

### Comments

- The bachelor thesis will be written in English
- It is possible your data to be used for research purposes
- It is possible to be co-authors in a scientific paper if you want (after you will have handed in your bachelor thesis)
- I can send you papers and theory-stuff

### References

Bendsøe, M.P. (1989) 'Optimal shape design as a material distribution problem', *Structural optimization*, 1(4), 193-202, available: <http://dx.doi.org/10.1007/BF01650949>.

Tyflopoulos, E., Flem, D.T., Steinert, M. and Olsen, A. (2018) '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 DESIGN IN THE ERA OF DIGITALIZATION* The Design Society, 15.

## 1.4 Changes

From the time we got the task from our supervisor, some minor changes has happened. We originally set out to choose 3 commercial available programs. This was later changed to 4 programs. Two open source, and two commercially available

programs. We conducted a literature research, and have done a comprehensive study that consisted of reviewing a great number of scientific articles. We will briefly mention the most relevant articles, however we will not explain in detail all the knowledge that we acquired through the work with the thesis.

# Literature Review

## Chapter Introduction

A literature review was conducted, to further elevate our knowledge about Topology Optimization to an adequate level needed for carrying out the thesis. The review consists primarily of publications, online articles and the knowledge acquired through discussions with other academicians. The sources that proved to be most beneficial, were the following:

**Topology Optimization: Theory, Methods, and Applications**, a book by Martin P. Bendsøe and Ole Sigmund. Very helpful title for beginners in the field, that does a great job of explaining the theory, terminology and methods in TO. [8]

**Structural Topology Optimization: Basic Theory, Methods and Applications**, a master of science thesis written by Steffen Johnsen. It showed to be extremely helpful in the process of introducing us to the advanced field that is Topology Optimization. [12]

**Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics**, a great article by G.I.N. Rozvany, outlining the theory behind the earliest methods of TO as well as the much needed introduction to the core concepts of the field. [17]

**On the trajectories of penalization methods for topology optimization**, an article by M. Stolpe and K. Svanberg that takes into its scope the the comparison of two ISE topologies. [20]

**Evangelos Tyflopoulos**, our guide and advisor who supported this group throughout our project, supplying us with the guidance and resources we needed to complete our bachelor's thesis.

## 2.1 Current state of TO

Topology optimization is a well researched topic. In the past two decades, this field of study has produced an extensive amount of academic articles. The knowledge acquired enables both the designers and the manufacturers, to be able to achieve weight savings never seen before in the optimized structures. Today's optimizing software enables increased strength, while maintaining the weight, or maintaining the strength while reducing the weight. There are several robust commercially available products available that are capable of calculating accurate results, but the programs may require enormous amounts of computational power. The ever increasing amount of power that newer CPU's provide, constantly reduce the computation time needed to perform an optimization cycle. The recent development in additive manufacturing also enables a quicker and cheaper production of intricate parts generated by TO, which makes the entire process even more feasible from a manufacturing standpoint.

## 2.2 What is TO (general)

TO is a family of algorithms with the goal of producing the most optimal material layout from a predefined set of parameters. The algorithms are given the design space the product has to be created within, loads applied, and physical constraints. In comparison to the standard CAD (Computer Aided Design) programs, TO algorithms do not rely on reuse of known shapes, as the program finds the most

effective shape strictly by considering a set of parameters. The list of programs capable of TO is long, and many of those utilize different mathematical approaches for their own optimization process.

### 2.2.1 Applications

TO has shown to be beneficial in several industries, and its use frequently discovers new applications. The algorithms have no human constraints, and all the freedom within the design space. This produces complex shapes that would be impossible for an engineer to imagine, and take an unfeasible amount of time to achieve by trial and error. Typical industries where mass is critical can benefit immensely from the potential weight savings this offers. One great example of this is the aerospace industry, as it can improve the fuel economy by replacing a wide variety of its components with lighter TO parts, that meet the same standards as conventionally designed parts. Tomlin and Meyer (2013) [22]

The first results obtained from TO software are a mere visualisation, and need to be revised, as the optimal solution proposed might still not meet the set criteria. Manufacturing of topology optimized parts also require a refinement, as the introduced shapes can be quite intricate, lack the needed smoothness, and be nearly impossible to mass produce with conventional production equipment if left unaltered.

We strongly believe that, as stated above, the ever faster development of Additive Manufacturing (AM) is the way to truly show the possibilities of TO. The big benefit of AM is the added degree of freedom in designing a shape, given infinitely better accessibility of enclosed areas, and the challenges involving finding an intricate tool path no longer appearing as a problem. On the other hand, the technology is currently more costly compared to any conventional mass production methods. It is mainly applicable for cases where the benefits of weight reduction hugely outweigh the manufacturing costs. The increased cost of manufacturing is however steadily on the decline as the technological advancements continue to lower the costs.



# Chapter 3

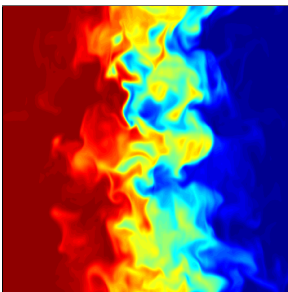
## Basic Theory

### **Chapter Introduction**

Topology optimization is aimed at finding the best material distribution within a given design space. TO software takes advantage of various mathematical methods to calculate the stresses in the individual finite elements of a three-dimensional model. All of the model's volume is considered in weight to strength/ stiffness contribution ratio, which given an input of a desirable volume retention, makes it possible to remove material in areas of little to no stress concentrations.

### 3.1 Topology definition

Defining topology for the sake of engineering purposes is somewhat elusive. The term is adherently abstract, yet it in topology optimization we often reference it as a physical measurement or property of a geometric representation. Now, there is two mathematical definitions of topology. The following definition of *topology* is taken from *Non-Hausdorff Topology and Domain Theory* by Goubault-Larrecq (2013) [9].



**Figure 3.1:** Flow visualisation that illustrates open sets (Courtesy of Mix (2019) [2])

*Let  $X$  be a set. A topology on  $X$  is a collection of subsets of  $X$ , called the opens of the topology, such that:*

- every union of opens is open;*
- every finite intersection of opens is open.*

Where a *set* is a collection of objects and *open* means that it geometrically has no boundary. An object is in this regard a placeholder which can contain any geometric definition. As for the set having no boundary, it does not constitute to it being infinitely large. It simply states that we can not specify *exactly* where it ends.

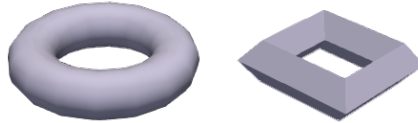
This is further illustrated by figure 3.1. Think of all the contents within the frame as a set and the red and blue color fields as two subsets. The subsets are clearly located within two separate areas, yet we cannot precisely define the transition between them. Since the boundary is not exact the sets is considered *open*. This is the basic principle behind open sets. Another common definition of topology:

*The study of geometrical properties and spatial relations unaffected by the continuous change of shape or size of figures [4].*

Where the geometrical properties are all those properties than can be derived from the geometry of a solid body or particle [1]. *Spatial relations refer to how objects are located in space in relation to a reference object [3].*

When we reference topology in this study it does not follow these conventions. That would mean that the two shapes in Figure 3.2 are the same topology. We use





**Figure 3.2:** Two figures of the same topological class. They are topologically equivalent by homeomorphisms.

the term to describe a fixed set of geometrical properties that are *not* unaffected by the continuous change of shape or size.

In this study there was a need to be able to reference a selection of subsets of the topology. It could be done by calling it *part* of the topology, but term *part* is in engineering used to describe a component of an assembly. The term given is a *topological feature*. A topological feature is a selection of subsets that meets the conditions:

- a. All subsets within the selection must be connected.
- b. The selection of subsets can not represent all subsets of the topology.

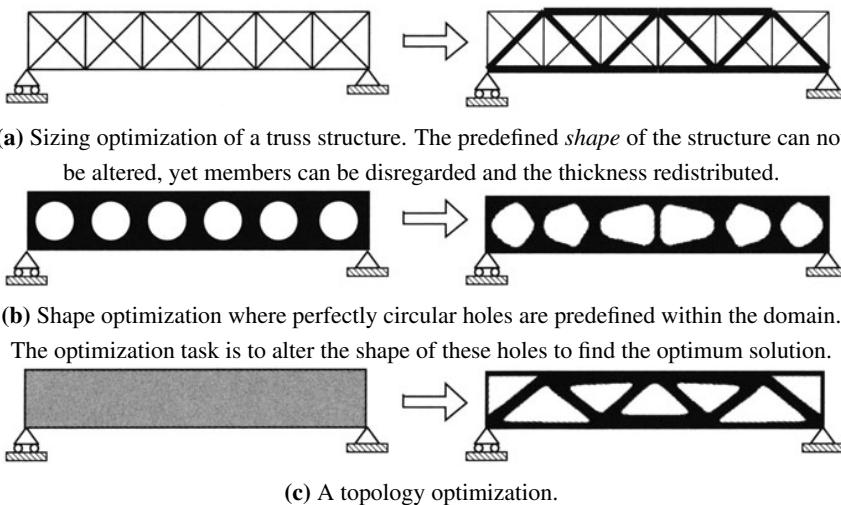
The topological complexity of a model can then be defined as: A topology has a higher level of complexity when it contains higher degree of topological features.

Other distinctions of importance is the usage of shape, structure and body. Shape is used to describe a layout. Structure is the representation of connection between a set of points, i.e. a truss is purely structural. A body is a closed solid with a given topology, and is not restricted by shape. Meaning that a body can be both deformed and scaled.

## 3.2 Categories of optimization

Assuming we are working with an isotropic, linearly elastic material, there are three ways of distributing the material in a structure in order to find the optimal layout that satisfies the given criteria. The three types of problems are sizing, shape and topology optimization problems.

The typical sizing problem would consider the optimal thickness distribution of a linearly elastic plate, or the optimal member areas in a truss structure. Resulting thickness would satisfy the equilibrium of design constraints, loads, and other parameters as deflection or weight. The difference between sizing and shape optimization is that in sizing optimization process the domain of the design model and state variables are set prior to, and fixed throughout the process, as opposed to the shape optimization problems, where the goal is to find the optimum shape of the domain, making it the design variable in stead. Topology optimization on the other hand involves the determination of features such as the number of , location and shape of the holes, and the connectivity of the domain. Bendsøe and Sigmund (2004) [8]



**Figure 3.3:** Different types of optimization

### 3.3 Methods/ solvers/ algorithms

Given the wide range of methods available for the execution of TO processes, we have chosen a set of software that utilizes three general, most popular and effective approaches of calculating the optimal solution to a given topology optimization problem. The methods chosen are SIMP (Solid Isotropic Material with Penalization), Optimality Criteria and Level Set Method. The methods chosen use different algorithm's for calculating which areas of the topology to keep, which to remove, and different methods for solving the problem overall. The goal of all these remains the same, and it is to maximise strength and stiffness, while only using a specific percentage of the given volume.

The next section will consist of a theoretical explanation for the three TO methods chosen. The content of this section however is heavily inspired by other, greatly written articles, describing the theory in even more detail. By no means did we calculate any of the equations provided, nor did we invent the theory behind the algorithms. The relevant articles used will be quoted, and we strongly recommend to read the mentioned work for all those interested in theory, as we merely scratched the surface of the topic.

#### 3.3.1 SIMP

The SIMP method is one of the better described approaches to topology optimization problems. The basic idea of this method is the use of a fictitious isotropic material, whose elasticity tensor is assumed to be a function of penalized density of the material. It is showing to be advantageous in a lot of ways that will be discussed in this section. As mentioned above, the term SIMP stands for Solid Isotropic Material with Penalization, and as the name implies, it considers the finite elements in a given model to be either solid or empty, and the material to be isotropic (homogeneous). This method only relies on one penalization criteria, which requires less computational power to solve. Biggest advantages ( Rozvany (2001) [17] ) of SIMP method for optimizing Isotropic Solid (IS)/ Isotropic Solid/ Empty (ISE) topologies are the following:

(a) Computational efficiency in terms of storage usage and CPU solving time, since there is only one free variable per element.

(b) Robustness in the sense that SIMP can be readily used for any combination of design constraints.

(c) Free adjustment of the penalization criteria

(d) Conceptual simplicity, since the algorithm does not require derivations involving higher mathematics.

(e) Since the p-value is increased progressively, we can start SIMP with a solution for  $p = 1$  for which some problems (e.g. compliance) are convex and the solution a global optimum. The subsequent gradual incrementation of the p-value is not likely to move the solution too far from the global optimum, but this is only an “experimental” finding at this stage.

(f) SIMP does not require homogenization of the micro-structure

A disadvantage of SIMP is that the solution depends on the degree of penalization (p-value) and it does not necessarily converge to the optimal solution. This problem is however also relevant to all the other ISE topologies. Stolpe and Svanberg (2001) [20]

Programs using the SIMP method are written with code, but at the fundamental level, they still consist of mathematical equations. This is the simplest way the calculations behind the SIMP method can be described:

$$E_{ijkl}(x) = \rho(x)^p E_{ijkl}^0, \quad p > 1$$

$$\int_{\Omega} \rho(x) d\Omega \leq V; \quad 0 \leq \rho(x) \leq 1, \quad x \in \Omega$$

The calculation process is further explained:

$$a(u, v) = \int E_{ijkl}(x) \epsilon_{ij}(u) \epsilon_{kl}(v) d\Omega$$

$$\epsilon_{ij}(u) = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

$$l(u) = \int_{\Omega} f u \, d\Omega + \int_{\Gamma_T} t u \, ds$$

$$\min(u \in U, E) = l(u)$$

$$a_E(u, v) = l(v)$$

$$a_E(u, v) = l(v), \text{ for } (v \in U) \text{ and } E \in E_{ad}$$

- $U$  Kinematically admissible displacement fields
- $f$  Body forces
- $t$  Boundary tractions
- $E_{ad}$  Admissible stiffness tensors for design problems

The various possible definitions of  $E_{ad}$  is the subject of different optimization algorithms. When working with a FE (Finite Element)-analysis, the problem must be stated in a discrete manner. Assuming a constant  $E$  for each element, the discrete form of can be expressed as:

$$\min(u, E_e) \quad f^T u, \text{ so that}$$

$$K(E_e)u = f$$

$$E_e \in E_{ad}$$

$$K = \sum K_e(E_e) \quad , \text{ summing } i = 1..N$$

- $u$  Displacement vector
- $f$  Load vector
- $K$  Global element stiffness matrix
- $K_e$  Stiffness matrix for element  $e$ , dep. on the stiffness  $E_e$  in the element

<sup>1</sup>Johnsen (2013) [12]

### 3.3.2 Optimality Criteria

Optimality criteria methods are based on a different way of thinking from those applied in the methods of mathematical programming (MP). Most of the MP methods

focus on getting information from conditions around the current design point in design space in order to find the answer those two questions: in what direction, and how far to go, to reduce the value of the objective function directly in a best way possible. This process is repeated until we arrive at a convergence value, within a given tolerance. Optimality criteria methods on the other hand, derive or state conditions which describe the optimum design, then find or alter the design to satisfy those conditions, while indirectly optimizing the structure itself. This way, the optimality criteria approach finds a result close to the optimal design very quickly. The procedure can be divided into four steps:

Step 1. derives the optimality criteria equations

Step 2. is the iteration procedure for the design variables.

Step 3. is the iteration procedure for the Lagrange multipliers.

Step 4. is the computer program implementation.

The optimality criteria methods are indirect methods of optimization and unlike mathematical programming methods which directly optimize the objective function, optimality criteria methods attempt to satisfy a set of criteria related to the behaviour of the structure. These criteria are derived either intuitively or rigorously. "Fully stressed design" and "simultaneous failure mode design" are examples of the optimality criteria methods.

The problem of topology optimization under multiple constraints can be stated as follows:

$$\min \left( \phi = \int_{\Omega} \rho(\mu) d\Omega \right),$$

such that

$$\int_{\Omega} \mathbf{u} \nabla : \mathbf{E}(\mu) : \mathbf{v} \nabla d\Omega = \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{v} d\Gamma,$$

$$\int_{\Gamma_t} h_{\alpha}(\mathbf{u}) d\Gamma - \bar{h}_{\alpha} \leq 0, \quad \alpha = 1, \dots, m,$$

$$\mu_{min} \leq \mu \leq \mu_{max},$$

where  $\mu$  denotes a design variable with the lower bound  $\mu_{min}$  and the upper

bound  $\mu_{max}$ ,  $\rho(\mu)$  the local density of the material,  $\mathbf{E}(\mu)$  the material stiffness,  $\Omega$  the design domain,  $\Gamma_t$  the traction boundary,  $\mathbf{t}$  the traction acting on the structure,  $\mathbf{u}$  and  $\mathbf{v}$  the actual and the kinematically admissible displacements of the structure,  $h_\alpha$  a pointwise function and  $\bar{h}_\alpha$  the imposed value for the  $\alpha$ th constraint, and  $m$  the total number of the displacement constraints. Yin and Yang (2019) [23]

### 3.3.3 Level set

With this method, the optimized structure is implicitly represented by a moving boundary, embedded in a higher dimensionality scalar function (the level set function). While the shape and topology of the structure may experience major changes, the level set function remains simple in its topology. This way, by a direct and efficient computation in the embedding space, the movement of the design boundaries under a relevant speed function can be tracked to capture changes in the shape and topology of the structure. The level set models may also be referred to as implicit moving boundary (IMB) models and they can easily represent complex boundaries that can form holes, split into multiple pieces, or merge with others to form a single one. Based on the concept of propagation of the level set surface, the design changes are carried out as a mathematical programming for the optimization problem.

In level set-based structural optimization methods, complex shape and topological changes can be handled and the obtained optimal structures are free from gray scales, since the structural boundaries are represented as the iso-surface of the level set function. Although these relatively new structural optimization methods overcome the problems of checkerboard patterns and gray scales, mesh dependencies have yet to be eliminated. Such methods implicitly represent target structural configurations using the iso-surface of the level set function, which is a scalar function, and the outlines of target structures are changed by updating the level set function during the optimization process.

The velocities required for the level set update are obtained by solving an op-

timization problem. A generic optimization problem can be formulated using the position of the structural boundary as the design variable:

$$\begin{aligned} & \underset{\Omega}{\text{minimize}} && f(\Omega) \\ & \text{subject to} && g_i(\Omega) \leq 0 \end{aligned}$$

where  $f(\Omega)$  is the objective function and  $g_i$  is the  $i^{\text{th}}$  inequality constraint function. The objective and constraint functions are linearized about the design variables at each  $k^{\text{th}}$  iteration using first-order Taylor expansion:

$$\begin{aligned} & \underset{\Delta\Omega^k}{\text{minimize}} && \frac{\partial f}{\partial\Omega^k} \cdot \Delta\Omega^k \\ & \text{subject to} && \frac{\partial f}{\partial\Omega^k} \cdot \Delta\Omega^k - g_i^{-k} \end{aligned}$$

where  $\Delta\Omega^k$  is the update for the design domain  $\Omega$  and  $g_i^{-k}$  is the change in the  $i^{\text{th}}$  constraint at iteration  $k$ . In the level-set description of the boundary, shape derivatives provide information about how a function changes over time with respect to a movement of the boundary point. They usually take the form of boundary integrals. In this case,

$$\begin{aligned} \frac{\partial f}{\partial\Omega} \cdot \Delta\Omega &= \Delta t \int_{\Gamma} s_f V_n d\Gamma, \\ \frac{\partial g_i}{\partial\Omega} \cdot \Delta\Omega &= \Delta t \int_{\Gamma} s_{g_i} V_n d\Gamma, \end{aligned}$$

where  $s_f$  and  $s_{g_i}$  are the shape sensitivity functions for the objective and the  $i^{\text{th}}$  constraints. Discretizing the boundary at  $nb$  points, one can rewrite:

$$\begin{aligned} \frac{\partial f}{\partial\Omega} \cdot \Delta\Omega &\approx \sum_{j=1}^{nb} \Delta t V_{nj} s_{f,j} l_j = \mathbf{C}_j \cdot \mathbf{V}_n \Delta t, \\ \frac{\partial g_i}{\partial\Omega} \cdot \Delta\Omega &\approx \sum_{j=1}^{nb} \Delta t V_{nj} s_{g_i,j} l_j = \mathbf{C}_{g_i} \cdot \mathbf{V}_n \Delta t, \end{aligned}$$

where  $l_j$  is the discrete length of the boundary around the boundary point  $j$ ,



$C_j$  and  $C_{g_i}$  are vectors containing integral coefficients and  $V_n$  is the vector of normal velocities. For a constrained problem, one can write:

$$V_n \Delta t = \alpha d,$$

where  $d$  is the search direction for the boundary update and  $\alpha > 0$  is the actual distance of the boundary movement. Then, the optimization formulation to obtain the optimal boundary velocities can be written as:

$$\begin{aligned} & \underset{\alpha^k, \lambda^k}{\text{minimize}} && \Delta t C_f^k \cdot V_n^k(\alpha^k, \lambda^k) \\ & \text{subject to} && \Delta t C_i^k \cdot V_n^k(\alpha^k, \lambda^k) \leq -g_i^{-k} \\ & && V_{n,min}^k \leq V_n^k \leq \bar{V}_{n,max}^k \end{aligned}$$

where  $\lambda$  are Lagrange multipliers for each constraint function. This optimization problem is solved at every iteration  $k$ .

The conjugate gradient method (`StationaryStudy::Solve_With_CG`) is used to solve the system of linear equations defined as  $[K] \{u\} = \{f\}$ , where  $[K]$  is the reduced global stiffness matrix,  $\{u\}$  is the reduced global displacement vector and  $\{f\}$  is the reduced global load vector. The algorithm is described as follows:

- set initial displacement:  $\{u_0\} = 0$
- set the initial residual:  $\{r_0\} = \{f\} - [K] \{u_0\}$
- set the conjugate gradient:  $\{p_0\} = \{r_0\}$
- set the iteration counter  $k = 0$
- while  $\|r_k\| > tol$ , do

$$\alpha_k = \frac{r_k^T r_k}{p_k^T K p_k}$$

$$u_{k+1} = u_k + \alpha_k p_k$$

$$r_{k+1} = r_k + \alpha_k \mathbf{K} p_k$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k$$

$$k = k + 1$$

The most time-consuming part of the conjugate gradient algorithm is the matrix vector multiplication.

<sup>2</sup>M2D (2018) [14]

### 3.3.4 Other methods

As concise as the list of TO algorithms we chose may sound like, there is actually a plethora of other methods, that either use a new approach at calculating the optimum design, or aim at slightly improving the existing process by approaching it from a new angle. The list of methods include, but is not limited to: RAMP, OMP, NOM, etc.

# Chapter 4

## Method

### **Chapter Introduction**

This chapter describes the steps taken to conduct a sensitivity analysis on a topology optimization problem with the objective of minimizing compliance. Starting out with determining which input parameters to consider for analysis, and subsequently choosing which to disregard. The intent of section 4.4-4.7 is to provide an in depth description on how the model implementation, simulation setup and analysis is carried out in each program. Variations and necessary modifications of the input parameters between the programs is explained, as well as the reasoning behind it. These sections can also be treated as supplementary user guides on how to set up a topology optimization task in the given software environment. Section 4.3 describes in detail the specifications of the experiment, presenting the exact values used when solving the minimum compliance problem for the different parameter sets.

## 4.1 Input variables

*Sensitivity analysis (SA) is the study of how to apportion the variation in the model output, qualitatively or quantitatively to variations in the model inputs.*

– Saltelli, Tarantola, and Campolongo (2000) [18]

As stated above the premise of a sensitivity analysis is to survey the output variations instigated by input variations. Or rather, to ascertain the correlation between input and output. In this study the different *input variables* are referred to as input parameters or, simply, parameters, as to not confuse it with the term *design variables*.

A topology optimization task includes a multitude of input parameters by default and in correspondence to the specific optimization problem, which in this case was that of finding minimum compliance with a static structural load case under a volume constraint. The input parameters are sorted according to which step of the setup they belong to. Be it part of; the static structural analysis, the topology optimization or the designation of computational resources. As the focus of this study is evaluating sensitivity in the pre-processing phase of topology optimization, the investigated parameters conform to the static structural setup. Table 4.1 show the default values.

Parameters	Program values			
	Abaqus	Ansys	TopOpt	OpenLSTO
<b>Static structural</b>				
Model	Hyperrectangle <b>4 x 1 x 4 domain</b> , from origin: $[X, Y, Z]=[200, 50, 200]$			
Element type	Hexahedral 8-node	Hexahedral 20-node	Hexahedral 8-node	Hexahedral 8-node
Mesh discretization	Discretized by $100 \times 25 \times 100 = 25000$ elements, where $X_{Element} = Y_{Element} = Z_{Element}$			
Young's modulus $E$	200e3	200e3		200e3
Poisson's ratio, $\nu$	0.3	0.3	0.3	0.3
Boundary constraint	<i>As specified per case</i>			
Load case(s)	<i>As specified per case</i>			
<b>Topological optimization</b>				
Optimization domain	Hyperrectangle <b>4 x 1 x 4 domain</b> (Same as model domain).			
Convergence criteria		*Relative compliance and density $[\frac{x_{n-1}}{x_n}] < 0.0005$	**Design change $\ x_k - x_{k-1}\ _\infty < 0.001$	***Compliance $\Delta C^k < 0.001$
Volume constraint (%)	15	15	15	15
Max # of iterations	30	40	80	80
Move limit	0.25	N/A	0.25	0.25
Penalty factor	3	N/A	3	N/A
<b>Designated computational resources</b>				
# of Nodes	N/A	1	1	1
# of Cores	****6	20	<i>(Two 10-core Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz)</i>	
Memory requirement	16GB	90GB	35GB	60GB

\*See the Ansys theory reference [7].

\*\*Design change calculated from  $x_k$  at iteration  $k$ . Calculations of the  $x_k$  vector can be found in the MMA class (Method of Moving Asymptotes, MMA.cc) within the TopOpt module [5].

\*\*\*The convergence criterion for OpenLSTO derives from  $\Delta C^k = \max(|C^k - C^m|/C^k)$ ,  $m \in [k-5, k-1]$  where  $C^k$  is the compliance at iteration  $k$ , and is calculated when the volume constraint is satisfied [15].

\*\*\*\*Intel Core i9 @ 2.90GHz 5 cores, 10 threads

**Table 4.1:** Overview of the chosen default parameter values.

## 4.2 Materials and limitations

The HPC cluster *IDUN* was used for conducting this study. Due to Abaqus being remarkably harder to use non-interactively, it was concluded it will be in our best interest to run the simulations with the user interface on a Macbook Pro instead. Because of this Abaqus is not used for all the cases.

## 4.3 Experiment

Part of the sensitivity analysis was done by dividing it into three assessments. Where each assessment has a chosen input parameter or set of parameters designated as variables that change between the cases, and the remaining input parameters are set as constants.

For further reference in this paper the cases under each of the three assessments are given the label: *Case* or abbreviated to  $C < Assessment\ number > . < Case\ number >$ .

In the description of how the different programs are set up (sections 4.4, 4.5, 4.6, 4.7) the first case is used as the basis. When variations of the setup are mentioned the pointer  $\blacklozenge$  is used together with the case label(s) to indicate that it was done this way for the referenced case(s).

### Assessment one: Load case variation

In this assessment the variables are the load setup and boundary condition definitions. Six cases are evaluated, where the first four are run across all programs, and use of Abaqus is omitted from the last two. The first four cases differentiate between having one or two loads and one or two boundary conditions at four fixed surfaces. The force value is set to  $F_z = -6000$  (meaning in the  $Z$ -direction) when one load is applied, and halved (two  $-3000$  loads) when two loads are present, effectively giving a sum total force value of  $F = |6000|$  for all cases. As for the fifth case the force value setup is changed. The load vector is now composed of forces in two directions, i.e  $F_z = -6000$  and  $F_y = 1000$ . In case six the surface area for

the load and BC was increased, and the force value was altered as well. The load is made up of a force  $F_x = 6000$  and  $F_z = -6000$ . The setup of the surface areas for all cases are shown in figure 4.1 on page 29. Table 4.2 below shows us how we can do a comparative evaluation of the different cases with regards to one another.

<b>Evaluation Matrix</b>		
	One load case	Two load cases
Two BC	Case1.1	Case1.4
One BC	Case1.2	Case1.3
<i>*The load size and BC area are different for the cases below</i>		
Two BC	Case1.5	
One BC	Case1.6	

**Table 4.2:** As we can see

### Assessment two: Mesh discretization dependency

Using the same setup case C1.1 we changed the level of discretization. Four new cases are introduced with a mesh discretization of:

Case 2.1:  $20^2 \times 5 = 2000$  *Elements*

Case 2.2:  $40^2 \times 10 = 16000$  *Elements*

Case 2.3:  $64^2 \times 16 = 65536$  *Elements*

Case 2.4:  $80^2 \times 20 = 128000$  *Elements*

These can be all be evaluated against each other and against case C1.1 which has a discretization level of  $100^2 \times 25 = 250000$  *Elements*. The use of Abaqus was omitted from case C2.2 and C2.4. A special case C2.5 with a discretization of  $128^2 \times 32 = 524288$  *Elements* was conducted solely in TopOpt. It was also started in OpenLSTO, but the time required per iteration to solve the problem made it unfeasible to see it through with our setup.

### Assessment three: Domain change

This assessment was conducted with the intent of evaluating if a global optimum was obtained in the previous cases and look at how symmetry reduction can be applied to TO problems. Five cases are set up with the domain size doubled in the  $Y$ -direction, equivalent to a  $4 \times 2 \times 4$  domain. The use of Abaqus was omitted from all cases. Seeing the result from assessment two, the mesh discretization

is lowered to  $64^2 \times 16$ . This gives faster computation time while still producing viable results, and as .

## **4.4 Process description of Abaqus setup**

Abaqus is one of two commercial programs we use, we did all the setup through the graphical user interface of the software. It uses the SIMP method for performing topology optimization.

### **4.4.1 Software**

The software used is the 2017 version of Abaqus with Tosca, developed by Dassault System. Often referred to as ATOM (Abaqus Topology Optimization Module). The Abaqus setup is not a detailed guide of every step. It serves as a brief explanation of the steps taken to perform the TO.

### **4.4.2 Model, meshing and static analysis**

Since running TO jobs can take a lot of computer resources, it is a good idea to check the analysis with a static study before running the TO. In Abaqus the steps for setting up a static study consist of creating a model geometry, assigning material properties, creating a static analysis step, defining BC and loads, generating a mesh, and finally, solving the problem.

#### **Model implementation**

There are several ways of creating the model in Abaqus. We chose to draw a square, scale it to 200x200, then extrude it to 50. It is also easy to import model files from other programs as long as the format is supported.



### **Loads and constraints**

Loads and constraints are added through the same module. The loads are evenly distributed on one or two 50x50 squares. The full load of 6000N is either all on one square, or split on two squares made up of 2x3000N loads. The constraints are also one or two 50x50 squares. Graphic plots will further explain the loads and constraints later in the chapter

### **Material properties**

For Abaqus the only needed material properties for conducting the TO is the Young's modulus and Poisson's ratio. These are added through the material properties manager, and submitted to the wanted geometry. We chose to assign the same material for the whole model, but there is also the option to create a composite material layup. The value used for the Young's modulus is 200 000. The value used for the Poisson's ratio is 0.3.

### **Meshing**

Operations for meshing is conducted by selecting the model and creating a mesh grid of squares with an element size  $X_{Element} = Y_{Element} = 2$ . We left the rest of the parameters to default settings. The mesh type used is hex elements. The mesh is then swept through the volume. We chose to use the automatic mesh generator.

### **Static analysis**

As mentioned briefly it is sound engineering practice to conduct a quick static study to check if the analysis is set up correctly. This is the normally the last step before setting up the TO. In Abaqus this is done by submitting a job through the job module. This normally takes a fraction of the time a TO would take. The loads, mesh and material properties is the same in both the TO and the static study, so there is no need for a separate setup later for the TO.

## **4.4.3 Topology optimization setup**

To set up the TO, at least four steps have to be taken. Step one is creating the task in the Optimization Task Manager. Step two is to create the design responses. In

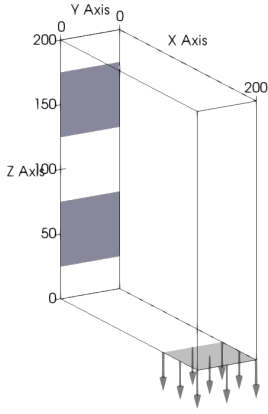
our TO the design response is to minimize the strain energy, and limit the volume. Step three is to choose the objective function. In this case it is the strain energy. We will choose to minimize it. Step four is to set the constraint to 15 percent of the original volume. Now the TO is ready for submission through the job module.

#### **4.4.4 Post process visualisation**

To view the TO results, the job needs to be processed with the combine function in the job manager. When it is combined the results can be viewed with the visualization module.

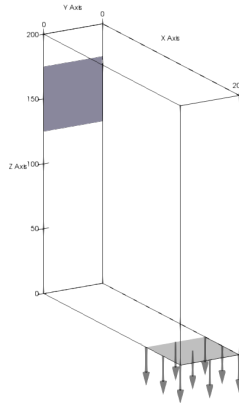
Load case variation  
Configuration figures for Assessment One

Case 1.1



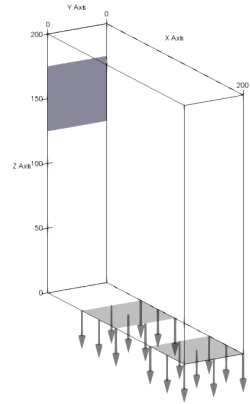
**(a)** Case 1.1: One load  $[150 < X < 200][Z = 0.0]$  and two BC  $[X = 0.0][25 < Z < 75]$  &  $[125 < Z < 175]$  present.

Case 1.2



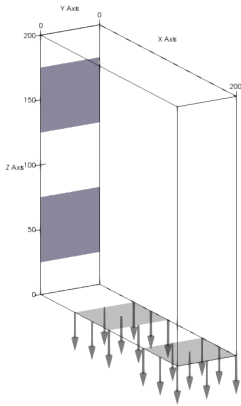
**(b)** Case 1.2: One load  $[150 < X < 200][Z = 0.0]$  and one BC  $[X = 0.0][125 < Z < 175]$  present.

Case 1.3



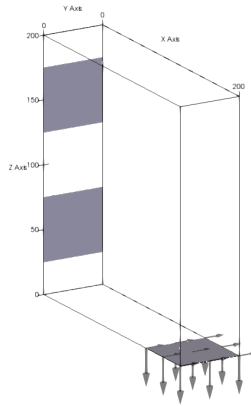
**(c)** Case 1.3: Two loads  $[150 < X < 200]$  &  $[50 < X < 100][Z = 0.0]$  and one BC  $[X = 0.0][125 < Z < 175]$  present.

Case 1.4



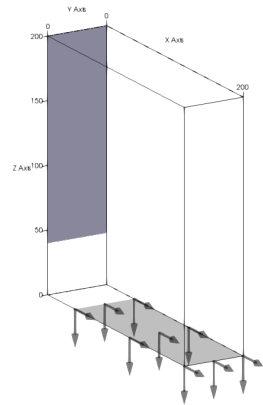
**(d)** Case 1.4: Two loads  $[150 < X < 200]$  &  $[50 < X < 100][Z = 0.0]$  and two BC  $[X = 0.0][25 < Z < 75]$  &  $[125 < Z < 175]$  present.

Case 1.5



**(e)** Case 1.5: One load  $[150 < X < 200][Z = 0.0]$  and two BC  $[X = 0.0][25 < Z < 75]$  &  $[125 < Z < 175]$  present.

Case 1.6



**(f)** Case 1.6: One load  $[40 < X < 200][Z = 0.0]$  and one BC  $[X = 0.0][40 < Z < 200]$  present.

**Figure 4.1:**  
 $[0.0 < Y < 50]$  for all cases. See Appendix A for further details.

## 4.5 Process description of Ansys setup

Setting up Ansys for a topological optimization run using the Mechanical APDL module is done by first creating a static analysis and then initiating an optimization process which uses the static structural input.

The full APDL input file, along with a selection of useful sites and guides for understanding the *Ansys Parametric Design Language*, can be found in Appendix C: under section 8.7.1.

Ansys Mechanical APDL version 19.3 is used for this study. The Optimality Criteria is chosen to solve the TO problem.

### 4.5.1 Model, meshing and static analysis

In Ansys Mechanical APDL the steps for setting up a static study consist of creating a model geometry, generating a mesh, defining BC and loads, and finally, solving the problem. Besides this, specifying material properties and units as well as issuing an equation solver solver is necessary for obtaining viable results.

#### Model implementation

The model is created through a series of steps. First, four *keypoints* are initiated in the *XY*-plane at the following coordinates:

$$K_1(0.0, 0.0) \quad K_2(200, 0.0) \quad K_3(200, 200) \quad K_4(200, 0.0)$$

Second, *lines* are created between the *keypoints* making a  $200 \times 200$  square:

$$K_1 \longrightarrow K_2 \quad K_2 \longrightarrow K_3 \quad K_3 \longrightarrow K_4 \quad K_4 \longrightarrow K_1$$

The third step is making a *surface* within the space confined by the *lines*, and the last step is extruding a *volume* from the *surface*.

<sup>1</sup> `ASEL,r,area,,1,1,`

<sup>2</sup> `VEXT,1,1,,0,0,zval`

The *Area Select* command *ASEL* is used to select the desired surface, which in this case is the only surface available; surface number one. *VEXT* is then issued. This *Volume extrusion* command selects surface number one from the all-

ready selected surfaces and extrudes this surface in the  $Z$ -direction by an amount given by the custom APDL parameter,  $zval$ . Since the domain of our model is supposed to be  $Depth \times 1/4 = Length = Height$  the parameter  $zval$  is given the value  $50 (= 200 \div 4)$ . We now have a volume of the *hyperrectangle* class by a  $200 \times 200 \times 50$  domain.

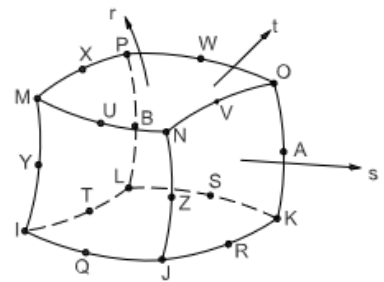
*Note:* Due to limitations by the *VEXT* command the orientation of the Ansys domain is not  $4 \times 1 \times 4$ , but  $4 \times 4 \times 1$ . All BC and loads are still equal to the rest of the programs relative to domain orientation.

## Meshing

Operations for meshing is conducted by choosing the surface and creating a mesh grid of squares with an element size  $X_{Element} = Y_{Element} = 2$ . The mesh is then swept through the volume.

- 1 *ET*,1,95,
- 2 *VSWEEP*,1,1,

First, all volumes are selected. Then the element class and type is determined by the *ET* command. The *VSWEEP* command then creates elements throughout the volume from the surface mesh. The only hexahedral element type supported by the topological optimization in Ansys Mechanical APDL is the *SOLID95*, which is given as a *3-D 20-node structural solid* [6].



**Figure 4.2:** The 20-node SOLID95 element has nodes located at every external corner and at the midpoint of each line connecting the box. (Courtesy of ANS (2009) [6])

## Material properties

Material properties are added by the *MP* command and a *Label*, which represent the exact type of property. The following code snippet shows how the values are implemented and table 4.3 lists the connection between the *Labels*, properties and assigned values.

- 1 *MP*,DENS,1,7.85e-09, ! tonne mm<sup>-3</sup>
- 2 *MP*,ALPX,1,1.2e-05, ! C<sup>-1</sup>
- 3 *MP*,C,1,434000000, ! mm<sup>2</sup> s<sup>-2</sup> C<sup>-1</sup>
- 4 *MP*,KXX,1,60.5, ! tonne mm s<sup>-3</sup> C<sup>-1</sup>
- 5 *MP*,RSVX,1,0.00017, ! ohm mm
- 6 *MP*,EX,1,200000, ! tonne s<sup>-2</sup> mm<sup>-1</sup>, Elastic moduli

```

7 MP,NUXY,1,0.3, ! Minor Poisson's ratio
8 MP,MURX,1,10000,

```

### *Material properties for Ansys Mechanical APDL*

<i>Property</i>	<i>Label</i>	<i>Value</i>	<i>Unit</i>
Density $\rho$	<i>DENS</i>	7.85	<i>Kg/Litre</i>
Coefficient for thermal expansion	<i>ALPX</i>	$1.2e - 5$	$C^{-1}$
Specific heat	<i>C</i>	434	<i>J/Kg°C</i>
Thermal conductivity	<i>KXX</i>	60.5	$W \cdot m^{-1} \cdot K^{-1}$
specific electrical resistance	<i>RSVX</i>	$1.7e - 7$	$\Omega \cdot m$
<b>Young's modulus <math>E</math></b>	<i>EX</i>	200000	<i>MPA</i>
<b>Poisson's ratio <math>\nu</math></b>	<i>NUXY</i>	0.3	
Magnetic relative permeability	<i>MURX</i>	10000	<i>H/m</i>

**Table 4.3:** In topology optimization material properties of particular interest is the Young's modulus  $E$  and Poisson's ratio  $\nu$ .

### **Boundary condition**

BC are added by selecting the nodes and then issuing the  $D$  command which assigns zero DoF for the selected dimension.

```

1 ALLSEL
2 NSEL,r,loc,x,125,175,
3 NSEL,r,loc,y,-0.01,0.01
4 NSEL,r,loc,z,0,zval
5 D,all,all

```

Making sure to not select within a previous selection, we use the *ALLSEL* command for selecting all entities of all types. We then specify the nodes to be selected using the *Node select* command *NSEL*. First the range in the  $X$ -direction is selected, then  $Y$ -direction, and at last the  $Z$ -direction thus, creating a volume,  $V$ , in which the selected nodes has to lay within:

$$V = X \in [125, 175], Y \in [-0.01, 0.01], Z \in [0.0, zval] \quad zval = 50$$

This selection, *NSEL,r*, is actually a *re-selection* of the previous selection. Distinction between selecting and re-selecting is an important consideration in the *APDL*-environment, and must be consistent and deliberate. If the *Node select*

command does not select within the previous selection, the currently selected nodes after the execution of line 4 would have been laying within a volume where  $Z \in [0.0, 50]$  and  $X = Y \in [|\infty|]$ . This is however, not the case, and the *D* command can select all entities of all current selections, and assign no degrees of freedom for all dimensions. Effectively creating a surface area of nodes to be clamped. When there are several BC applied to the model the commands can simply be copied and re-instated with changed coordinate values (*Cases: C1.1, C1.3-1.5, C2.1-2.4, C3.1-3.5*).

*Note:* Our model only requires the surface to be clamped, yet the *Y*-value for node selection is set from  $-0.01$  to  $0.01$ . This small tolerance is because the *NSEL* command needs a selection range  $> 0$ .

### Load case

A load is applied the by the *F* command in similar way as assigning BC. The previous selection is selected and the nodes are given a force value in a specified dimension.

```
1 FORCE= -6000
2 F, all , fx ,FORCE,
```

Here we have chosen *fx* and the custom APDL parameter *FORCE* as the dimension and force value, respectively. As *fx* means the *X*-direction and *FORCE*= $-6000$ , the nodes are assigned a force of 6000 in the negative *X*-direction. Applying a second load (*Cases: C1.3-1.4*) is done by re-instating the command with another node selection – much the same as for BC . For a force value not aligned with the global dimensions the *F* command can be issued again with the *same* node selection and a different dimension *Label* thereby creating the amount of force vectors needed to represent the total force direction and size (*Cases: C1.5, C3.5*).

### Static analysis

In this study the force(s) are collected and saved under one *Load case* given by *LSWRITE*. The loads are then deleted so we avoid getting a duplicated load state. We then enter the solution processor and issue the desired equation solver. Preconditioned conjugate gradient solver, *pcg*, is chosen here.

```
1 LSWRITE,1
2 FDEL, all
3 /solu
4 EQSLV,pcg
```

This is the end of the static analysis setup as the solution of the structural problem is done within the topological optimization loop.

## 4.5.2 Topology optimization setup

First the objective and constraint for the topological optimization is defined for the given load case. Compliance is set as the objective and volume as the constraint, with an upper limit of 85% removal ( 92.5% for Cases: C3.1, C3.3, C3.5). The TO solver is set as either *OC*, *Optimality Criteria*, or *SCP*, *Sequential Convex Programming* approach. Here we see that *TOTYPE* lists the solver as *OC*, meaning that *Optimality Criteria* is chosen. The accuracy needed for obtaining convergence is set by the *TODEF* command as 0.0005.

```
1 TOCOMP,comp,single,1 ! Defines compliance function for 1 load case
2 !TOCOMP, Refname , Type , NUMLC , LCARR – Defines single or multiple
   compliance as the TO function.
3 TOVAR,comp,obj ! Sets compliance as the objective with a
4 TOVAR,VOLUME,con,,.85 ! volume constraint with an upper limit 15% of max
   volume
5 !TOVAR, Refname , Type , LOWER , UPPER , Boundtype – Specifies the
   objective and constraints for TO problem.
6 TOTYPE,oc
7 !totype, type – set TO method as either OC or SCP
8 TODEF, 0.0005 ! sets convergence accuracy
```

The optimization loop is executed by the *TOLOOP* command, and the two successive values determine the numbers of iterations to be performed and if plots should be created on every iteration, 1, or only at the last, 0. The option for only writing plots for the last iteration is chosen and the max number of iterations is set to 40.

```
1 TOLOOP,40,0
```



### 4.5.3 Post process visualisation

Ansys Mechanical APDL has the capabilities to produce a wide variety of plots. Convergence graphs for volume and compliance, as well as sectional nodal result and element density plots is produced for each case. Still, these plots do not show a concurrent view of the resulting model, or rather, it is hard to see the whole picture. The nodal results only show the model as the domain box and the element density results are bulky and does not give any indication to internal structure. A macro for exporting the element results to ParaView where therefore written:

```

1 *CFOPEN, modeldens , csv , , append
2 *GET, num_elem_ , elem , 0 , COUNT !Get the number of Elements
3 *GET, elem_ , elem , 0 , NUM, MIN !Get label of the first Element
4 etable , edens , topo
5 etable , estress , s , eqv
6
7 *DO, i , 1 , num_elem_ , 1
8   ! Define some parameters *GET, Par , Entity , ENTNUM, Item1 , IT1NUM, Item2
9   , IT2NUM
10  *GET, nx_ , elem , elem_ , cent , X
11  *GET, ny_ , elem , elem_ , cent , Y
12  *GET, nz_ , elem , elem_ , cent , Z
13  *GET, dens , elem , elem_ , etab , edens
14  *GET, stress , elem , elem_ , etab , estress
15
16  *VWRITE, elmm_ , nx_ , ny_ , nz_ , dens , stress
17  ( E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 )
18  ! select the next element
19  *GET, elem_ , elem , elem_ , NXIH
20
21 *ENDDO
22 *CFCLOSE

```

This macro takes the element density, equivalent stress values and coordinates and writes it to a *.csv* file that can be read by ParaView.

## 4.6 Process description of TopOpt setup

### 4.6.1 Software: Large scale topology optimization code using PETSc

TopOpt is a open source software available from: <http://www.topopt.mek.dtu.dk/>

It uses the PETSc: Portable, Extensible Toolkit for Scientific Computation. The Mod-SIMP approach is used to solve the topology optimization as well as the Method of Moving Asymptote [21].

### 4.6.2 Optimization problem, mesh, and general settings: TopOpt.cc

The first step we do withing the *TopOpt.cc* code is set up the number of grid-points in the each dimension. We then define the domain size and Poisson's ratio, as well as decide the number of multigrid solvers.

```
1  PetscErrorCode TopOpt::SetUp(){
2  PetscErrorCode ierr;
3
4  // SET DEFAULTS for FE mesh and levels for MG solver
5      nxyz[0] = 101;
6      nxyz[1] = 26;
7      nxyz[2] = 101;
8      xc[0] = 0.0;
9      xc[1] = 4.0;
10     xc[2] = 0.0;
11     xc[3] = 1.0;
12     xc[4] = 0.0;
13     xc[5] = 4.0;
14     nu=0.3;
15     nlvls = 1;
```

The values of  $nxyz[0, 1, 2]$ , determine the number of grid-points in the  $X$ -,  $Y$ - and  $Z$ -direction, respectively. The number of grid-points must always be set to one more than the desired amount of elements, since hexahedral elements are created between the points. We see from the code above, at lines 5-7, that the values of 101, 26, 101 are assigned. In turn this gives us the desired mesh discretization

of  $100 \times 25 \times 100 = 250000$  *Elements*. When we need to change the level of discretization (Cases: C21-2.4, C3.1-3.5) it can easily be done by altering these values. The domain size is initiated by setting up the *Max/Min* values for each dimension where:

$$X_{min} = xc[0], X_{max} = xc[1], Y_{min} = xc[2], Y_{max} = xc[3], Z_{min} = xc[4], Z_{max} = xc[5]$$

Effectively creating a  $4 \times 1 \times 4$  domain ( $\blacklozenge Y_{max} = xc[3] = 2$  for Cases: C3.1-3.5). Poisson's ratio  $\nu$  is set to 0.3 and the number of multigrid levels is set to one. It is desirable to set the highest applicable level of multigrid levels active, and this is therefore not a constant throughout this study, but changes depending on the mesh discretization ( $\blacklozenge$  Cases: C2.2-2.4, C3.1-3.5)

*Note:* There is no need to specify any other material properties then Poisson's ratio as TopOpt calculates the element stiffness matrix without implementing Young's modulus.

We then issue the parameters for the topological optimization. Starting with determining the volume constraint, given as a percentage of the retained volume. Here it has a given value of  $volfrac = 0.15 = 15\%$  ( $\blacklozenge 0.075$  for Cases: C3.1, C3.3, C3.5). The maximum number of iterations is set to 80 by assigning a value to  $maxItr = 80$  ( $\blacklozenge 140$  itr. Cases: C3.1-3.5). The SIMP penalty factor is appointed the value  $penal = 3$ .

```

1 // SET DEFAULTS for optimization problems
2 volfrac = 0.15;
3 maxItr = 80;
4     rmin = 0.08;
5     penal = 3.0;
6     Emin = 1.0e-9;
7     Emax = 1.0;
8     filter = 0; // 0=sens,1=dens,2=PDE - other val == no filtering
9     m = 1; // volume constraint
10    Xmin = 0.0;
11    Xmax = 1.0;
12    movlim = 0.25;
13    restart = PETSC.TRUE;
14
15    ierr = SetUpMESH(); CHKERRQ(ierr);
16
17    ierr = SetUpOPT(); CHKERRQ(ierr);
18
19    return(ierr);

```

20 }

*Note:* TopOpt has the ability to chose different radius filtering,  $rmin$ , and gives direct control of the assigned max/min element density values. The move limit can also be changed easily and diffrent filters can be used. They are all set as a constants in this study and the values above are used for *all* cases.

### Physics class: LinearElasticity.cc

This where we define the physics problem for our optimization. In *LinearElasticity.cc* there are two vectors of interest, i.e.  $N$  for BC and  $RHS$  for loads. To assign values and specify the region where these two vectors apply the *if* statement is used. It follows this syntax: *if*(Condition one = *TRUE* && Condition two = *TRUE* && ...){*The following code is applied*}. It checks whether the first condition is *TRUE*, and terminates the sequence if it is *FALSE*. Then it checks the second condition, and so on. Where all conditions are *TRUE* the code within the brackets `{}` can be applied.

```
1     for (PetscInt i=0;i<nn;i++){
2     // Make a BC at x=0 and 125<z<175 with all dofs clamped
3     if ( i % 3 == 0 && lcoorp[i+2] < 4*0.875 && lcoorp[i+2] > 4*0.625
4     && PetscAbsScalar(lcoorp[i]-opt->xc[0]) < epsi ){
5     VecSetValueLocal(N,i,0.0,INSERT_VALUES);
6     VecSetValueLocal(N,++i,0.0,INSERT_VALUES);
7     VecSetValueLocal(N,++i,0.0,INSERT_VALUES);
8     }
9
10    // Make a BC at x=0 and 25<z<75 with all dofs clamped
11    if ( i % 3 == 0 && lcoorp[i+2] < 4*0.375 && lcoorp[i+2] > 4*0.125
12    && PetscAbsScalar(lcoorp[i]-opt->xc[0]) < epsi ){
13    VecSetValueLocal(N,i,0.0,INSERT_VALUES);
14    VecSetValueLocal(N,++i,0.0,INSERT_VALUES);
15    VecSetValueLocal(N,++i,0.0,INSERT_VALUES);
16    }
17    //Load surface 1
18    if ( i % 3 == 0 && lcoorp[i] < 4 && lcoorp[i] > 4*0.75 &&
19    PetscAbsScalar(lcoorp[i+2]-opt->xc[4]) < epsi ){
20    VecSetValueLocal(RHS,i+2,-6000.0,INSERT_VALUES);
21    }
22 }
```

The first condition is a modulo operation which states that  $i$  must be dividable by

---

three. We then issue a number of  $lcoop[i+n]$  is bigger or smaller than statements. This is a PETSc scalar and states that the location of coordinate points in the given  $[i+n]$  dimension are bigger/smaller than the designated value. We have set up two BC at  $Z \in [4 \times 0.625, 4 \times 0.875]$  and  $Z \in [4 \times 0.125, 4 \times 0.375]$  for  $X \approx 0.0$  and  $Y = |\infty|$ . *PetscAbsScalar* defines the tolerance for the dimension normal to the surface plane as the absolute value of *epsi*, which is variable dependent of domain size defined by PETSc utilities. The vector  $N$  for this areas is then assigned the values of 0.0 in each dimension by the *VecSetValueLocal* function, creating two clamped surfaces with no degrees of freedom. Changing the BC is simply done by altering the *lcoop* statement or commenting out code (♦ Cases: C1.2-1.3, C1.6, C3.3-3.4). The load, vector  $RHS$ , is issued the same way, but this time *VecSetValueLocal* is set to the desired load size instead of being clamped. From line 20 we see that a force of -6000 is applied in the  $i+2 = Z$ -direction. Adding a second load (♦ Cases: C1.3-1.4) is done easily by re-inserting the code and changing the values.

## 4.7 Process description of OpenLSTO setup

### 4.7.1 Software: Open Source Level Set Topology Optimization

The *Open Source Level Set Topology Optimization* software suite is, as the name suggest, a free to use open source level set TO software, and uses two C++ based modules for performing level set topology optimization tasks [14]. For this study the latest currently available version, *V1.0 as of 2019*, was used. This version is restricted to only solving tasks with minimum compliance objectives set by a volume constraint. The software was created by M2DO Multiscale Multiphysics Design Optimization Laboratory and can be downloaded from their website, M2DO website: <http://m2do.ucsd.edu/>, or from OpenLSTO's GitHub repository:

```
git clone https://github.com/M2DOLab/OpenLSTO.git
```

The level set optimization used is adapted from Hedges (2017) [10] *LibSLSM* library. As such, the level set module of OpenLSTO can be viewed as extension of the *Stochastic level-set method* presented by Hedges, Kim, and Jack (2017) [11]. OpenLSTO updates the level set for each run in the optimization loop using the *fast marching method* (FMM) introduced by Sethian (1996) [19]. Beyond this, a conjugate gradient solver is used to solve the linear equations, and the *Marching Cubes* method first presented by Lorensen and Cline (1987) [13] is used for boundary discretization [16].

### 4.7.2 Model, meshing and static analysis

#### Model and mesh grid

The first step is defining the dimensionality of the problem. Which is initialised by setting up a *space dimension* value.

```
1  const int spacedim = 3 ;
```

The next step is then setting up the domain. For a 3D problem this is done by choosing the desired number of elements in the X, Y and Z-direction, initiated by

$nelx$ ,  $nely$ ,  $nelz$ , respectively.

```
1      const unsigned int nelx = 100, nely = 25, nelz = 100;
```

Now that the values for the mesh discretization are set up, the volume and subsequent mesh grid must be determined. OpenLSTO does this by first making a *hyperrectangle* given by  $fea\_box$ , then initiating the vector  $nel$  which contains the values for discretization in each space dimension, and last, choosing an element type in  $int\ element\_order$ . The mesh,  $fea\_mesh$ , is then generated from the function  $MeshSolidHyperRectangle$ .

*Note:*  $fea\_box$  is a matrix defined by two steps. First,  $MatrixXd\ fea\_box(8, 3)$  sets up an eight by three matrix environment for  $fea\_box$ , which defines a box with 8 external corner points by 3 space dimensions. Second, the  $fea\_box$  is then given the coordinate values for the eight corners of the *hyperrectangle*.

```
1      MatrixXd fea_box (8, 3) ;
2
3      fea_box <<  0.0, 0.0, 0.0,
4                  nelx, 0.0 , 0.0,
5                  nelx, nely , 0.0,
6                  0.0, nely,0.0 ,
7                  0.0, 0.0 ,nelz ,
8                  nelx, 0.0 ,nelz ,
9                  nelx, nely ,nelz ,
10                 0.0, nely ,nelz ;
11
12      vector<int> nel = {nelx , nely , nelz} ;
13
14      int element_order = 3 ;
15      fea_mesh.MeshSolidHyperRectangle (nel, fea_box , element_order , false)
16      ;
17      fea_mesh.is_structured = true ;
18      fea_mesh.AssignDof () ;
```

The object  $fea\_mesh$  is given degrees of freedom from the function  $AssignDof$ . With the chosen element type of 3, the following DoF are assigned:

”For 3D meshes, the solid element implemented has 8 nodes, each having x-, y- and z-direction displacements as degrees of freedom.”  
[14, Programmer Manual for OpenLSTO, pages 8-9.]

## Material Properties

Material properties are added to the *fea\_mesh* vector by using a *push\_back* function with the values obtained from *solid\_materials*. In this case:

Young's modulus  $E = 2e5$     Poisson's ratio  $\nu = 0.3$     Density  $\rho = 1.0$

```
1 // SolidMaterial (<geometry dimension of structure >, <Young's modulus >, <
   Poisson's ratio >, <density >)
2 fea_mesh.solid_materials.push_back (FEA::SolidMaterial (spacedim ,
   200000, 0.3, 1.0)) ;
```

Now *fea\_mesh* holds all the information needed. And so, before adding BC and loads, the object *fea\_study* is defined, with the *fea\_mesh* as the starting value.

```
1 FEA::StationaryStudy fea_study (fea_mesh) ;
```

## Boundary condition

Adding a boundary condition in OpenLSTO is done by; first, initiating vectors which hold the coordinates of a given set of nodes, and second, assigning new DoF to these nodes. The function *GetNodesByCoordinates* is used to retrieve a set of nodes from *fea\_mesh*. It does so by setting up a point in space and the tolerance in each direction. The following format is used; ( $\{X \text{ coordinate}, Y \text{ coordinate}, Z \text{ coordinate}\}, \{X \text{ direction absolute tolerance}, Y \text{ direction abs tol}, Y \text{ direction abs tol}\}$ ). Looking at our setup, we see that on line five in the code below the retrieved nodes are contained within the volume  $V$ :

$$V = X \in [-1e - 12, 1e - 12], Y \in [-1e9, 1e9], Z \in [0.125 \times nelz, 0.375 \times nelz]$$

```
1 /*
2 Add a homogeneous Dirichlet boundary condition (fix some nodes).
3 */
4
5 vector<int> fixed_nodes = fea_mesh.GetNodesByCoordinates ({0.0, 0.0 ,
   0.25*nelz }, {1e-12, 1.0e9, 0.125*nelz});
```

*Note:* Even though our study only specifies **BC areas** the  $X$ -tolerance is not set to zero, but given an infinitesimal small value of  $tol = 1e - 12$ . This is because the function will not retrieve any nodes when it is given an infinitely thin plane. For all practical purposes the nodes within the volume can be viewed as a surface area.

For our case we needed to add a second set of fixed nodes. The method we

---



chose was to initiate a new vector and, as prior, retrieve nodes from the mesh. We then inserted the second vector *fixed\_2* at the end of the first vector *fixed\_nodes* using the *insert* function. For cases with only one BC the second vector and *insert* function can easily be commented out (Cases: C1.2-1.3, C1.6).

```

1  vector<int> fixed_2 = fea_mesh.GetNodesByCoordinates ({0.0, 0.0 , 0.75*
      nelz }, {1e-12, 1.0e9, 0.125*nelz});
2  fixed_nodes.insert(fixed_nodes.end(), fixed_2.begin(), fixed_2.end());
3  vector<int> fixed_dof = fea_mesh.dof (fixed_nodes) ;
4
5  vector<double> amplitude (fixed_dof.size(),0.0) ; // Values equal to
      zero .
6
7  fea_study.AddBoundaryConditions (FEA::DirichletBoundaryConditions (
      fixed_dof , amplitude , fea_mesh.n_dof)) ;

```

The nodes are finally chosen and the desired DoF can be assigned. Vector *fixed\_nodes* is used as the input for the function *fea\_mesh.dof*. The result is set as the initial values of a new vector, called *fixed\_dof*. Then the vector *amplitude* is defined by taking *fixed\_dof* and using the function *size* to set the values to zero – meaning no degrees of freedom, the nodes are clamped at their location. The two set of nodes and their DoF are ultimately added to *fea\_study* object using the function *AddBoundaryConditions*.

### Load case

Applying loads is done under the same principle as adding BC. Get the coordinates from the mesh and assign DoF. Adding a second load (Cases: C1.3-1.4) is also done using the *insert* function. For our default setup we only have one load surface active. The second vector *load\_node2* is therefore commented out, as well as the *insert* function (Cases: C1.1, C1.4-1.5, C2.1-2.4, C3.1-3.5).

```

1  /*
2     Apply load .
3  */
4
5  // vector<int> load_node2 = fea_mesh.GetNodesByCoordinates ({0.375*nelx ,
      0.0*nely ,0.0*nelz }, {0.125*nelx , 1.0e9 , 1.e-12});
6  vector<int> load_node = fea_mesh.GetNodesByCoordinates ({0.875*nelx , 0.0*
      nely ,0.0*nelz }, {0.125*nelx , 1.0e9 , 1.e-12});
7  // load_node.insert(load_node.end(), load_node2.begin(), load_node2.end())
      ;
8  vector<int> load_dof = fea_mesh.dof (load_node) ;

```

The force at the location is added by specifying the the size of the *load\_node* vector in each space dimension, and the values are stored in the *load\_val* vector. The default value for this study equals  $-6.0$  in  $spacedim * i + 2$  (which means the *Z*-direction).

```
1     vector<double> load_val (load_node.size() * spacedim) ;
2
3     for (int i = 0 ; i < load_node.size() ; ++i) {
4
5         load_val[spacedim*i]   = 0.0 ;
6         load_val[spacedim*i+1] = 0.0 ;
7         load_val[spacedim*i+2] = -6.0 ;
8
9     }
```

The values for location, size and DoF are then added to the *fea\_study* using the *AssembleF* function.

```
1     FEA::PointValues point_load (load_dof , load_val) ;
2     fea_study.AssembleF (point_load , false) ;
```

### 4.7.3 Topology optimization setup

The first step of the topological optimization setup is defining an object of the level set class. The object, which is named *level\_set\_3d*, is then given the values for setting up dimensions. As seen in the code below, the level set dimensions are the same as the *nel* dimensions.

```
1     // Create an object of the levelset class
2     LevelSet3D level_set_3d ;
3
4     //Declare box dimensions and initialize the box
5     std::vector<double> LS2Femap(3,1) ;
6
7     uint box_x = nelx*LS2Femap[0];
8     uint box_y = nely*LS2Femap[1];
9     uint box_z = nelz*LS2Femap[2];
10
11     level_set_3d.SetBoxDimensions(box_x, box_y, box_z); // Set up dimensions
```

Now the function *MakeBox* is used on the *level\_set\_3d* object. This creates the

actual model that will be optimized for the level set class, and is a *hyperrectangle* with the same size as the dimension values.

```
1 level_set_3d.MakeBox ();
```

*Note:* An alternative function is the *MakeLBeam* (makes an L-beam) where the following *uint nxl* =< value >; and *uint nzl* =< value >; for *X*-thickness and *Z*-thickness, needs to be specified.

The volume constraint is set up as *MaxVol* and then passed on to the *SensData* object. The value equals the retained volume in percentage. It is set to 15% as our default value (♦ 7.5 for Cases: C3.1, C3.3, C3.5).

```
1 double MaxVol = 15.0; // in percentage
2 SensData.MaxVol = MaxVol;
```

Specifying the move limit is done by assigning a value to *move\_limit*. The move limit information is also passed on to the *SensData* object. The move limit a constant and the value of 0.25 is used throughout the study.

```
1 double move_limit = 0.25;
2 SensData.move_limit = move_limit;
```

Setting the limit for the number of iterations is done by changing the value of *max\_iterations* to the desired maximum value. Here given as a maximum of 80 iterations to be performed (♦ 140 for Cases: C3.1-3.5).

```
1 int n_iterations = 0;
2 int max_iterations = 80;
3
4 while (n_iterations < max_iterations) {
5
6     ++n_iterations ;
7
8     //The rest of the optimization loop is not included here. See comp_min.cc
9
10 }
11
12 // write to stl
13 int box_smooth = 0;
14 level_set_3d.WriteSTL(box_smooth);
```

The loop *while* counts the number iterations and terminates when the criteria is no longer fulfilled. The results are written to an *STL* file. In our case the smoothing option is turned *off*, this means *box\_smooth* = 0. With smoothing turned *on* the results often contain unsupported geometry as the smoothing refines the texture too much in the process of obtaining an even surface-model.

## 4.8 Post process visualisation in ParaView

The post-processing visualisation is done through ParaView. For OpenLSTO a change in the *WriteSTL* function gives us file we can use directly:

```
1      txtfile << "endloop" << endl;
```

Changing *end loop* to *endloop*

```
1      txtfile << "endloop" << endl;
```

The Ansys results are given as weighted element plots, where each element is represented by a dot. This is due to the models containing internal structures that are otherwise hard to visualize.

The TopOpt results are viewed directly without any modification.

# Chapter 5

## Assessment One: Load case variation

### Chapter introduction

Results of particular interest:

Does the topology of a specific case conform to a similar shape or do we see considerable differences between the programs? Are there topological features that are present between all programs, i.e. *global key features*? How does the complexity of the load and boundary setup effect solution time and convergence? Do we see any trends in the programs with regards to the structural shape? In cases where the topology is similar do we also see the same compliance values? Does one program perform better than the others with regards to compliance and solution time?

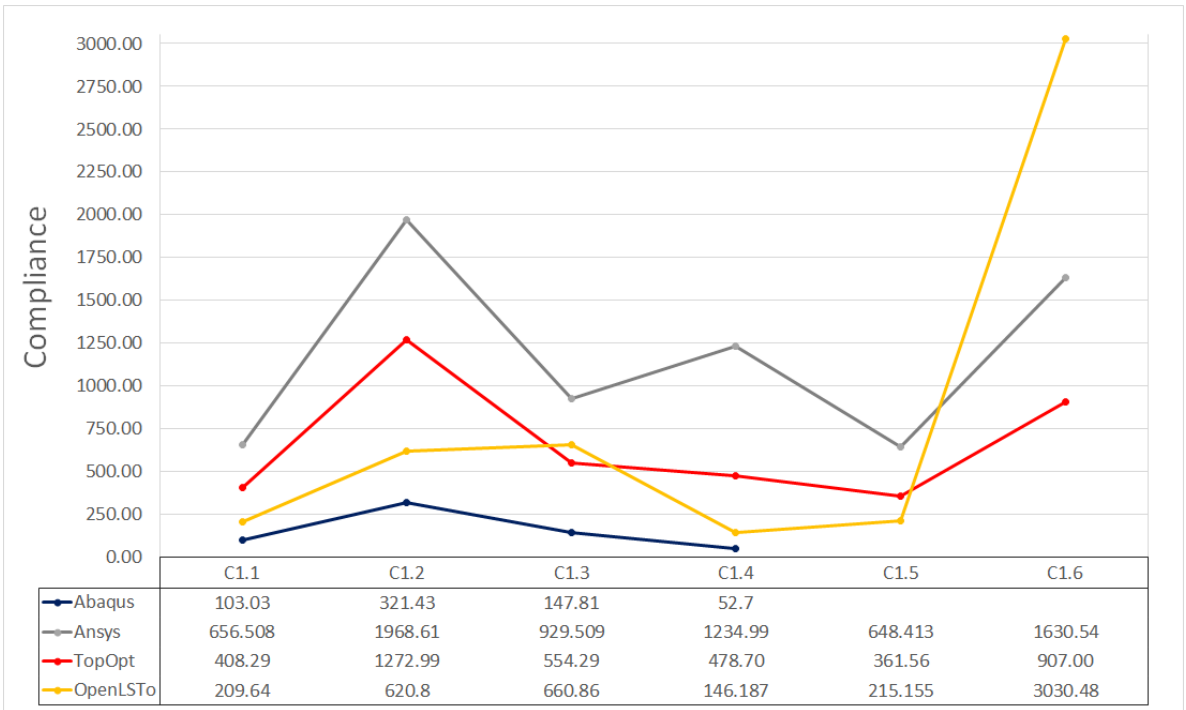
## 5.1 Results for cases C1.1-1.6

### Main findings

The assessment show that the different programs default to a preferred topological orientation. Abaqus and TopOpt generally create a model with a truss like structure. The difference being that Abaqus prefers thicker truss members, while TopOpt usually produces a slimmer design with more topological features. Ansys typically utilize more of the domain width than the others, and create a model with thin walls containing internal truss structures. OpenLSTO generally produces a single solid body without holes (with the exception of Case C1.3). The investigation also show, rather surprisingly, that topological similarity does not equal the same compliance. Furthermore, the compliance levels for Ansys generally conform to a similar relative increase from the other programs. A proposed reason for this behavior is the higher node order element used. The result show that TopOpt has the fastest solution time per iteration, with Abaqus coming at a close second. The complexity of the load configuration show no global trends with regards to solution time.

### 5.1.1 Compliance

Figure 5.1 shows the compliance results for all cases across all programs. With the objective of minimizing compliance, the Abaqus results are the most satisfactory. Following this, the general trend is that OpenLSTO performs second best, then TopOpt, and Ansys show the highest compliance for most cases. The deviation from this trend is the case C1.3 where the compliance of TopOpt is 83.9% that of OpenLSTO, and case C1.6 where the compliance of OpenLSTO is 186% and 335% of Ansys and TopOpt respectively. Table 5.1 lists the variances between the programs for cases C1.1-1.4, with the largest discrepancy being between Abaqus and Ansys, where the compliance of the Ansys solution is a whole 2343.4% of Abaqus. The smallest variation in result is found between TopOpt and OpenLSTO in the previous mentioned case C1.3, i.e. TopOpt compliance is 83.9% of OpenLSTO.



**Figure 5.1:** Compliance results for all cases across all programs

### 5.1.2 Topology

Table 5.2 shows the results for the topology of cases C1.1-1.4, and describes shortly the similarity or discrepancy between the solutions, as well as it lists the topological key features. The abbreviations Abaqus = **Aba**, Ansys = **Ans**, TopOpt = **Top** and OpenLSTO = **LSTO** is used in the table to differentiate between the programs.

In the case C1.1, the model has two fixtures and one load. The key feature between all four models is the reinforcements from the top fixture continuing down to the load surface, and a reinforcement from the bottom fixture to the load surface. Effectively creating a simple truss structure consisting of two members. The topology of Abaqus and TopOpt closely resemble one another, yet from Table 5.1 the difference in compliance is listed as TopOpt being 396.2% of Abaqus, i.e. almost four times as big. This does not correlate with the assumption that models

		Abaqus	Ansys	TopOpt	OpenLSTO
case 1.1	Abaqus		637.2	396.2	203.4
	Ansys	15.6		62.19	31.9
	TopOpt	25.2	160.8		51.3
	OpenLSTO	49.1	313.1	194.7	
case 1.2	Abaqus		612.5	396.0	193.1
	Ansys	16.3		64.7	31.5
	TopOpt	25.3	154.6		48.7
	OpenLSTO	51.8	317.1	205.1	
case 1.3	Abaqus		628.9	375.0	894.2
	Ansys	15.9		59.6	71.1
	TopOpt	26.7	167.7		<b>119.2</b>
	OpenLSTO	22.4	140.7	83.9	
case 1.4	Abaqus		<b>2343.4</b>	908.3	277.4
	Ansys	4.3		38.8	11.5
	TopOpt	11.0	258.0		30.5
	OpenLSTO	36.0	844.8	327.5	

**Table 5.1:** Variances between the programs for cases C1.1-1.4

with seemingly equal topology obtain roughly the same compliance. The Ansys model have the same main features as Abaqus and TopOpt, as well as a thinner and more intricate set of supporting sub-beams. Again, the difference in compliance is substantially higher then the difference in model topology would indicate, Ansys is over six times more compliant than Abaqus. With OpenLSTO the topology does not resemble the other models to a high degree. The distinction between topological features is not easy to make, as the model is one solid body without holes present. The compliance here is roughly twice as high as that of Abaqus.

Case 1.2 removes the bottom fixture from the previous case and leaves the model with one fixture and one load. Abaqus, Ansys and TopOpt show the same key features, while the OpenLSTO model conforms to as single solid body with virtually no holes. Table 5.1 shows that the variation in compliance is roughly the same as in case C1.1.

Case 1.3 features one fixture and two loads. The key features between all four models is the two legs dropping down to support the loads. Beyond this, all models contain some members between the legs, with the exact topology of the



strengthening varying. Abaqus and Ansys share the most features, but Abaqus has the lowest compliance, while Ansys has the highest. TopOpt and OpenLSTO which share very few features, yet have almost the same compliance, with 554.3% and 660.9% respectively.

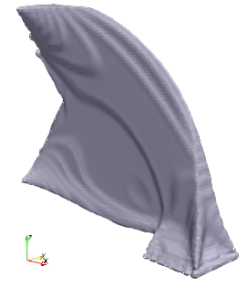
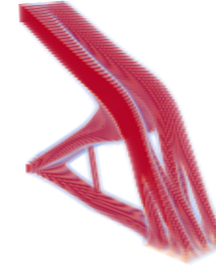
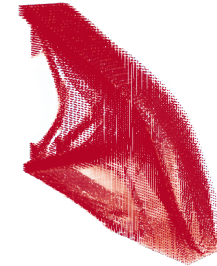
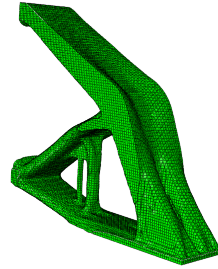
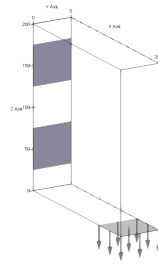
Case 1.4 features two fixtures and two loads. A distinctive main support beam running from the top fixture to the furthest load is recognisable. All programs except OpenLSTO make use of the entire domain width. OpenLSTO keeps all material centralised in the middle of the width of the model. The three first methods result in a truss-like structure seen from the side, with varying degree of attachment to the fixture wall. Here the two most unlike models have similar low compliance, while TopOpt and Ansys reach 478.7 and 1235.0 respectively.

Table 5.2 : Load case Variation C1.1 – 1.4 Results

Configuration      Abaqus      Ansys      TopOpt      OpenLSTO

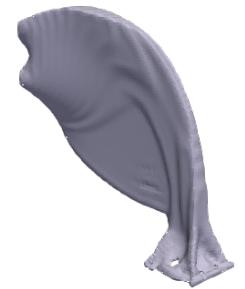
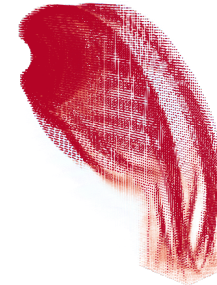
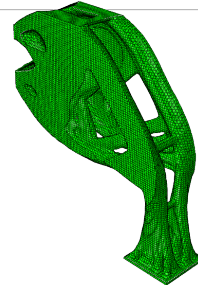
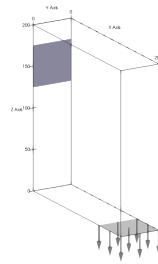
case1.1

**Aba** and **Top** produces a near identical model. They inherit all the same topological features. The **Ans** result is similar, yet it represents a more complex topology. **LSTO** appears alien in relation to the others. Still, there are global key features; an upper and lower reinforcement.



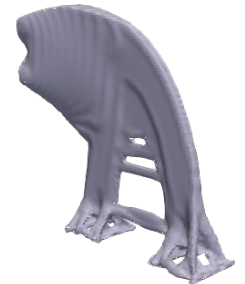
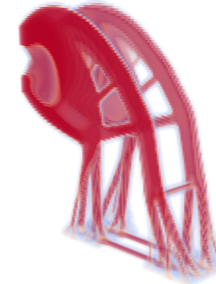
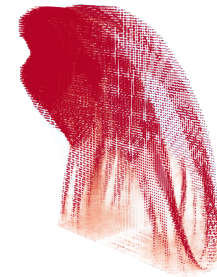
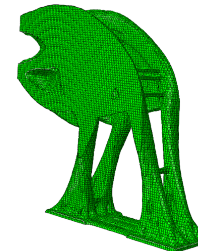
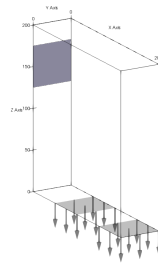
case1.2

The discrepancy between **Aba**, **Ans** and **Top** is here small. Global features would be the upper reinforcement and the dual connectivity at the boundary location. **LSTO** is an outlier again. Instead of two *truss structures* at the width of the domain a single body is produced.



case1.3

Global features; an upper body with dual connectivity at the boundary, two supporting *legs* and a connecting bridge in between. Interestingly **Aba** and **Ans** share similar features, and **Top** and **LSTO** also closely resemble another (not accounting for the dual versus single body issue).



case1.4

Again we see that **Aba** and **Ans** share a set of key features. The key features of **LSTO** are also present within the more complex **Top**. The global features are the same as in C1.1, an upper and lower reinforcement. This due to the middle section, though similar, containing different features across the programs.

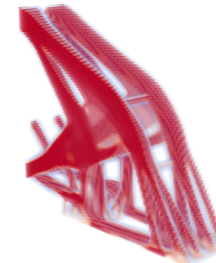
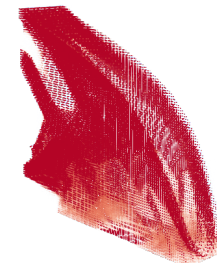
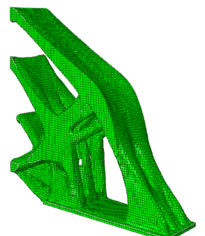
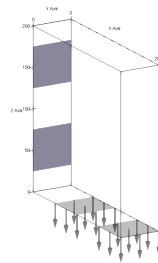
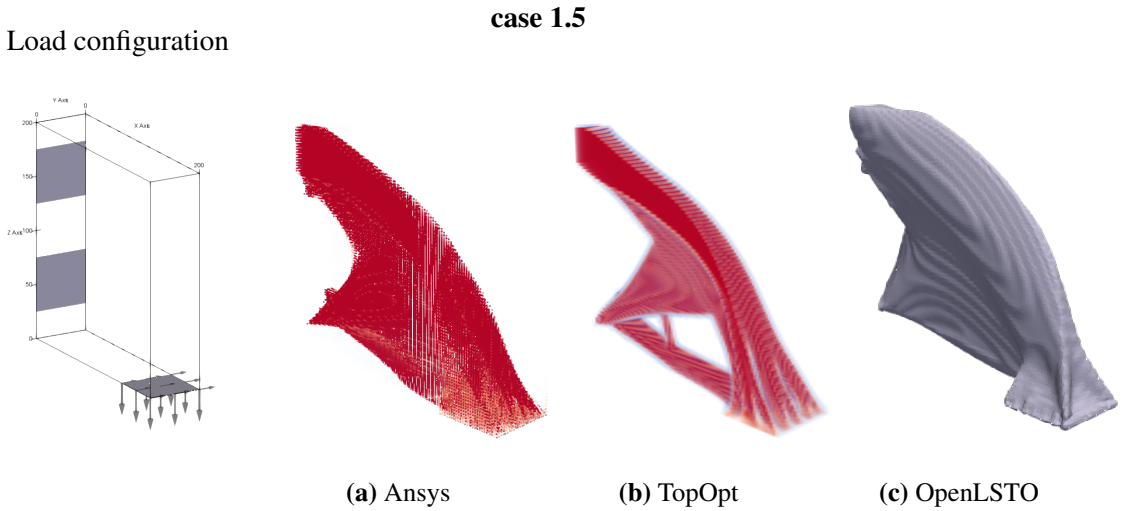
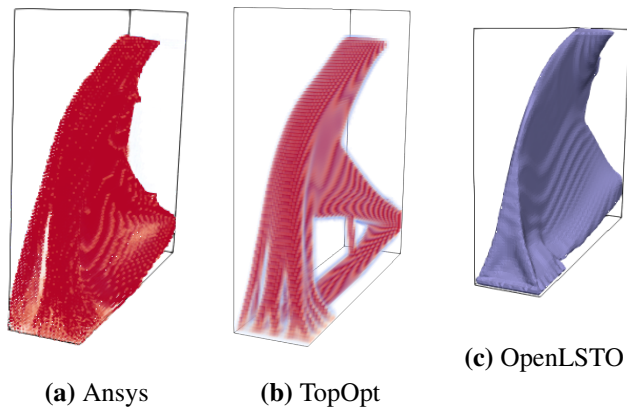


Figure 5.2 shows the model results for case C1.5. This case is identical to C1.1, except a new load vector is added with an orientation parallel to the  $y$ -direction and with a value of  $F_y = 1000$ . OpenLSTO produces a big thick arch, while the rest of the body remains thin walled. Ansys is almost identical to OpenLSTO, while TopOpt is the one which stands out as different, having a more complex topology. While Ansys and OpenLSTO produce practically the same model, it is interesting to see that the compliance of Ansys is three times higher.

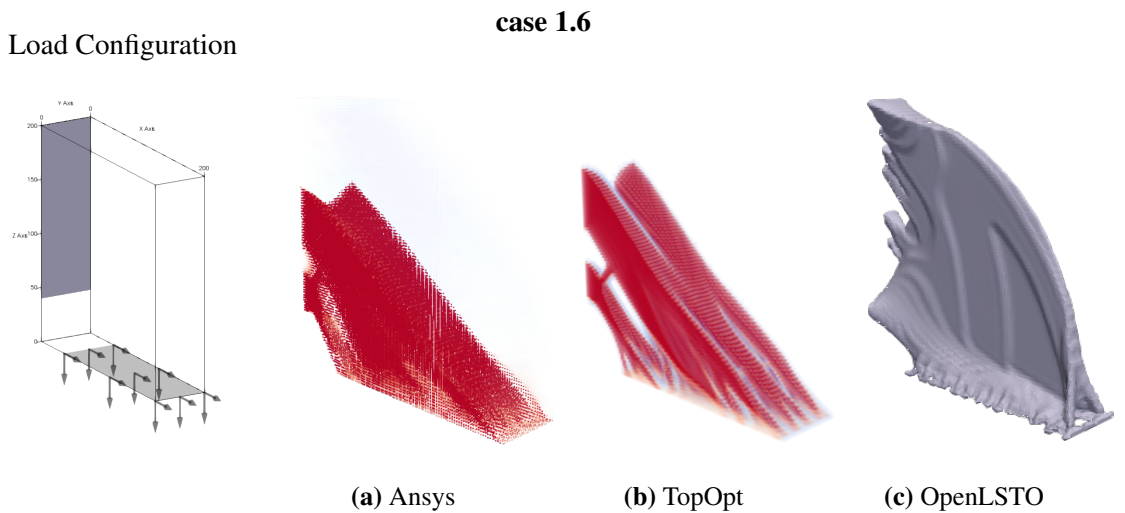


**Figure 5.2:** Description of load configuration and the resulting models in case 1.5

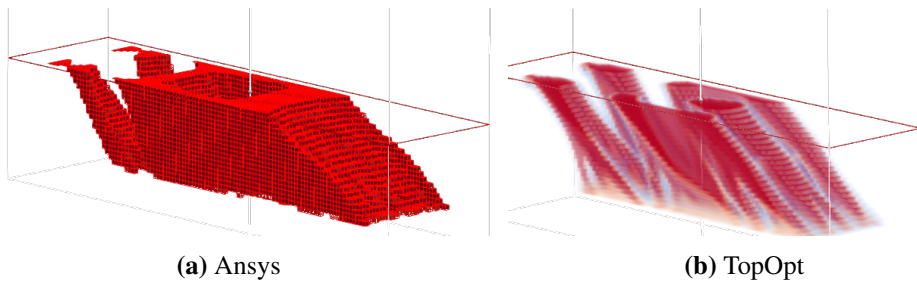


**Figure 5.3:** Continuation of model views from case 1.5

The cross section of the case 1.6 results are shown in Figure 5.5. Here we can observe that TopOpt uniformly disperses the material onto seven thin beams, across the entire section, while Ansys joins five of the seven beams into one hollow feature, roughly remaining the same material distribution. Compliance for OpenLSTO very high, not surprising as the load vector is 45 degrees and creates a pure tensile stress situation. The "arch" created by OpenLSTO does not help the structure significantly.



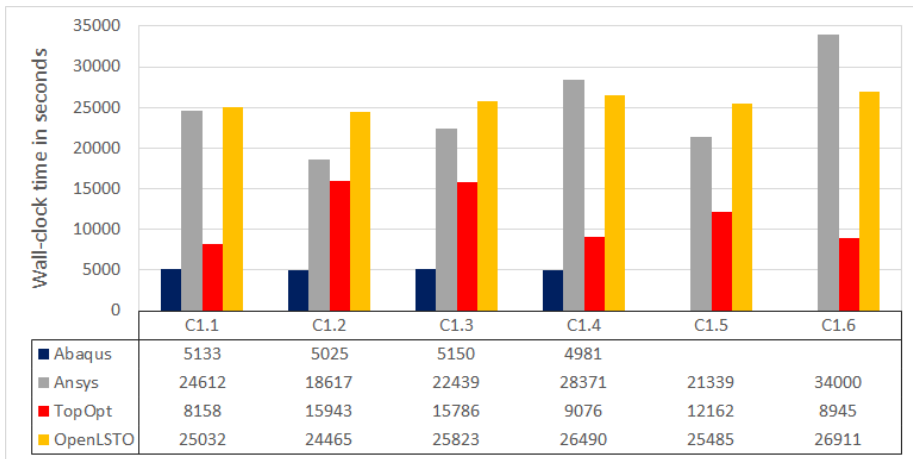
**Figure 5.4:** Description of load configuration and the resulting models in case 1.6



**Figure 5.5:** Case 1.6, cross section looking down the y-axis

### 5.1.3 Computation time

Figure 5.6 and Table 5.3 show the results for the total time usage and average time per iteration. Abaqus uses less time in total, but stops at max 30 iterations while Ansys goes to 40, and TopOpt and OpenLSTO solves 80 iterations. From the results in Table 5.3 it can be observed that TopOpt has the fastest average solver time per iteration at 146.0 seconds. The question as to which program solves the problem most rapidly depends on at which iteration the solver reaches a satisfactory solution.



**Figure 5.6:** Total time usage in seconds

	case 1.1	case 1.1	case 1.1	case 1.1	case 1.1	case 1.1	Total Average
Abaqus	171.1	<b>167.5</b>	<b>171.6</b>	166			169.1
Ansys	615.3	465.42	560.9	709.3	711.3	1619.0	780.2
TopOpt	<b>102.0</b>	199.3	197.3	<b>113.4</b>	<b>152.0</b>	<b>111.8</b>	<b>146.0</b>
OpenLSTO	312.9	305.8	322.8	331.1	318.6	336.4	321.3

**Table 5.3:** Average time used per iteration in seconds

## 5.2 Discussion

The assessment show that the different programs default to a preferred topological orientation. Abaqus and TopOpt generally create a model with a truss like structure. The difference being that Abaqus prefers thicker truss members, while TopOpt usually produces a slimmer design with more topological features. Another noticeable difference is how smooth the TopOpt models are. There are no sharp edges present throughout the study. This most likely comes from the radius filtering ability TopOpt possesses. Back to the preferred topological orientation we see that Ansys typically utilize more of the domain width. It creates a model with thin walls containing internal truss structures. OpenLSTO generally produces a single solid body without holes. From a manufacturing perspective these default orientations would indicate that Abaqus, Ansys and TopOpt models would require substantial post-processing. This bridges the connection between additive manufacturing and TO, as we understand that the complexity of the topology makes traditional processes unsuitable. The models from OpenLSTO on the other hand, can with minimal changes be manufactured in for example a molding process.

Perhaps the most curious observation from this assessment; models with the same topology do not obtain the same compliance. Also: the relative difference in compliance between programs is mostly the same throughout the study. In case C1.1 Abaqus and TopOpt are practically the same, and in case C1.5 Ansys and OpenLSTO are near identical. The same on the other hand, can not be said about the compliance values. The results from Table 5.1 show that for C1.1 the difference in compliance is the second largest between the programs. For the case C1.5 the difference in compliance between Ansys and OpenLSTO (33.2%) is practically the same as it is for cases C1.1 (31.9%) and C1.2 (31.5%), yet the topology in these cases do not share a resemblance. Why do we see this behavior?

One reason could be that different constitutive matrices are used to calculate the stiffness. TopOpt does for example not implement the elastic modulus in its calculations, while we know this is a necessary input for Abaqus and Ansys. Of course it could be that one model is in fact much stiffer, yet with the same material and topology the difference should be limited. Continuing this train of thought,

could it be that the topology is in fact not similar? Meaning the visual inspection is not satisfactory. This does not seem likely. The answer probably lies within how the sensitivities are handled internally between different classes in the programs.

As for the case with Ansys, a 20-node element instead of 8-node element is used, this may be why the compliance values are scaled up. Each element contains 2.5 times more nodes. This scaling magnitude is seen between Ansys and either TopOpt or OpenLSTO in most cases.

TopOpt and Abaqus both show a quick solution time, with TopOpt being slightly faster. A consideration to be made is that TopOpt is not utilized to its full potential. The number of multi grid solvers used is set to one, since the discretization of  $100^2 \times 25$  is not dividable by two,  $25 \div 2 = 12.5$  and elements cannot be split. The evaluation that Abaqus and TopOpt performs equally well may therefore be erroneous. This relationship is further investigated in the following assessment. As stated a prior, Ansys also uses a higher order element, which most likely affect the solution time considerably.





# Chapter 6

## Assessment Two: Mesh discretization dependency

### Chapter introduction

Results of particular interest:

At what level of discretization do the individual programs obtain viable results? Does the level of discretization affect the similarity or discrepancy between programs? Do we only see a refinement or do we actually see a different topology as the mesh size increases? How does the discretization affect the solution time and the convergence? How is the compliance affected by the mesh size? And do we see the same trends here as in Assessment One?

## 6.1 Results Cases C2.1-2.5

### 6.1.1 Main findings

A mesh size of  $64^2 \times 16 = 65536$  *elements* gave viable results for all programs. OpenLSTO also show the most satisfactory solutions at coarser mesh sizes, and is the only model where new topological features are introduced at every mesh size. Furthermore, the mesh discretization dependency investigation show that for Ansys, TopOpt and OpenLSTO the compliance increases with mesh size. This establishes the proposition that higher level of mesh discretization increases the accuracy of the results.

Building on the results from the previous assessment the results here show that the relative difference between Ansys and the other programs, especially TopOpt, is still the same, reinforcing the proposition that the higher node order element is the cause of the increased compliance. The investigation also shed light on the effect of smoothing of non-viable model results, proposing that post-processing of undesirable models can easily be done to attain viable results. The computation time increases as the mesh discretization increases, and TopOpt has the fastest solution time. Abaqus performs noticeably slower than the other programs on coarser mesh grids.

## 6.1.2 Explanation of categorization of dependency

A categorization of dependency is introduced for this assessment. This provides exact criteria needed to evaluate whether or not the topology of a model has reached a satisfactory state. The following labels are used:

**Unsupported** – The model contains unsupported geometry. Checkerboard patterns and/ or high concentration of low density elements exist in vital locations. The results are not satisfactory.

**Non-conflicting** – There are no conflicting elements present, yet the model has not reached a desired topology. It is coarse and it can be hard to separate it into key features.

**Preferable** – We have a desired topology. It is relatively smooth and there are no conflicting elements appearing. It is easy to assess the key features of the part.

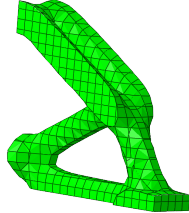
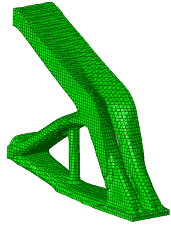
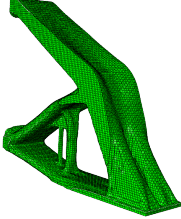
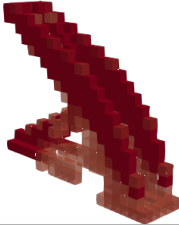
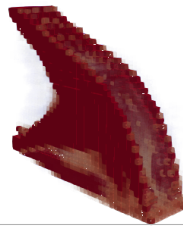
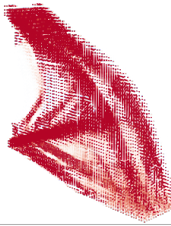
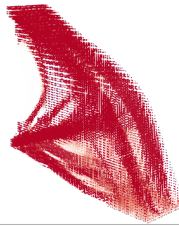
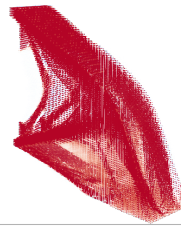
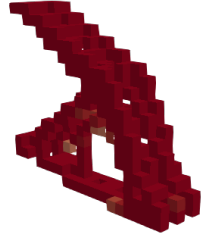
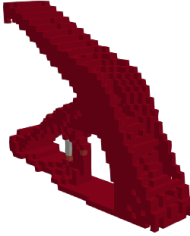

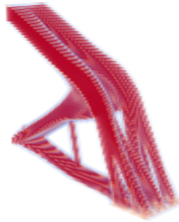
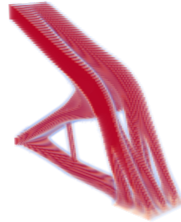
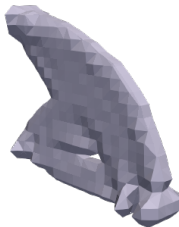
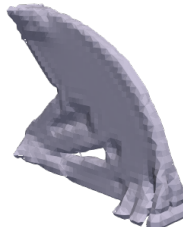


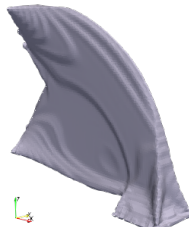
**Refined** – Still being preferable, the model has reached a state where the topology is now the same as in the previous case, now even more refined. No new features or changes are present.

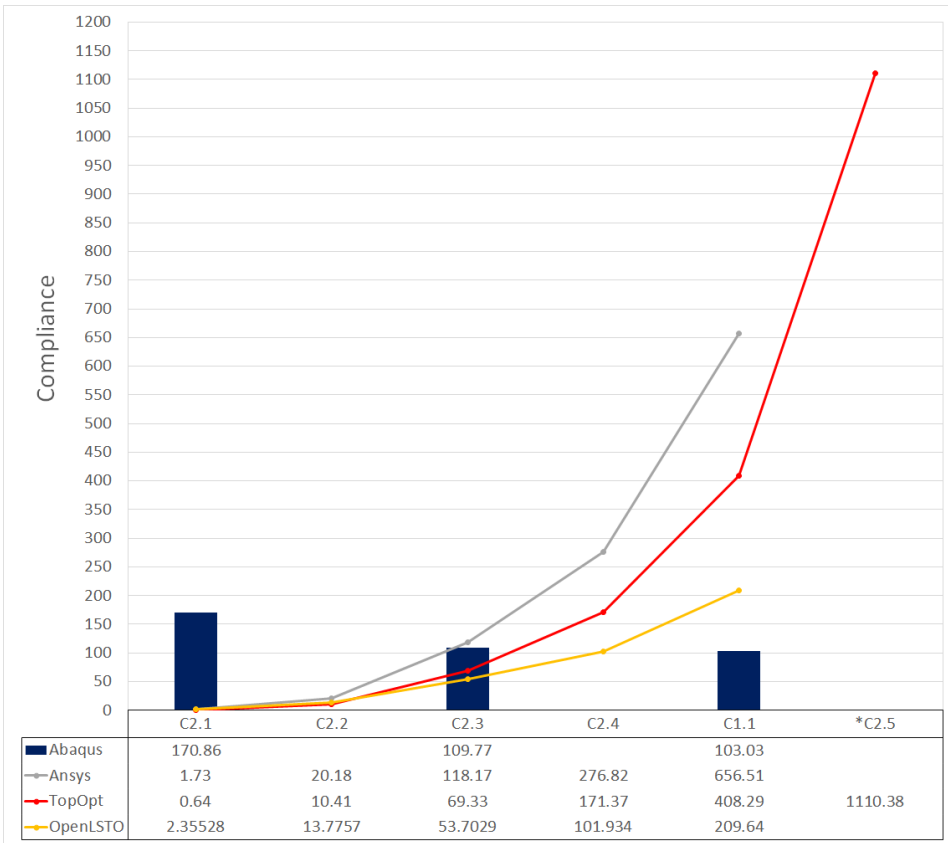
**New feature** – The same as refined, yet there is also the introduction of one or more new features.

## 6.1.3 Presentation of results

The results for topology and assessment of dependency is given in Table 6.1. The abbreviations Abaqus = **Aba**, Ansys = **Ans**, TopOpt = **Top** and OpenLSTO = **LSTO** is also used in this table to differentiate between the programs. Figure 6.1 show the compliance results and explains the trends in this assessment. Table 6.2 show the result for solver time per iteration and Figure 6.2 show the total time used. Information regarding convergence is found in Table 6.3. The special case C2.5 is presented in Figure 6.3. Figure 6.4 show how an initially conflicting model can obtain a viable result by post-processing.

*Table 6.1 : Mesh Discretization Dependency Results*

	<i>Case 2.1</i> 20 × 20 × 5	<i>Case 2.2</i> 40 × 40 × 10	<i>Case 2.3</i> 64 × 64 × 16	<i>Case 2.4</i> 80 × 80 × 20	<i>Case 1.1</i> 100 × 100 × 25
<p><b>Abaqus</b></p> <p>A preferable model is obtained in C2.3. Interestingly the coarse solution does not feature any conflicting elements, and the surface is smooth compered to <b>Ans</b> and <b>Top</b>. Case C1.1 is only a refinement of C2.3.</p>	Non-conflicting 	N/A	Preferable 	N/A	Refined 
<p><b>Ansys</b></p> <p>A preferable model is obtained in C2.3. The cases C2.1-2.2 both contain unsupported geometry and a high degree of elements with low density. The model is refined at higher levels of mesh discretization.</p>	Unsupported 	Unsupported 	Preferable 	Refined 	Refined 
<p><b>TopOpt</b></p> <p>The cases C2.1-2.2 both contain checkerboard patterns. A preferable model is obtained in C2.3. The model is refined when higher level of mesh discretization is implemented. The key featurer are present even at the coarsest mesh.</p>	Unsupported 	Unsupported 	Preferable 	Refined 	Refined 
<p><b>OpenLSTO</b></p> <p>Non-conflicting results are present in cases C2.1-2.2. The results seen here are the most satisfactory for these coarse mesh sizes. A preferable result is obtained in case C2.3, but new topological features are introduced at each successive step of the mesh refinement.</p>	Non-conflicting 	Non-conflicting 	Preferable 	New feature 	New feature 



**Figure 6.1:** Compliance for cases C2.1-2.5 benchmarked against case C1.1. The cases are sorted in order of mesh discretization level. Ansys, TopOpt and OpenLSTO follow the same trend, the mesh discretization increases along with compliance. Abaqus shows the exact opposite pattern, i.e. the compliance decreases. The overall variation in compliance is lower than previously observed for case C2.3, with a  $64^2 \times 16$  mesh. Ansys shows a compliance 2.2 times that of OpenLSTO. The previous case with the lowest maximum variation was case C1.2, where the difference was a magnitude of 6.12 between Ansys and Abaqus.

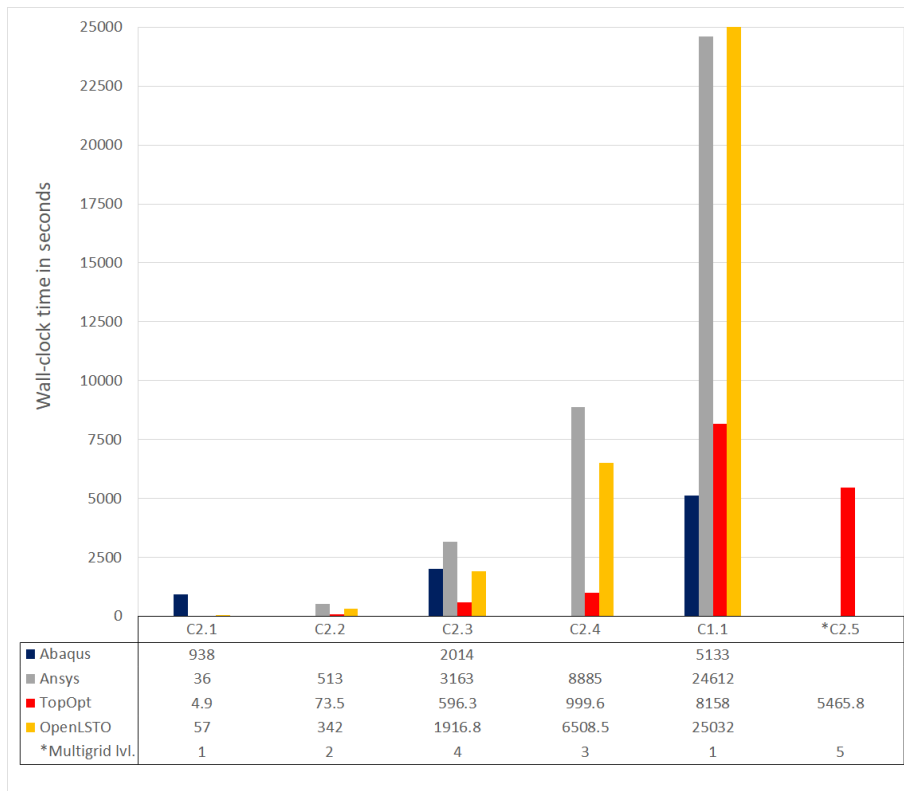
\*C2.5 is a TopOpt exclusive result, the problem size proved to be too big for the other programs with the current setup.

Mesh	Case 2.1 $20^2 \times 5$	Case 2.2 $40^2 \times 10$	Case 2.3 $64^2 \times 16$	Case 2.4 $80^2 \times 20$	Case 1.1 $100^2 \times 25$	Case 2.5 $128^2 \times 32$
Abaqus	24.2		59.7		171.1	
Ansys	1.4	13.2	79.1	222.1	615.3	
TopOpt	<b>0.2</b>	<b>1.2</b>	<b>7.5</b>	<b>12.5</b>	<b>102.0</b>	<b>68.3</b>
*Multigrid lvl.	1	2	4	3	1	5
OpenLSTO	0.7	4.3	24.0	81.4	312.9	**1300

**Table 6.2:** Average time used per iteration in seconds. Abaqus performs poorly in case C2.1, solving each iteration 121 times slower than TopOpt.

\*TopOpt utilizes the capabilities of the multigrid levels where applicable and performs exceedingly well.

\*\*The time for Case C2.5 for OpenLSTO was calculated from the time it took to run the three first iterations and was then terminated. Lower level of accuracy is to be expected in this result.

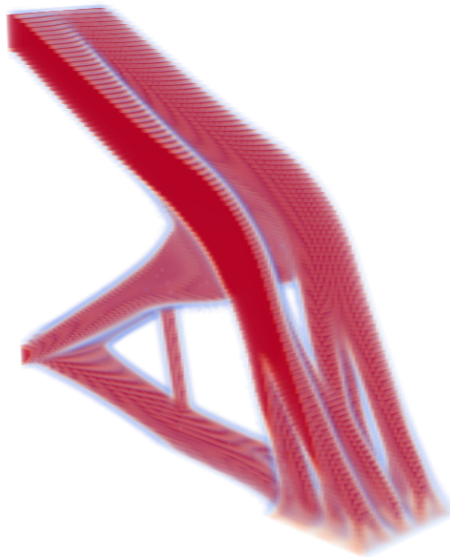


**Figure 6.2:** Total time usage in seconds.

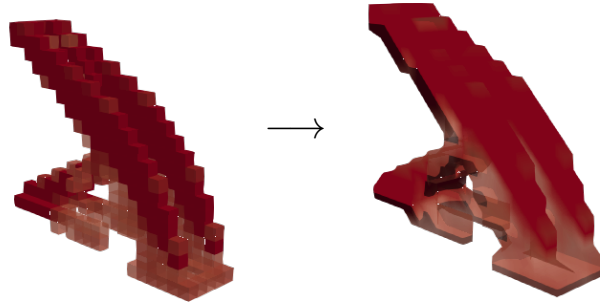
TopOpt performs exceedingly well when looking at the total time as well.

		Case 2.1	Case 2.2	Case 2.3	Case 2.4	Case 1.1	Case 2.5
Abaqus	Convergence	<i>No</i>		<i>Yes</i>		<i>No</i>	
	Last itr.	30		29		30	
Ansys	Convergence	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>No</i>	
	Last itr.	25	39	40	40	40	
TopOpt	Convergence	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>
	Last itr.	29	61	80	80	80	80
OpenLSTO	Convergence	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>
	Last itr.	80	80	80	80	80	3

**Table 6.3:** Convergence status, and the iteration at which convergence occurs, for cases C2.1-2.5, also benchmarked against case C1.1



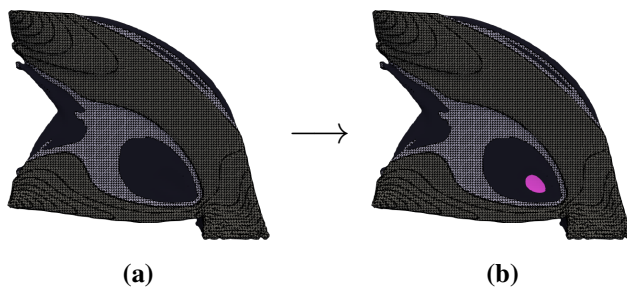
**Figure 6.3:** TopOpt result from the special case 2.5, with a mesh discretization of  $128^2 \times 32 = 524288$  elements. It is very smooth, yet shows no new features, and is therefore only a refinement of the model we see in cases C2.3-2.4, C1.1.



**Figure 6.4:** Delauney smoothing through ParaView of Case 2.1 Ansys model. After a smoothing of the coarse  $20 \times 20 \times 5$  mesh discretization we obtain a solution with no apparent conflicts. Still, we see that the initial density distribution is not satisfactory, low weighted elements are represented in key locations and there are regions where fully weighted elements do not share a connection.

### 6.1.4 The special case of OpenLSTO

Figure 6.5 show how we can interpret the different elevation in the OpenLSTO structure as different topological features. The model is introduced to new features as the level of mesh discretization increases. It creates an interesting question as to why?



**Figure 6.5:** The magenta colored spot seen in (b) can be interpreted as a new topological feature.



## Assessment Three: Domain size dependency

### **Chapter introduction**

Results of particular interest:

Compared to the previous results, can we obtain a better optimum, i.e. was the domain a restricting factor? Is there a need for an increased domain size or can we rely on symmetry?

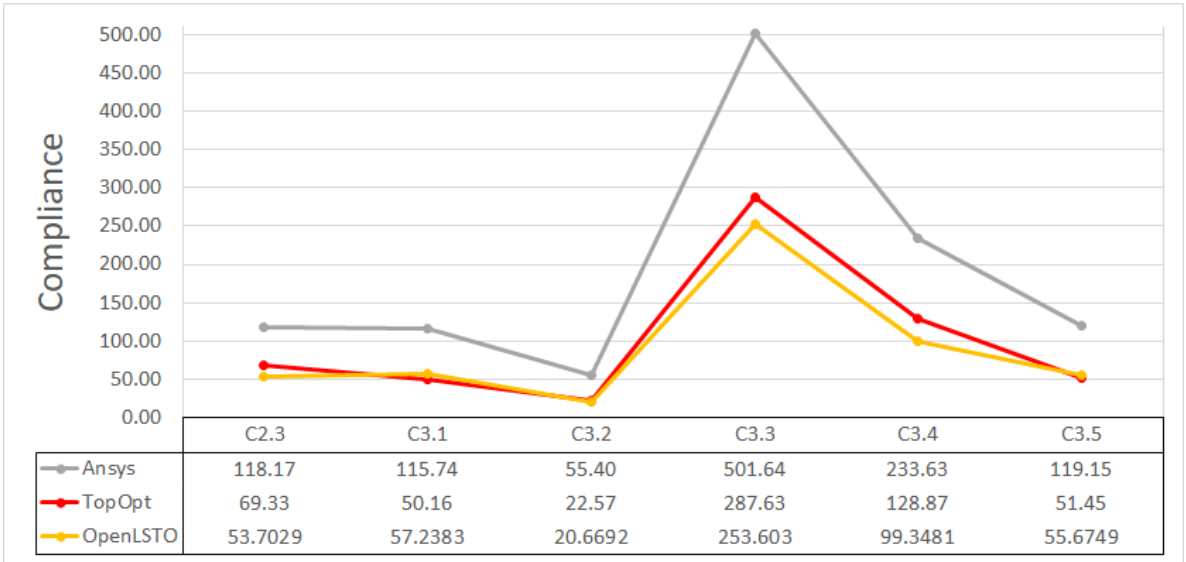
## 7.1 Results C3.1-3.5

### Main findings

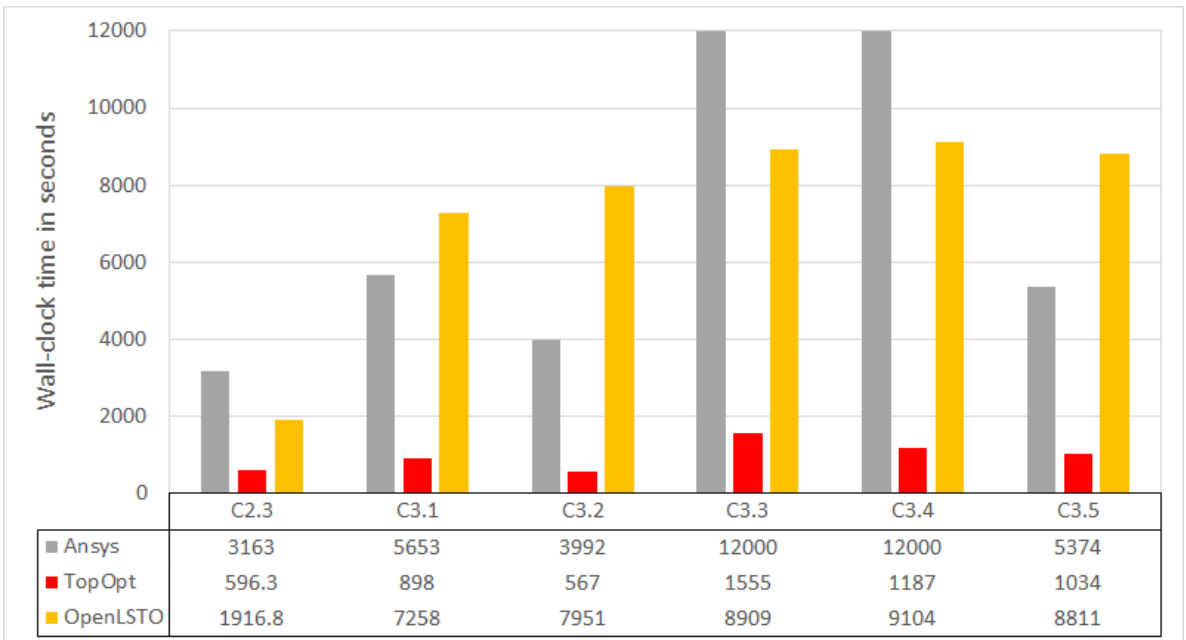
Benchmarked against the previous case C2.3 the results for C3.1 show that for Ansys and TopOpt a model with lower compliance is obtained when the domain size is increased. The opposite is true for OpenLSTO. Contrary to the trend found in assessment one, where models with equal topology have different compliance, they now share the same compliance. The relative difference between Ansys and the other programs is almost exactly twice as big throughout the cases. There is also a trend where bigger loads and boundary surfaces increase the solution time. Ansys and TopOpt both produce symmetric results for cases C3.3-3.4.

### 7.1.1 Presentation of results

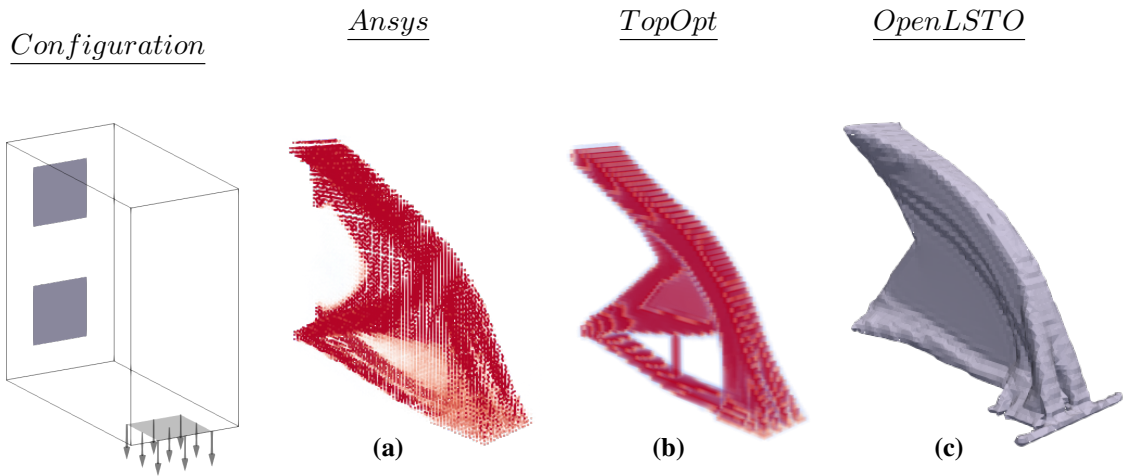
Figure 7.1 show the result for compliance benchmarked against case C2.3 and Figure 7.2 show the total time usage for the cases. Figure 7.3, 7.4, 7.5, 7.6 and 7.7 show the topology result and load configuration. From Figure 7.1 it is observed that for case C3.1 valued against C2.3 Ansys and TopOpt have lower compliance, i.e. a more desirable solution is reached.



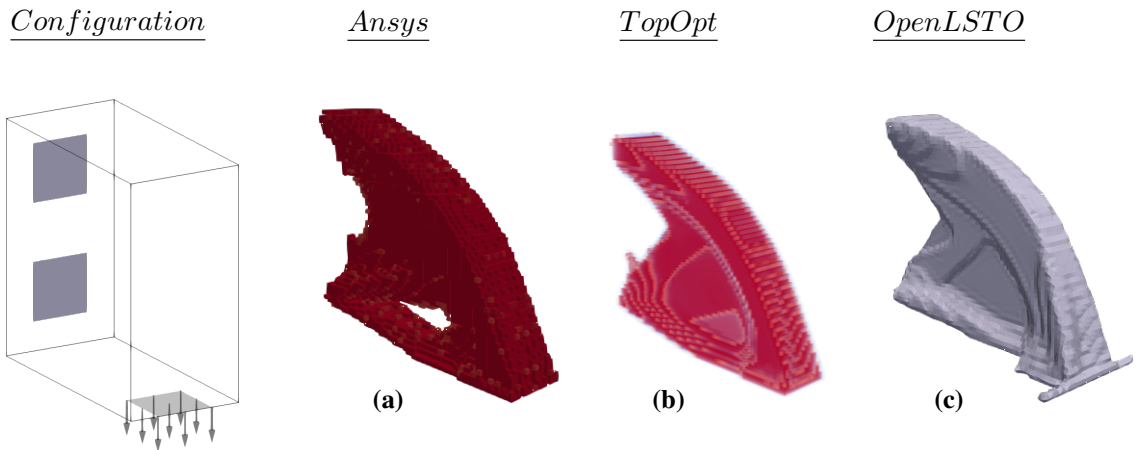
**Figure 7.1:** Compliance across the three programs, Ansys, TopOpt and OpenLSTO. The relative difference between Ansys and the other programs is almost exactly twice as big throughout the cases.



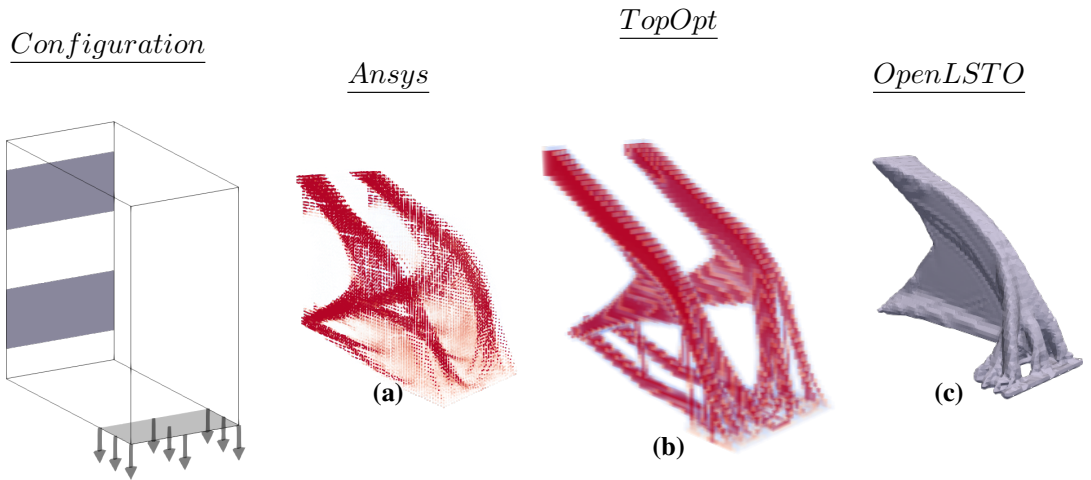
**Figure 7.2:** Total time usage benchmarked against case C2.3. The trend is that bigger loads and boundary surfaces increase the solution time, as seen by C3.3-3.4.



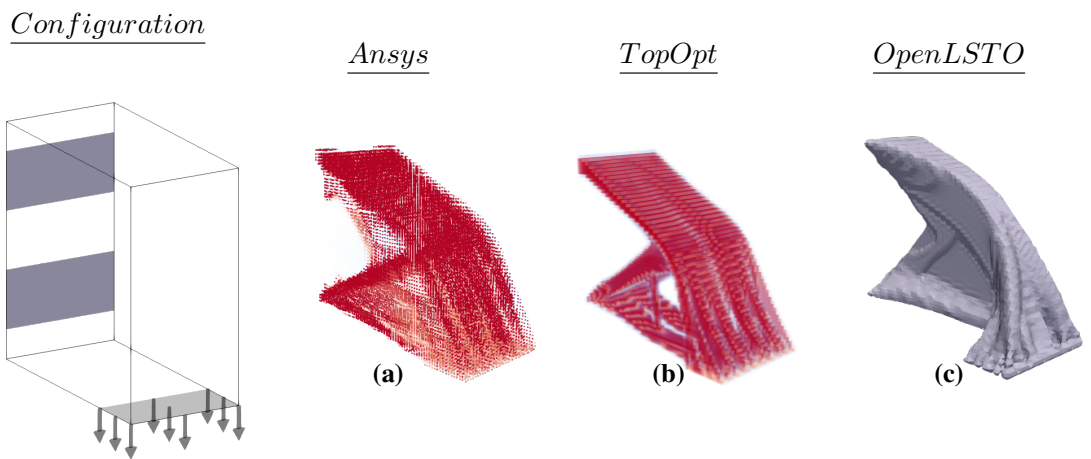
**Figure 7.3:** C3.1: Volume constraint: 7.5%. The topology is similar between all programs, with Ansys and Topopt having one more feature, represented by a hole.



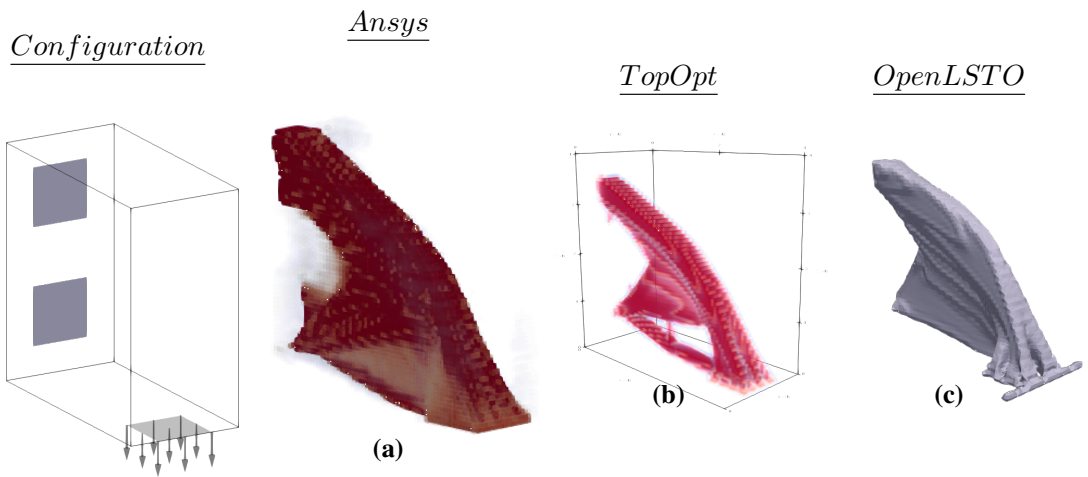
**Figure 7.4:** C3.2: Volume constraint: 15%. TopOpt and OpenLSTO produces the same result, Ansys being very similar.



**Figure 7.5:** Case 3.3: Volume constraint: 7.5%. Ansys and TopOpt show a symmetric representation, while OpenLSTO produces a single solid body, as per usual.



**Figure 7.6:** Case 3.4: Volume constraint: 15%. Slightly different from the case before Ansys and TopOpt show a symmetric representation, while OpenLSTO produces a single solid body again.



**Figure 7.7:** Case 3.5: Volume constraint: 7.5%.

## Conclusion

Our study shows that the different programs default to a preferred topological orientation, which we could imagine. For engineering purposes this is an interesting trend and allows the user to assess certain criteria before choosing the topology optimization tool. Surprisingly, the investigation also shows that topological similarity does not equal the same compliance in a few instances. Two reasons for this behavior is proposed. Different constitutive matrices are used to calculate the stiffness and/or sensitivities are handled differently internally between classes in the programs.

Ansys generally tends to show higher compliance values whilst still showing a similar model shape. A proposed reason for this behavior is the higher node order element used, 20-node versus 8-node.

With higher level of mesh discretization, we saw an improvement of the model, as well as a refinement. The lower limit for obtaining viable results seemed to be the mesh size of  $64^2 \times 16 = 65536$  *elements* for all programs. As an open source program, TopOpt surprised us all by how exceedingly well it performed, having the fastest solution time and showing smooth models repeatedly.

The thorough evaluation between cases C2.3 and C3.1 reveals that for both

Ansys and TopOpt, a model with lower compliance is obtained when the domain size is increased. This correlation may be useful for anyone that want to further explore the programs limits, as well as anyone that want to learn the programs well, and take the advantage of all its possibilities.

## 8.1 A subjective evaluation of the programs

This is a purely subjective evaluation of the *user experience* of the different program suites based on our experience from this project. Presented in order of which one we consider to be the most user friendly to which one is the least user friendly.

**OpenLSTO** – This proved to be the easiest program to get into. The manuals are thorough and the code is presented clearly. It does not rely on other extended libraries to work, only the requirements for compiling (*GCC*) is needed. Also, the example projects can quickly be modified and they are presented in such a way that previous experience with C++ it not a necessity. However, once the complexity of the problem increases the need for initiating new code arises, as was the case for our study. The options presented in the manuals conform to the relatively simple example projects, meaning that finding the limitations of what can be done was accomplished by a bit of trial and error. In version 1.0 there is also restricted optimization options, only compliance objective under a volume constraint is possible when solving in three dimensions.

**TopOpt** – The *Large scale TopOpt based on PETSc* can initially seem a bit daunting. Compared to OpenLSTO it requires a deeper understanding of C++ to use, and the PETSc functions does not make it easier either. That being said, the way the code is presented is so clear that it is practically self explanatory, given some proficiency with C++. The framework has a lot of capabilities, though they are not necessarily easy to utilize. It also relays on having PETSc installed and needs several *Intel* libraries to compile and run.

**Ansys** – We need to make short digression from the main point. Had we used the *Workbench* environment instead, this would with out a doubt been the most user



friendly program on our list. Nevertheless, we chose to use *Mechanical APDL*, were we made an input file due to the extended control of parameters and easy of use non-interactively. The abundance of manuals and guides available, both official and from users, is plentiful. The primary reason why this method is listed third is because of the workload needed to create an input file. For new users this will probably seem unattainable at first glance. There is no template to start from, such as with OpenLSTO and TopOpt, and the *APDL* environment provides a command library that can be set up to reach the same solution with different command configurations. It can create the impression that it is too complicated a task, and that finding a suitable approach is unfeasible. This is of course not true, choosing a fitting approach is straight forward once the environment becomes familiar, yet the initial encounter decreases the user friendliness substantially.

**Abaqus** – Ranked as the least user friendly Abaqus proved extensively troublesome to use non-interactively. We deemed that the workload required to learn how to use Abaqus specific python scripts, needed for setting up and running optimization tasks without the interface, would claim to much our designated time.

## 8.2 Further work

An investigation of the trend that topological similarity does not equal the same compliance, could further unveil the inner working of topology optimization methods. This seems counter intuitive as the goal to this thesis was partially to figure if the four programs would converge to a global optimum solution. A solution like that would most definitely firmly decide the topological features of the model, practically making all the solutions identical. The question we can ask ourselves is: is this the case with other programs and TO methods, and if so, can we realistically talk about a global optimum?

Further research could also include the relation of new features introduced as the level of mesh discretization increases for OpenLSTO, an evaluation of the effect of higher order node elements, and perhaps level set methods in general.



# Bibliography

- [1] Geometric Properties: Size and Shape, 2007. [Online; accessed 1. May 2019].
- [2] Mixing and Reacting Flows, Apr 2019. URL <https://people.umass.edu/debk/Mixing.html>. [Online; accessed 14. May 2019].
- [3] Spatial relations | Nuffield Foundation, May 2019. URL <https://www.nuffieldfoundation.org/key-ideas-teaching-mathematics/spatial-relations>. [Online; accessed 1. May 2019].
- [4] topology | Definition of topology in English by Oxford Dictionaries, May 2019. URL <https://en.oxforddictionaries.com/definition/topology>. [Online; accessed 5. May 2019].
- [5] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using PETS: An easy-to-use, fully parallel, open source topology optimization framework. *Struct. Multidiscip. Optim.*, 51(3):565–572, Mar 2015. ISSN 1615-147X. doi:10.1007/s00158-014-1157-0.
- [6] *ANSYS Elements Reference*. ANSYS Inc., release 12.0 edition, Apr 2009.
- [7] *Theory Reference for the Mechanical APDL and Mechanical Applications*. ANSYS Inc., 12.0 edition, Apr 2009.
- [8] Martin P. Bendsøe and Ole Sigmund. *Topology Optimization* | SpringerLink. Springer, Berlin, Heidelberg, 2004. ISBN 978-3-642-07698-5. doi:10.1007/978-3-662-05086-6.
- [9] Jean Goubault-Larrecq. Non-Hausdorff Topology and Domain Theory by Jean Goubault-Larrecq. *Cambridge Core*, Mar 2013. doi:10.1017/CBO9781139524438.
- [10] Lester O. Hedges. LibSLSM, May 2017. URL <https://github.com/lohedges/slsm>. [Online; accessed 7. Apr 2019].

- 
- [11] Lester O. Hedges, H. Alicia Kim, and Robert L. Jack. Stochastic level-set method for shape optimisation. *J. Comput. Phys.*, 348:82–107, Nov 2017. ISSN 0021-9991. doi:10.1016/j.jcp.2017.07.010.
- [12] Steffen Johnsen. *Structural Topology Optimization: Basic Theory, Methods and Applications*. PhD thesis, NTNU, Institutt for produktutvikling og materialer, 2013. URL <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/241794>.
- [13] William E. Lorensen and Harvey E. Cline. *Marching cubes: A high resolution 3D surface construction algorithm*, volume 21. ACM, Aug 1987. ISBN 978-0-89791-227-3. doi:10.1145/37401.37422.
- [14] *Programmer Manual for OpenLSTO v1.0*. M2DO Multiscale Multiphysics Design Optimization Laboratory, Aug 2018. URL <http://m2do.ucsd.edu/software/>.
- [15] *Theoretical Background for OpenLSTO v1.0*. M2DO Multiscale Multiphysics Design Optimization Laboratory, Aug 2018. URL <http://m2do.ucsd.edu/software/>.
- [16] *Tutorial for OpenLSTO v1.0*. M2DO Multiscale Multiphysics Design Optimization Laboratory, Aug 2018. URL <http://m2do.ucsd.edu/software/>.
- [17] G. I. N. Rozvany. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. *Struct. Multidiscip. Optim.*, 21(2):90–108, Apr 2001. ISSN 1615-147X. doi:10.1007/s001580050174.
- [18] A. Saltelli, S. Tarantola, and F. Campolongo. Sensitivity Analysis as an Ingredient of Modeling. *Statistical Science*, 15(4):377–395, Nov 2000. ISSN 0883-4237. doi:10.1214/ss/1009213004.
- [19] J. A. Sethian. A Fast Marching Level Set Method for Monotonically Advancing Fronts. *PNAS*, 93(4):1591–1595, Feb 1996. ISSN 0027-8424. doi:10.2307/38628.
- [20] M. Stolpe and K. Svanberg. On the trajectories of penalization methods for topology optimization. *Struct. Multidiscip. Optim.*, 21(2):128–139, Apr 2001. ISSN 1615-147X. doi:10.1007/s001580050177.
- [21] Krister Svanberg. The method of moving asymptotes - A new method for structural optimization. *Int. J. Numer. Methods Eng.*, 24(2):359–373, Feb 1987. ISSN 1097-0207. doi:10.1002/nme.1620240207.
- [22] Matthew Tomlin and Jonathan Meyer, Apr 2013. URL <https://www.mmsonline.com/cdn/cms/uploadedFiles/Topology-Optimization-of-an-Additive-Layer-Manufactured-Aerospace-Part.pdf>. [Online; accessed 15. May 2019].
-

- 
- [23] Luzhong Yin and Wei Yang. Optimality criteria method for topology optimization under multiple constraints, May 2019. ISSN 0045-7949. URL [https://www.academia.edu/4332031/Optimality\\_criteria\\_method\\_for\\_topology\\_optimization\\_under\\_multiple\\_constraints](https://www.academia.edu/4332031/Optimality_criteria_method_for_topology_optimization_under_multiple_constraints). [Online; accessed 13. May 2019].

---

---

# Appendix

## 8.3 Appendix B1: Abaqus results

### 8.3.1 Listing of convergence plots for Abaqus

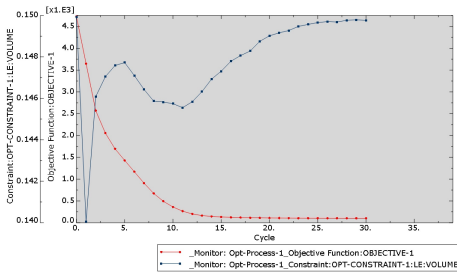


Figure 8.1: Case 1.1

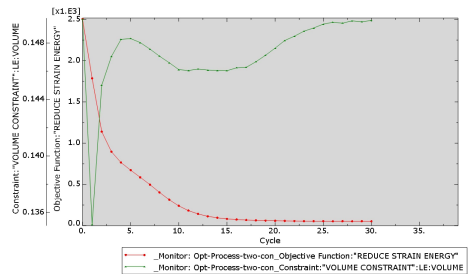


Figure 8.4: Case 1.4

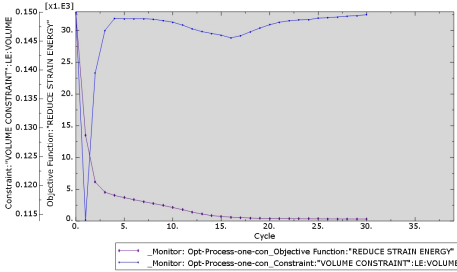


Figure 8.2: Case 1.2

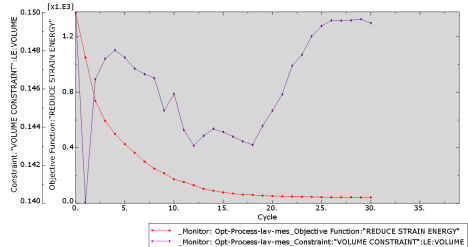


Figure 8.5: Case 2.1

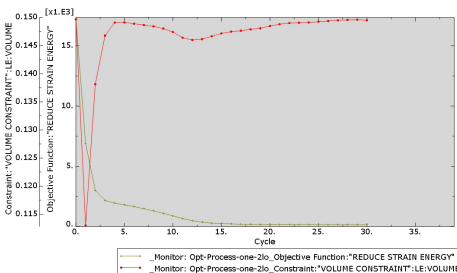


Figure 8.3: Case 1.3

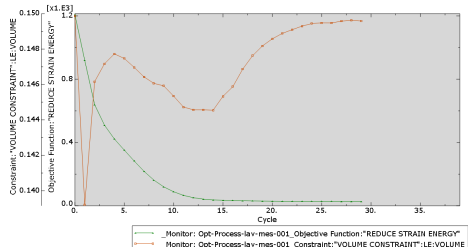
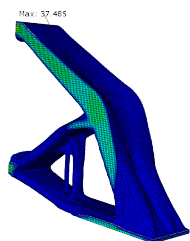


Figure 8.6: Case 2.3

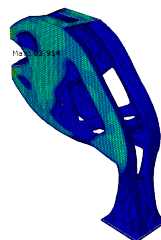


---

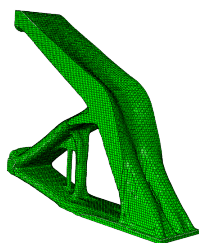
### 8.3.2 Listing of strain, mesh and model plots for Abaqus



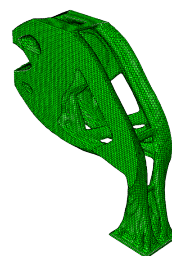
**Figure 8.7:** Case 1.1



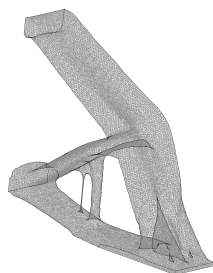
**Figure 8.10:** Case 1.2



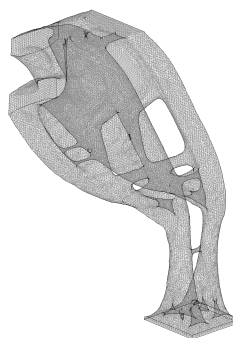
**Figure 8.8:** Case 1.1



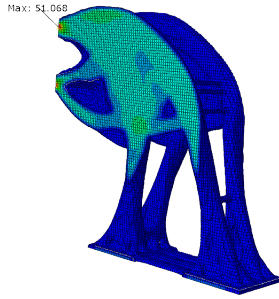
**Figure 8.11:** Case 1.2



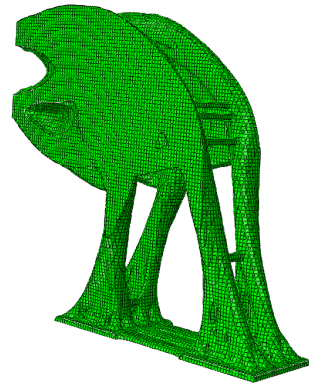
**Figure 8.9:** Case 1.1



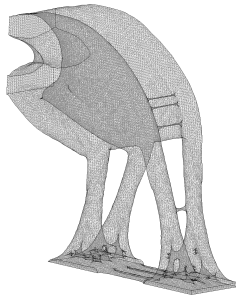
**Figure 8.12:** Case 1.2



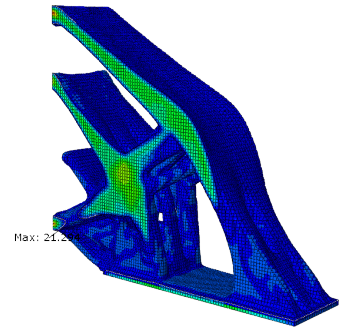
**Figure 8.13:** Case 1.3



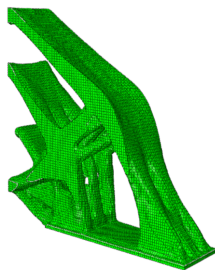
**Figure 8.14:** Case 1.3



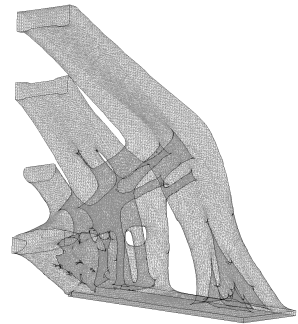
**Figure 8.15:** Case 1.3



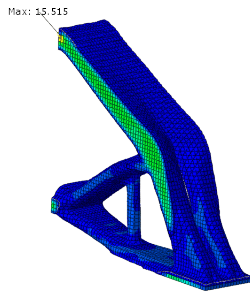
**Figure 8.16:** Case 1.4



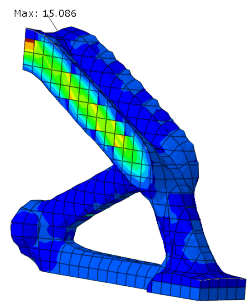
**Figure 8.17:** Case 1.4



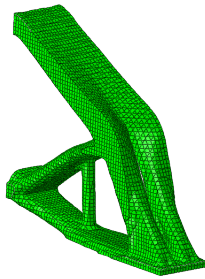
**Figure 8.18:** Case 1.4



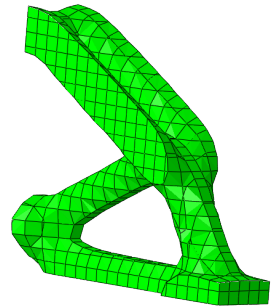
**Figure 8.19:** Case 2.1



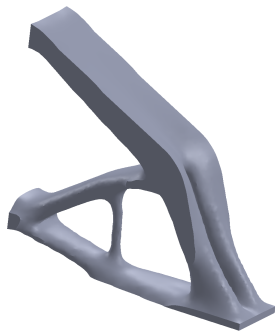
**Figure 8.22:** Case 2.3



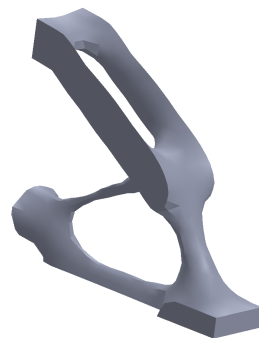
**Figure 8.20:** Case 2.1



**Figure 8.23:** Case 2.3



**Figure 8.21:** Case 2.1



**Figure 8.24:** Case 2.3

## 8.4 Appendix B2: Ansys results

### 8.4.1 Listing of convergence plots for Ansys

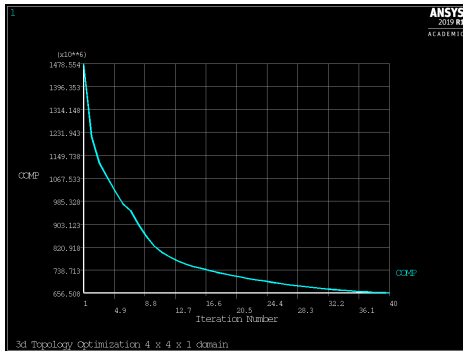


Figure 8.25: Case 1.1 Compliance

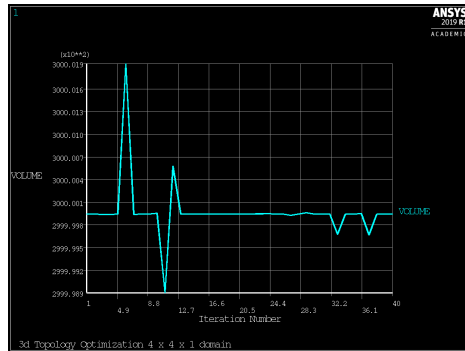


Figure 8.27: Case 1.1 Volume

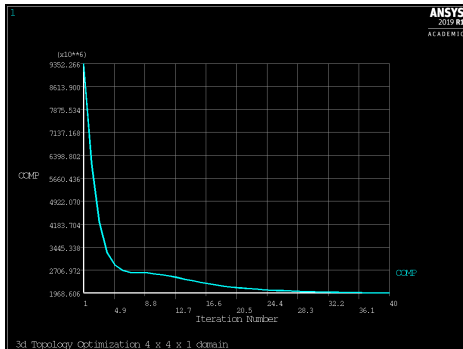


Figure 8.26: Case 1.2 Compliance

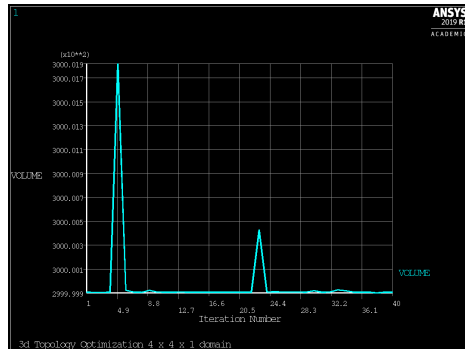
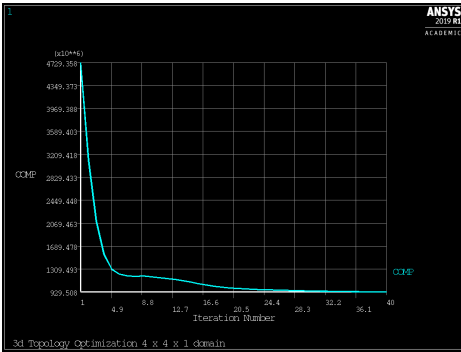
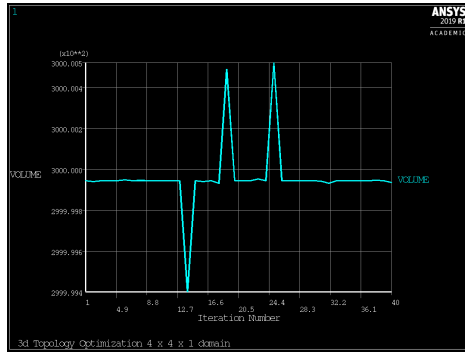


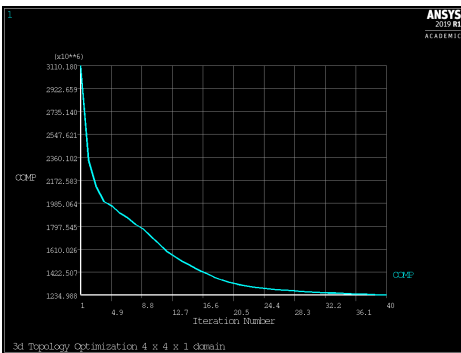
Figure 8.28: Case 1.2 Volume



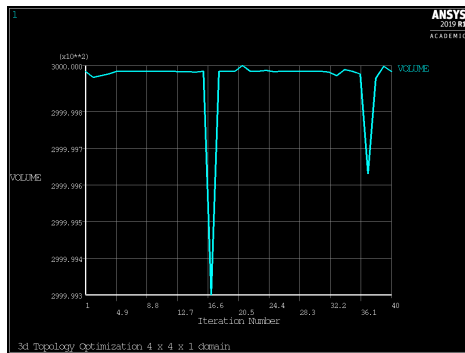
**Figure 8.29:** Case 1.3 Compliance



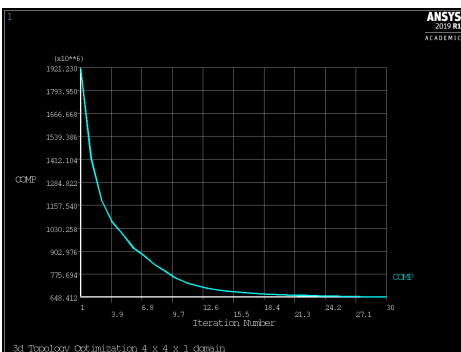
**Figure 8.32:** Case 1.3 Volume



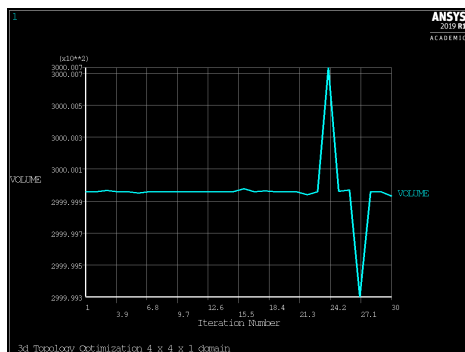
**Figure 8.30:** Case 1.4 Compliance



**Figure 8.33:** Case 1.4 Volume



**Figure 8.31:** Case 1.5 Compliance



**Figure 8.34:** Case 1.5 Volume

## 8.4.2 Listing of model plots for Ansys

### Case 1.1

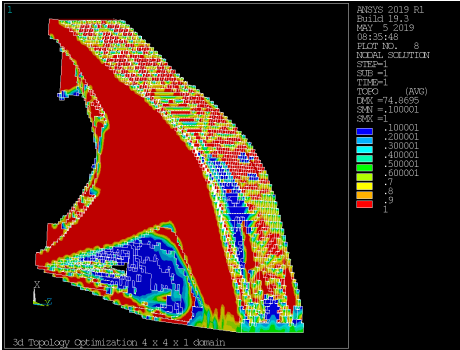


Figure 8.35: Case 1.1  $p=0.1$

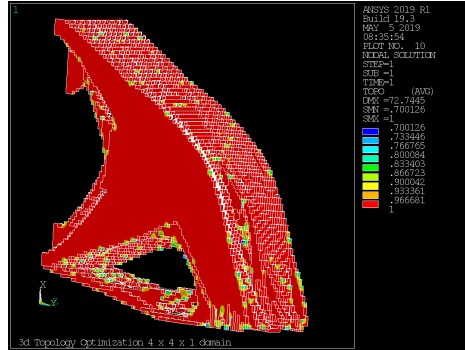


Figure 8.37: Case 1.1  $p=0.7$

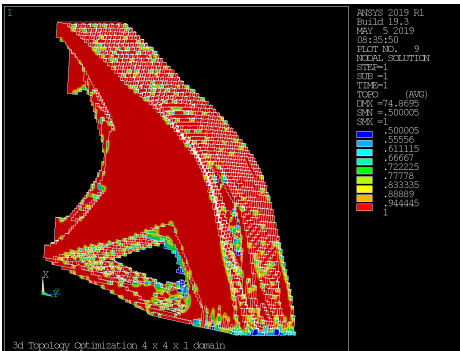


Figure 8.36: Case 1.1  $p=0.5$

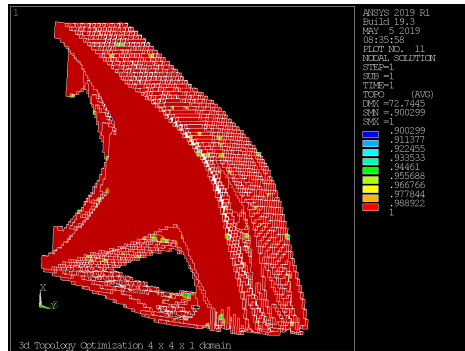


Figure 8.38: Case 1.1  $p=0.9$

Case 1.2

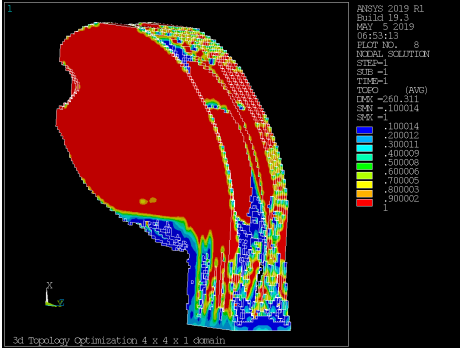


Figure 8.39: Case 1.2  $p=0.1$

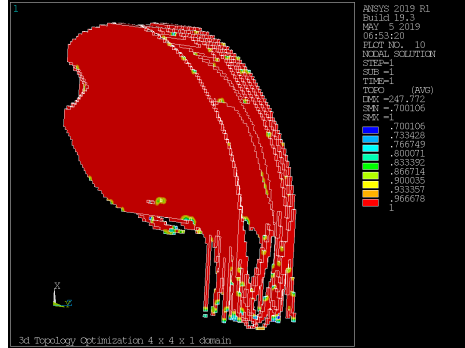


Figure 8.41: Case 1.2  $p=0.7$

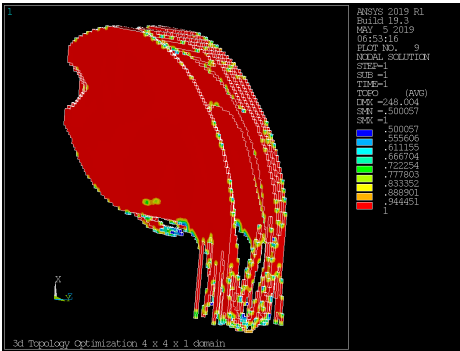


Figure 8.40: Case 1.2  $p=0.5$

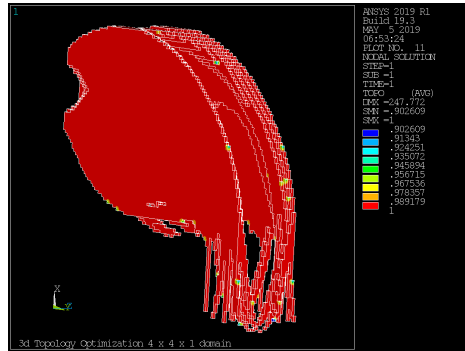


Figure 8.42: Case 1.2  $p=0.9$

Case 1.3

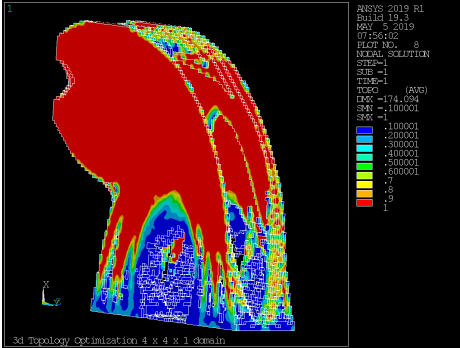


Figure 8.43: Case 1.3  $p=0.1$

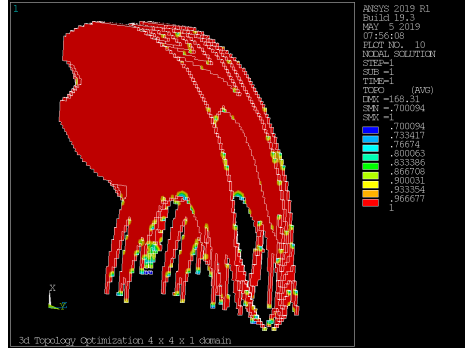


Figure 8.45: Case 1.3  $p=0.7$

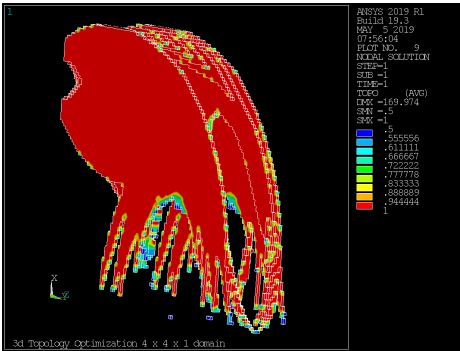


Figure 8.44: Case 1.3  $p=0.5$

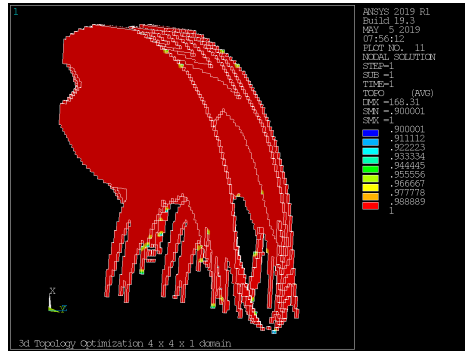


Figure 8.46: Case 1.3  $p=0.9$



Case 1.4

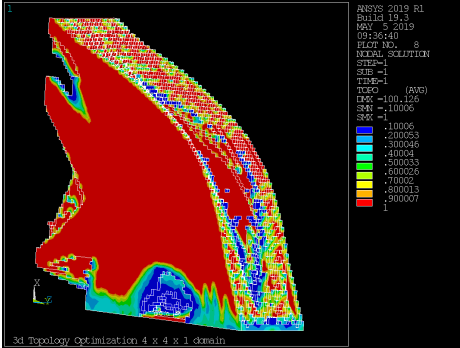


Figure 8.47: Case 1.4  $p=0.1$

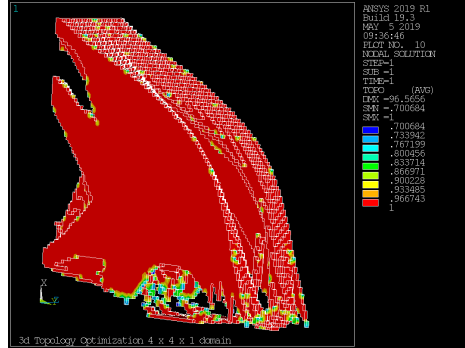


Figure 8.49: Case 1.4  $p=0.7$

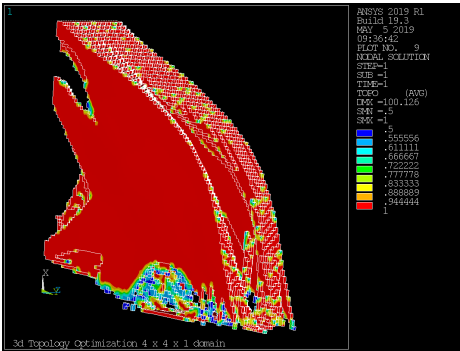


Figure 8.48: Case 1.4  $p=0.5$

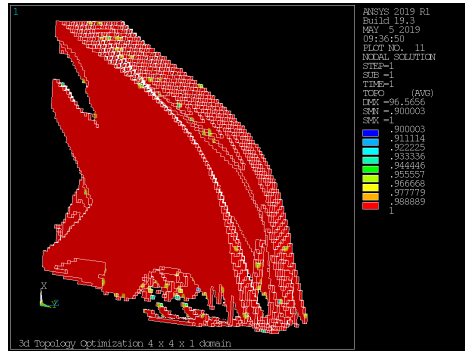
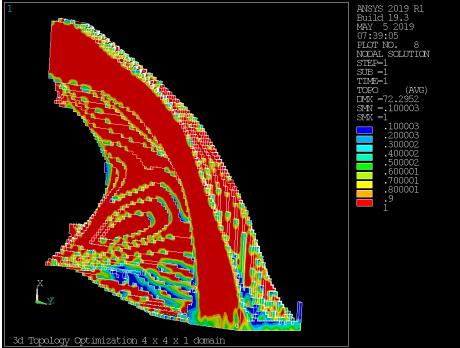
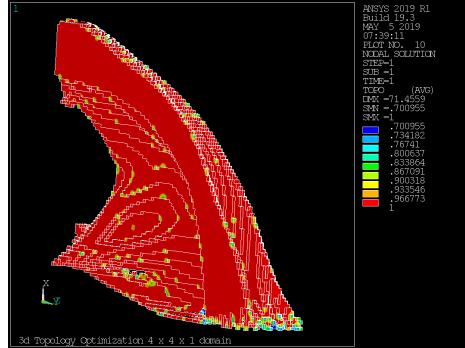


Figure 8.50: Case 1.4  $p=0.9$

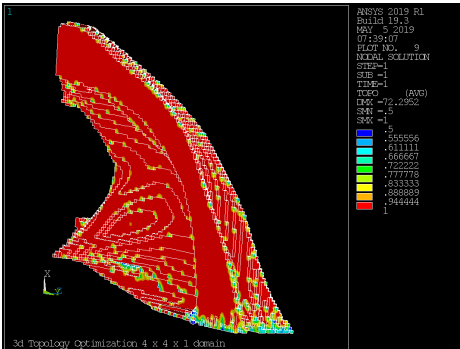
**Case 1.5**



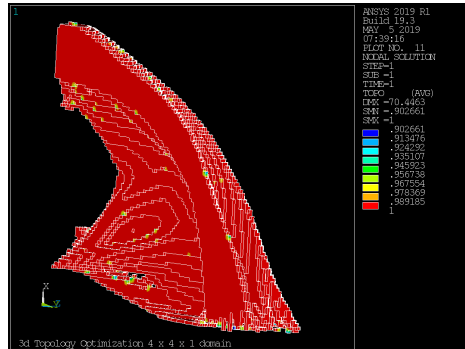
**Figure 8.51: Case 1.5  $p=0.1$**



**Figure 8.53: Case 1.5  $p=0.7$**



**Figure 8.52: Case 1.5  $p=0.5$**



**Figure 8.54: Case 1.5  $p=0.9$**

---

## 8.5 Appendix B3: TopOpt results

### 8.5.1 Listing of convergence plots for TopOpt

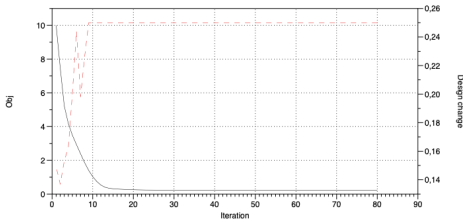


Figure 8.55: Case 1.1

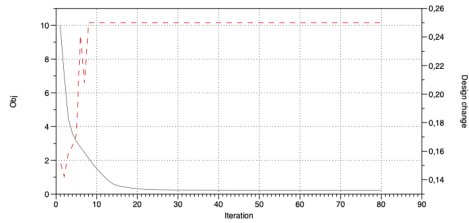


Figure 8.58: Case 1.4

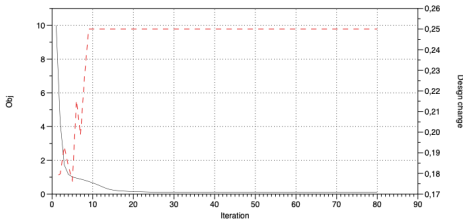


Figure 8.56: Case 1.2

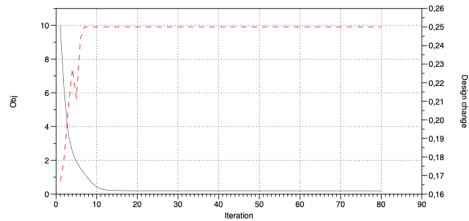


Figure 8.59: Case 1.5

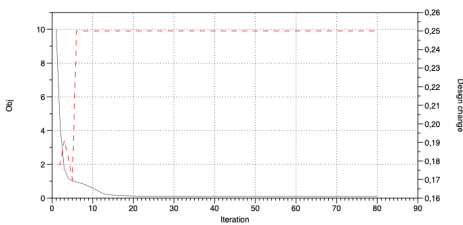


Figure 8.57: Case 1.3

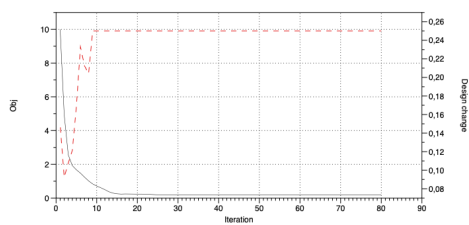
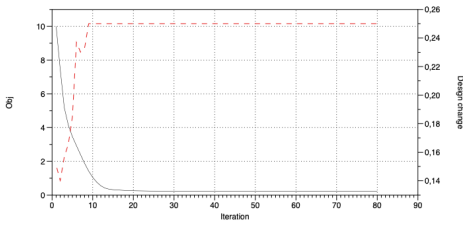
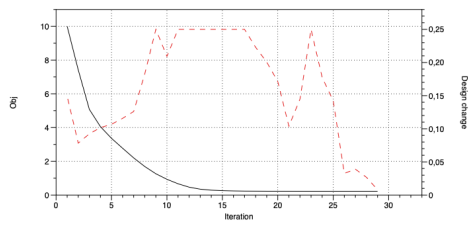


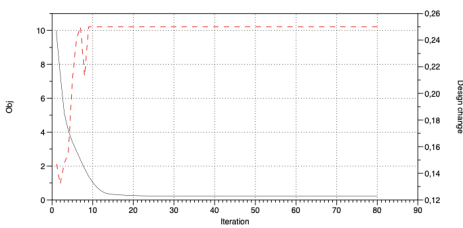
Figure 8.60: Case 1.6



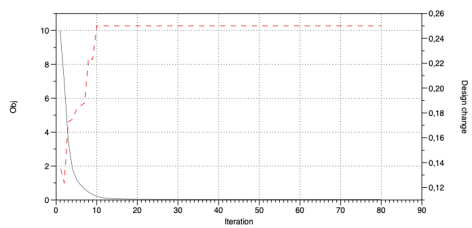
**Figure 8.61:** Case 2.1



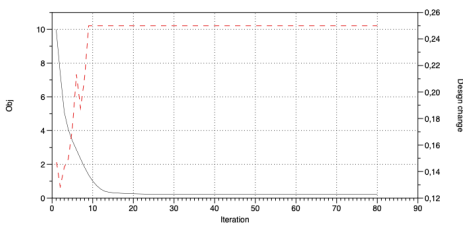
**Figure 8.65:** Case 2.5



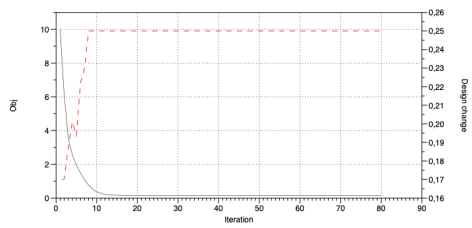
**Figure 8.62:** Case 2.2



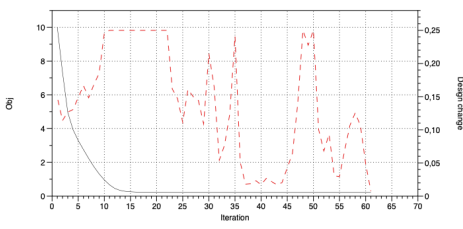
**Figure 8.66:** Case 3.1



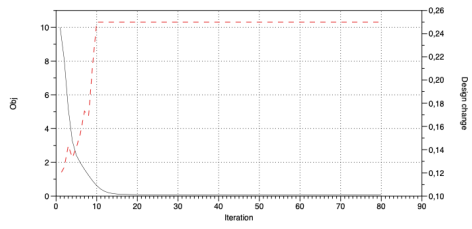
**Figure 8.63:** Case 2.3



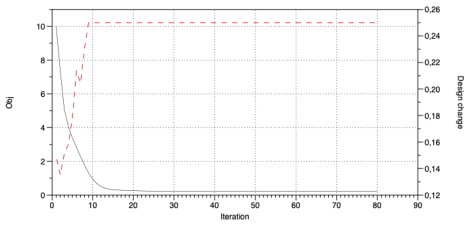
**Figure 8.67:** Case 3.2



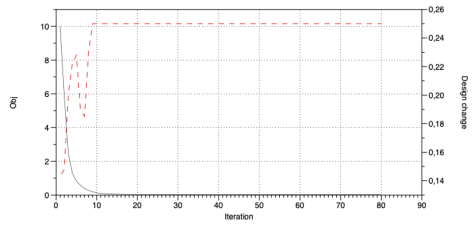
**Figure 8.64:** Case 2.4



**Figure 8.68:** Case 3.3



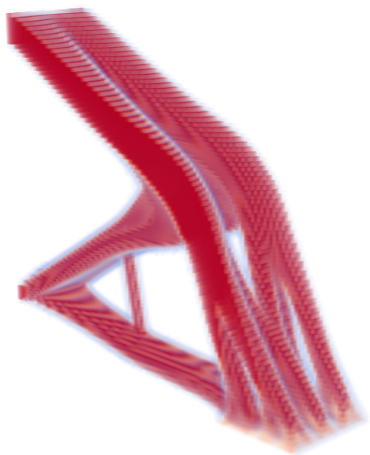
**Figure 8.69:** Case 3.4



**Figure 8.70:** Case 3.5

---

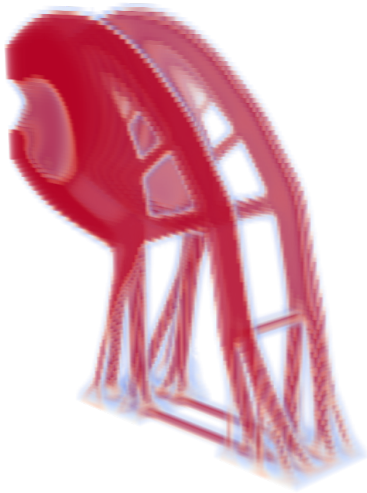
## 8.5.2 Listing of model plots for TopOpt



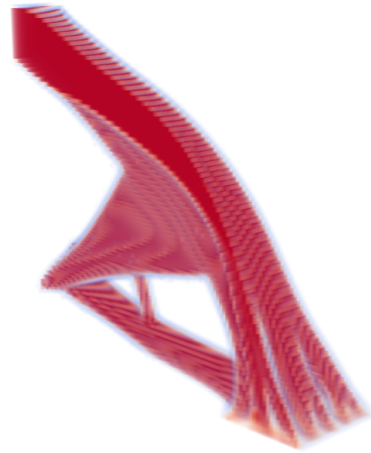
**Figure 8.71:** Case 1.1



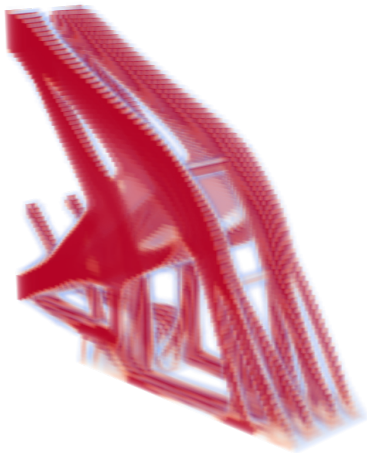
**Figure 8.72:** Case 1.2



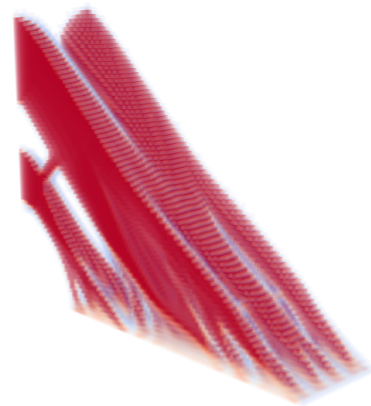
**Figure 8.73:** Case 1.3



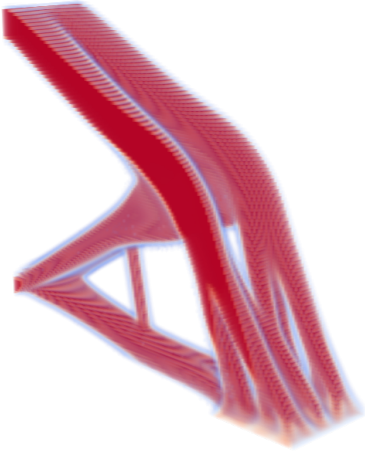
**Figure 8.75:** Case 1.5



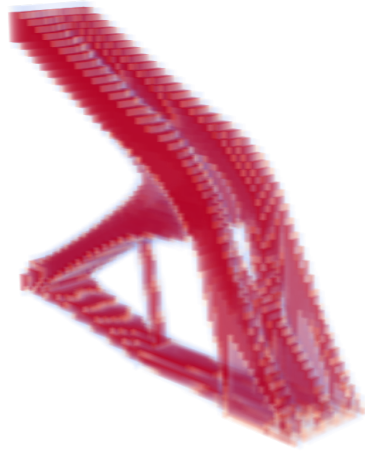
**Figure 8.74:** Case 1.4



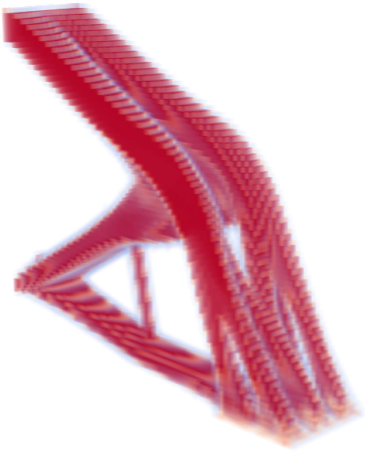
**Figure 8.76:** Case 1.6



**Figure 8.77:** Case 2.1



**Figure 8.79:** Case 2.3

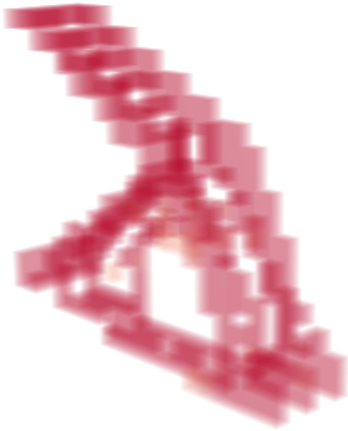


**Figure 8.78:** Case 2.2

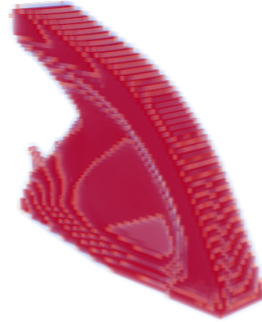


**Figure 8.80:** Case 2.4

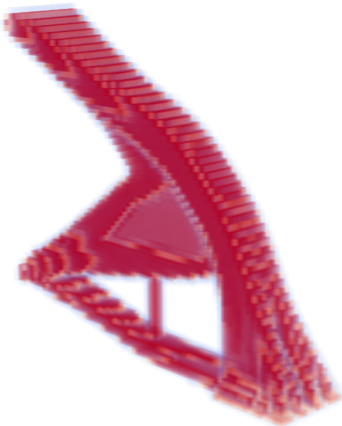




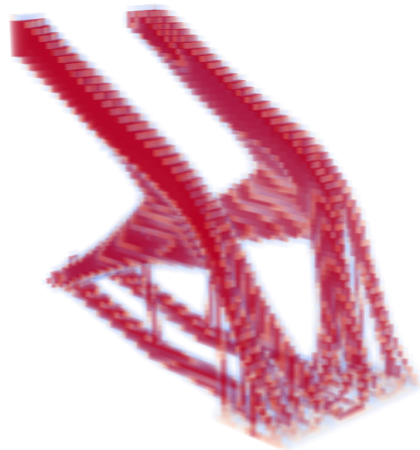
**Figure 8.81:** Case 2.5



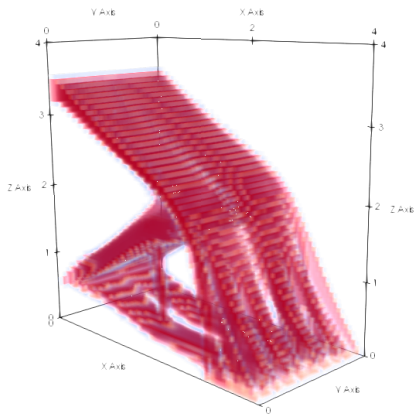
**Figure 8.83:** Case 3.2



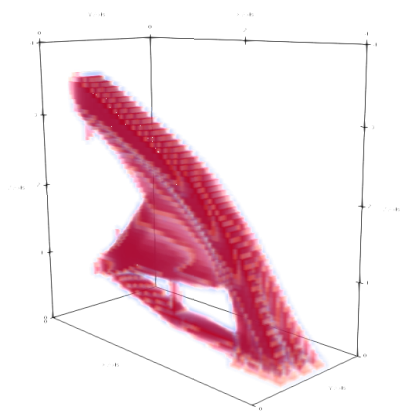
**Figure 8.82:** Case 3.1



**Figure 8.84:** Case 3.3



**Figure 8.85:** Case 3.4



**Figure 8.86:** Case 3.5

---

## 8.6 Appendix B4: OpenLSTO results

### 8.6.1 Listing of convergence plots for OpenLSTO

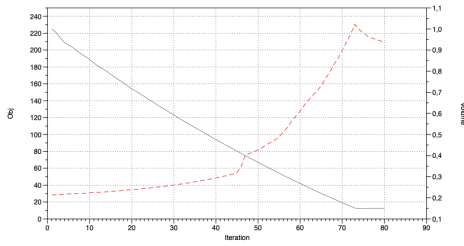


Figure 8.87: Case 1.1

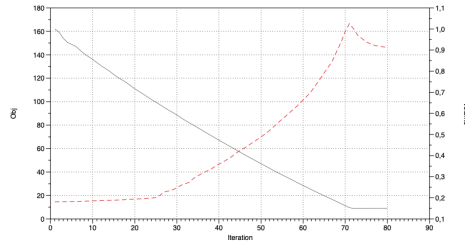


Figure 8.90: Case 1.4

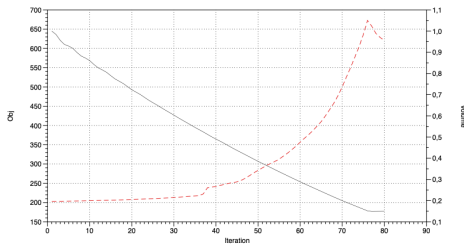


Figure 8.88: Case 1.2

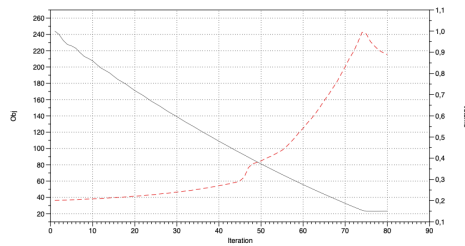


Figure 8.91: Case 1.5

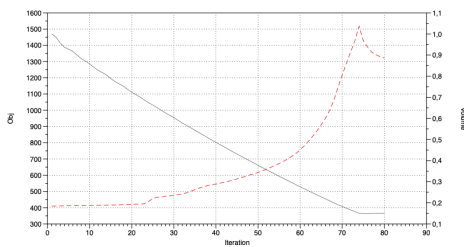


Figure 8.89: Case 1.3

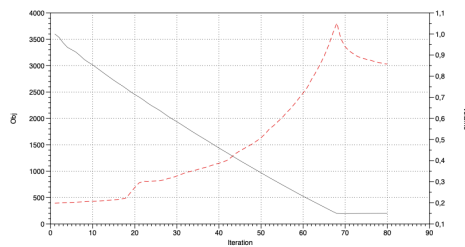
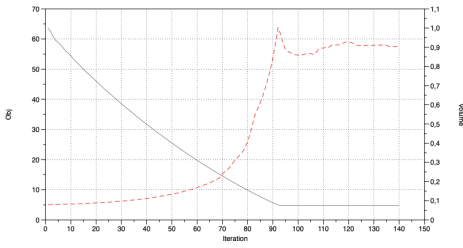
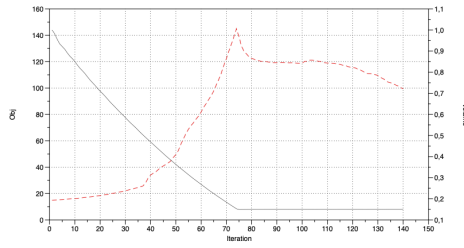


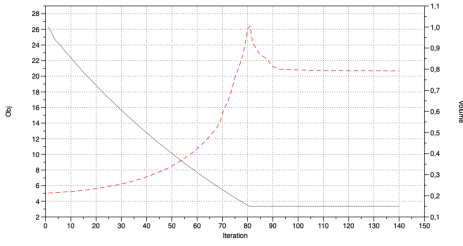
Figure 8.92: Case 1.6



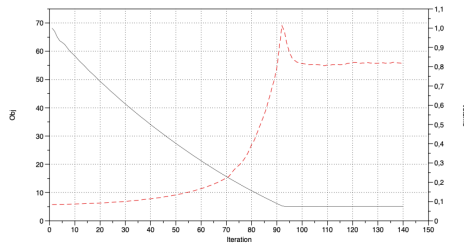
**Figure 8.93:** Case 3.1



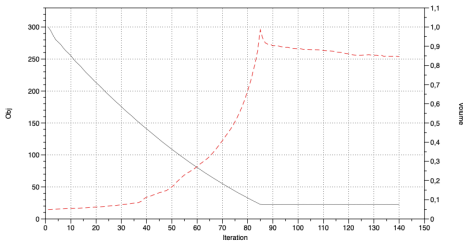
**Figure 8.96:** Case 3.4



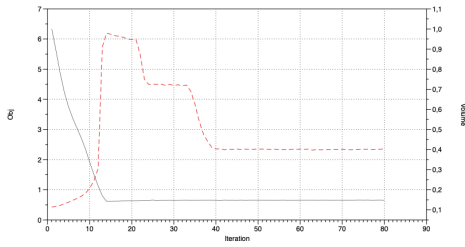
**Figure 8.94:** Case 3.2



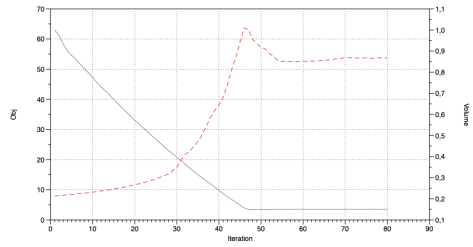
**Figure 8.97:** Case 3.5



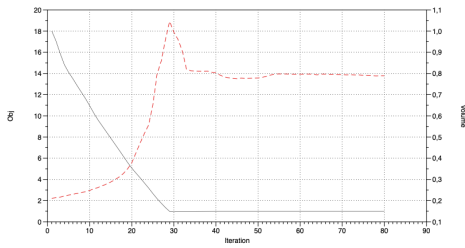
**Figure 8.95:** Case 3.3



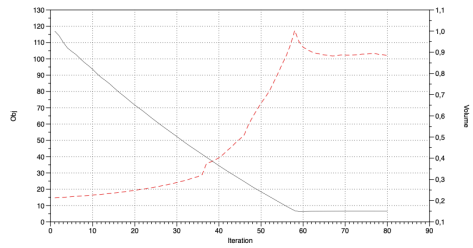
**Figure 8.98:** Mesh size 20



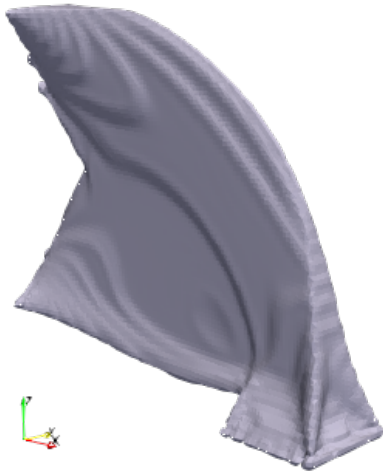
**Figure 8.100:** Mesh size 64



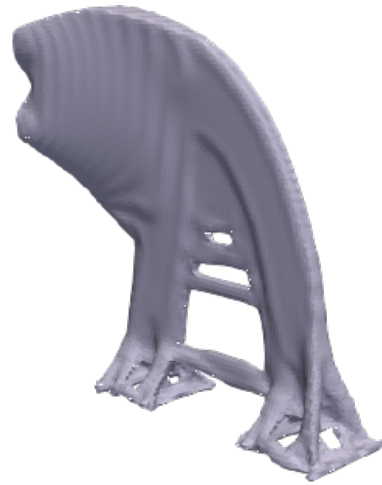
**Figure 8.99:** Mesh size 40



**Figure 8.101:** Mesh size 80



**Figure 8.102:** Case 1.1



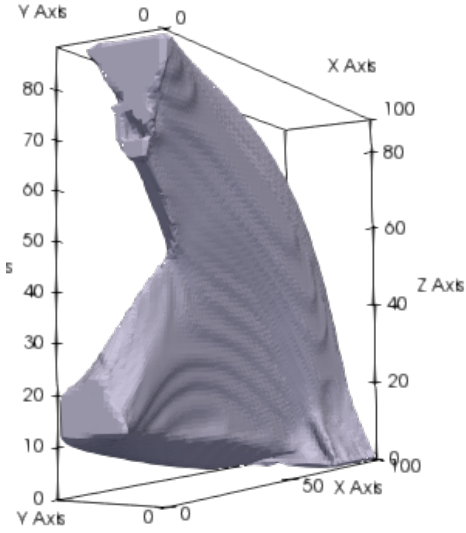
**Figure 8.104:** Case 1.3



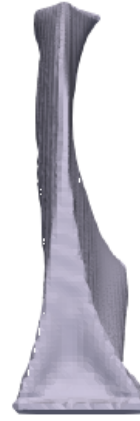
**Figure 8.103:** Case 1.2



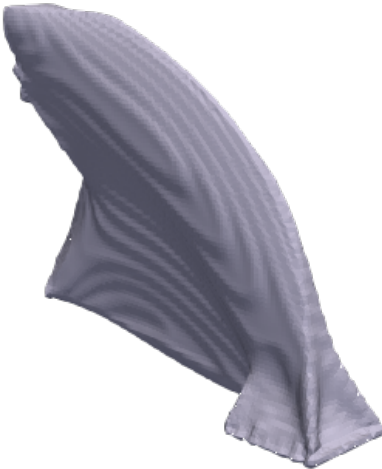
**Figure 8.105:** Case 1.4



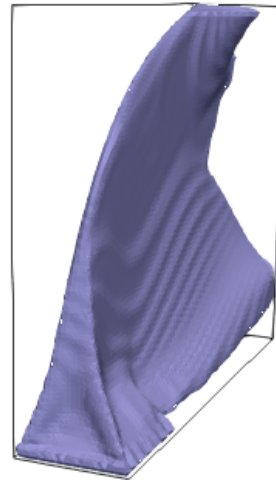
**Figure 8.106:** Case 1.5



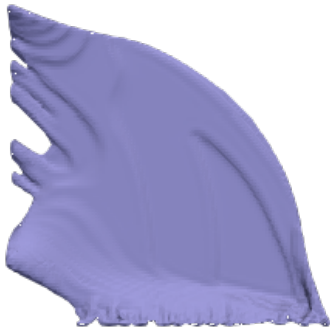
**Figure 8.108:** Case 1.5



**Figure 8.107:** Case 1.5



**Figure 8.109:** Case 1.5



**Figure 8.110:** Case 1.6



**Figure 8.111:** Case 1.6



---

## 8.7 Appendix C:

### 8.7.1 Ansys, APDL input file

For more information see the ANSYS Mechanical APDL Command Reference and Chapter 2: Topological Optimization from Mechanika2.

```
1 /batch
2 !tell ansys this is a batch file
3 /filename , Box
4 /Title , 3d Topology Optimization 4 x 4 x 1 domain
5 !Give the file the name Box
6 /units ,MPA
7 !parameters L-beam geometri
8 xval=200 ! lenght in x direction
9 yval=200 ! lenght in y direction
10 zval=50 ! lenght in z direction
11 !parameters loads/boundries
12 xboundary=175
13 yboundary=0
14 xload=0
15 yload=150
16
17 /prep7
18 !keypoints for square in xy-plane
19 k,1,0,0,
20 k,2,xval,0,
21 k,3,xval,yval,
22 k,4,0,yval,
23
24 l,1,2 ! #line 1
25 l,2,3 ! #line 2
26 l,3,4 ! #line 3
27 l,4,1 ! #line 4
28
29 allsel , all , line
30 al , all , ! Area #1
31
32 allsel , all , area
33 asel , r , area , , 1 , 1 ,
34 et , 2 , 183
35 esize , 2 ,
36 amesh , 1 , 1 ,
37 asel , all
38 CM , remove , area
```

---

```

39
40 allsel , all , area
41 asel , r , area , , 1 , 1 ,
42 vext , 1 , 1 , , 0 , 0 , zval
43 allsel , all , volu
44 et , 1 , 95 ,
45 vsweep , 1 , 1 ,
46 aclear , remove ,
47
48 MP,DENS,1,7.85e-09, ! tonne mm-3
49 MP,ALPX,1,1.2e-05, ! C-1
50 MP,C,1,434000000, ! mm2 s-2 C-1
51 MP,KXX,1,60.5, ! tonne mm s-3 C-1
52 MP,RSVX,1,0.00017, ! ohm mm
53 MP,EX,1,200000, ! tonne s-2 mm-1, Elastic moduli
54 MP,NUXY,1,0.3, ! Minor Poisson's ratio
55 MP,MURX,1,10000,
56
57 allsel
58 nsel , r , loc , x , 125 , 175 ,
59 nsel , r , loc , y , -0.01 , 0.01
60 nsel , r , loc , z , 0 , zval
61 d , all , all
62
63 allsel
64 nsel , r , loc , x , 25 , 75 ,
65 nsel , r , loc , y , -0.01 , 0.01
66 nsel , r , loc , z , 0 , zval
67 d , all , all
68
69 FORCE=(-6000)
70 allsel
71 nsel , r , loc , x , -0.01 , 0.01
72 nsel , r , loc , y , yload , yval ,
73 nsel , r , loc , z , 0 , zval ,
74 f , all , fx , FORCE ,
75 allsel
76
77 allsel
78 nsel , r , loc , x , -0.01 , 0.01
79 nsel , r , loc , y , 50 , 100 ,
80 nsel , r , loc , z , 0 , zval ,
81 !f , all , fx , FORCE ,
82 allsel
83
84 lswrite , 1
85 fdel , all
86 /solu

```

---

---

```

87 eqslv ,pcg
88 tocomp ,comp ,single ,1 ! Defines compliance function for 1 load case
89 !TOCOMP, Refname , Type , NUMLC , LCARR – Defines single or multiple
    compliance as the TO function .
90 TOVAR ,comp ,obj ! Sets compliance as the objective with a
91 TOVAR ,VOLUME ,con , ,85 ! volume constraint with an upper limit 15% of max
    volume
92 !TOVAR, Refname , Type , LOWER , UPPER , Boundtype – Specifies the
    objective and constraints for TO problem .
93 TOTYPE ,oc
94 !totype , type – set TO method as either OC or SCP
95 todef , 0.0005 ! sets convergence accuracy
96
97 /angle , all ,130 ,ym ,1
98 /angle , all ,0 ,xm ,1
99 /angle , all ,95 ,zm ,1
100 /SHOW ,png ! Put graphics in a file (remove if interactive)
101 /dscale , ,off
102 TOLOOP ,40 ,0 ! Optimization loop toloop , Max # itr , display solution plot
    for each itr (yes)/(no) as 1 / 0
103
104 TOGRAPH , obj
105 TOGRAPH , con
106 TOSTAT
107 /post1
108 topplot ,0
109 /edge , ,0
110
111 !section view
112 allsel
113 WPOFFS ,0 ,0 ,10 ! Offset the working plane for cross–section view
114 WPROTA ,0 ,0 ,180 ! Rotate the working plane
115 /CPLANE ,1 ! Cutting plane defined to use the WP
116 /TYPE ,1 ,5
117 topplot ,0
118
119 WPOFFS ,0 ,0 ,–10 ! Offset the working plane for cross–section view
120 /CPLANE ,1 ! Cutting plane defined to use the WP
121 /TYPE ,1 ,5
122 topplot ,0
123
124 WPOFFS ,0 ,0 ,–10 ! Offset the working plane for cross–section view
125 /CPLANE ,1 ! Cutting plane defined to use the WP
126 /TYPE ,1 ,5
127 topplot ,0
128
129 WPOFFS ,0 ,0 ,–10 ! Offset the working plane for cross–section view
130 /CPLANE ,1 ! Cutting plane defined to use the WP

```

---

---

```

131 /TYPE,1,5
132 PLNSOL, topo
133 WPOFFS,0,0,50
134 ! run 1 with 0.1 dens
135 ETABLE, EDENS, topo
136 ESEL, S, ETAB, EDENS, 0.1, 1.0
137 NSLE, r, all
138 topplot, 0
139
140 !run 2 with 0.5 dens
141 ETABLE, EDENS, topo
142 ESEL, S, ETAB, EDENS, 0.5, 1.0
143 NSLE, r, all
144 PLNSOL, topo
145 !run with 0.7 dens
146 ETABLE, EDENS, topo
147 ESEL, S, ETAB, EDENS, 0.7, 1.0
148 NSLE, r, all
149 topplot, 0
150 !run 4 with 0.9 dens
151 ETABLE, EDENS, topo
152 ESEL, S, ETAB, EDENS, 0.9, 1.0
153 NSLE, r, all
154 topplot, 0
155 /reset
156 /dscale, , off
157 allsel
158 /angle, all, 90, zm, 1
159 /angle, all, 180, xm, 1
160 /angle, all, -90, ym, 1
161 topplot, 0
162 /angle, all, 90, ym, 1
163 /angle, all, 90, xm, 1
164 topplot, 0
165
166 allsel
167
168 *CFOPEN, modeldens, csv, , append
169 *GET, num_elem_, elem_, 0, COUNT !Get the number of Elements
170 *GET, elem_, elem_, 0, NUM, MIN !Get label of the first Element
171 etable, edens, topo
172 etable, estress, s, eqv
173
174 *DO, i, 1, num_elem_, 1
175     ! Define some parameters *GET, Par, Entity, ENTNUM, Item1, IT1NUM, Item2
176     , IT2NUM
177     *GET, nx_, elem_, elem_, cent, X
178     *GET, ny_, elem_, elem_, cent, Y

```

---

---

```
178 *GET, nz_ , elem , elem_ , cent , Z
179 *GET, dens , elem , elem_ , etab , edens
180 *GET, stress , elem , elem_ , etab , estress
181
182 *VWRITE, elmm_ , nx_ , ny_ , nz_ , dens , stress
183 (E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 , ' , ' , E10.3 )
184 ! select the next element
185 *GET, elem_ , elem , elem_ , NXTH
186
187 *ENDDO
188 *CFCLOSE
189
190 *GET, TOPCV, TOPO, , CONV ! If TOPCV = 1 (converged)
191 *GET, ECOMP, TOPO, , COMP ! ECOMP = Compliance Energy
192 STAT
193 save , Box , db ! Topology optimization database
194 /eof
```