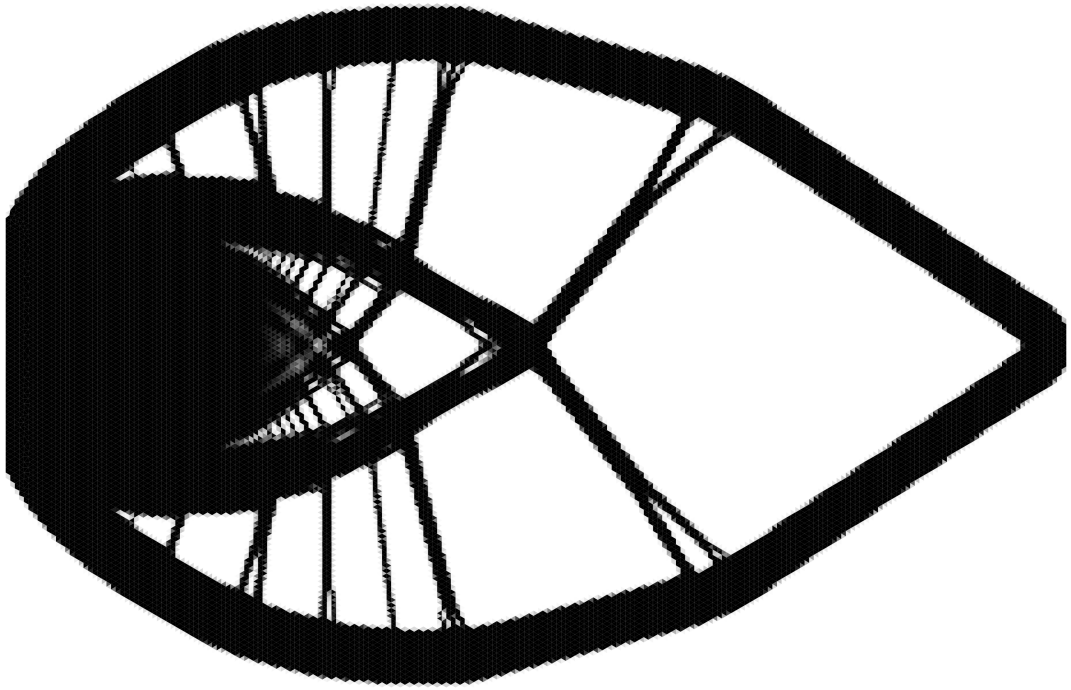


PyTOpt

A Nonlinear Topology Optimisation Program in Python

Master's thesis in Applied Mechanics

Daniel Pettersson
Erik Säterskog

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021:44

PyTOpt

A Nonlinear Topology Optimisation Program in Python

DANIEL PETTERSSON
ERIK SÄTERSKOG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2021

PyTOpt
A Nonlinear Topology Optimisation Program in Python
DANIEL PETTERSSON
ERIK SÄTERSKOG

Supervisor: Knut Andreas Meyer, Department of Industrial and Materials Science
Supervisor: Magnus Ekh, Department of Industrial and Materials Science
Examiner: Jim Brouzoulis, Department of Mechanics and Maritime Sciences

Master's Thesis 2021:44
Department of Mechanics and Maritime Sciences
Division of Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A topology optimisation simulation results of a cantilever beam with approximately 74000 elements.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

PyTopt
A Nonlinear Topology Optimisation Program in Python
DANIEL PETTERSSON
ERIK SÄTERSKOG
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Ever since the finite element method was introduced in the 1940's, the behaviour of structures could more easily be predicted. Combining this with an optimisation method made it possible to minimise a structure's weight while keeping the strength. Several topology optimisation programs have been created for this specific purpose since then. In this thesis, a topology optimisation program is implemented in Python with the method of moving asymptotes and the optimality criteria method as optimisation algorithms. By utilising previous works from universities, such as Lund University and the Technical University of Denmark, and combining the best parts a new topology optimisation program was created. The program has new functionalities not previously implemented in similar topology programs. I.e. functionalities such as the ability to utilise a variety of material models, being able to use an arbitrary design domain in 2D, body forces and asymmetrical yielding. This program also fits well in educational purposes due to its versatility. The results shows that optimality criteria method gives a better and faster result than the method of moving asymptotes. The results are also compared with commercial topology optimisation in ANSYS. The both programs gives indistinguishable results but ANSYS outperforms the thesis's program in regards to computational time.

Keywords: Topology, Optimisation, MMA, OC, FEM, Python, Body forces, Compliance, Density, SIMP

Acknowledgements

We would like to thank Knut Andreas Meyer for his support though-out the whole project. Without his inputs and assistance Pytopt would not be functioning in such general manner as it is.

Our friend Henrik has given us countless hours of friendly support and well-needed breaks during intensive hours of work.

We also thank Emma-Liisa Mikkola for her loving support during the whole master's thesis process.

We would also like to extend our gratefulness to Kårrestaurangen which has provided us with nutritious and tasteful food since the beginning of the project.

Daniel Pettersson and Erik Säterskog, Gothenburg, June, 2021

Contents

List of Figures

List of Tables

List of Symbols

α	Small strain in bilinear material model
$\boldsymbol{\epsilon}$	Strain
$\boldsymbol{\lambda}$	Lagrangian mulitplier in sensitivity analysis
$\boldsymbol{\sigma}$	Stress
$\boldsymbol{\varepsilon}_{\text{dev}}$	Deviatoric strain tensor
$\boldsymbol{f}^{\text{ext,C}}$	External force vector on the constrained nodes
$\boldsymbol{f}^{\text{ext,F}}$	External force vector on the free nodes
$\boldsymbol{f}^{\text{int,C}}$	Internal force vector on the constrained nodes
$\boldsymbol{f}^{\text{int,F}}$	Internal force vector on the free nodes
$\boldsymbol{f}^{\text{pre}}$	Prescribed force vector
δ_{ij}	Kronecker delta
η	Damping coefficient in optimality criteria
$\frac{\partial G_0}{\partial x_e}_{\text{mod}}$	Derivative of the objective function with filtering applied
γ_{kl}	Shear strains in bilinear material model
λ	Lagrangian multiplier in optimality criteria
$\boldsymbol{\Delta}\mathbf{h}_j$	Zero vector with a small value at location j
$\mathbf{a}$	Nodal displacement vector in FE solution
$\mathbf{B}$	Gradient of shape functions
$\mathbf{D}$	Tangent material stiffness matrix
$\mathbf{K_e}$	Element stiffness matrix
$\mathbf{K}$	Global stiffness matrix
$\mathbf{N}$	Shape functions
$\mathbf{R}$	Residual from nonlinear FE solution
$\mathbf{u}^{\text{C}}$	Constrained nodal displacement
$\mathbf{u}^{\text{F}}$	Free nodal displacement
∇	Gradient operator
Ω	Total area
I_{ijlk}	The fourth order identity tensor in Einstein notation.
$\tilde{G}_0$	Objective function with added lagrange multipliers
ε_{y}	Strain yield limit
ξ	Step length in optimality criteria
A_0	Maximum allowed area
B_k^η	Optimiality criteria collector variable
c	SIMP penalty factor coefficient
C_j	Constraint functions
E_{ijkl}	The fourth order stiffness tensor

G	Shear modulus
G_0	Objective function
H_f	Weight factor for filtering
K	Bulk modulus
p_i	Parameter i in optimisation problem
$r_{\min}$	Radius that specifies filter intensity
x_j	Density of element j
$\mathbf{b}$	Body force
$\mathbf{I}$	Second order identity tensor
e	Small hydrostatic strain in bilinear material model
k	Bilinear material model coefficient

1

Introduction

There is a lack of easily understood topology optimisation codes. The commercial programs are not open source, and the codes that are available to the public are often simplified, lacking important features or restricted to specific structures. To address these issues, a versatile topology optimisation program has been written in Python. This thesis will cover the development of this program.

1.1 Background

Designing load bearing components used to require vast knowledge of how forces interact with material. During the 1940's the finite element method was starting to be used [1] [5]. This made it significantly easier to design a structure that could withstand a force without collapsing, while keeping the mass of the structure low. Nowadays, a step further has been taken. By using optimisation algorithms together with the finite element method, an optimal structure can easily be designed with the help of topology optimisation programs.

One of the more known papers in the field is made by Bendsøe and Sigmund [6], the same authors also wrote a book [15], where elastic topology optimisation was thoroughly explained. However, when the deformations of a structure is sufficiently large, the assumption of linear elastic material behaviour is no longer valid. This creates the need of nonlinear material models. By introducing nonlinear materials in the optimisation a more realistic result is obtained when the deformations of the structure are large. Nonlinear topology optimisation has also been described in many papers, Jung and Chang [12] did this in their paper together with geometrical nonlinearity. In the article by Zhang and Paulino [18] topology optimisation with multiple nonlinear materials was covered. Nonlinear materials are vital in many applications. In a civil engineering approach, bridges and buildings are commonly built using concrete. This is a material with higher strength in compression than in tension [16].

Undiscovered fields within topology optimisation remain. Even though concrete is very common, topology optimisation of concrete structures are scarce. Most optimisation papers regarding concrete is about mixture and reinforcement optimisation [3, 11]. Furthermore, there is a lack of easily understood topology optimisation codes. Sigmund published a 99 lines long topology optimisation program in Matlab which is easy to understand [10] but it has limited capabilities. The user is restricted to a single geometry and material. These are things addressed in this thesis.

1.2 Purpose and Problem Description

The purpose of this Master's Thesis is to develop a versatile code capable of minimising compliance for an arbitrary 2D-finite element structure, using nonlinear material models and do this within a reasonable time. More specifically, the energy absorbed by the structure will be the objective function to be minimised. Boundary conditions, loads and the structure should be defined by the user. In addition, the user should be able to easily implement new material models, objective functions and optimisation routines. With the hope of a future application of the program in a course, the program should be easy to use and to understand.

To be able to say that this project fulfilled its purpose the following goals must be accomplished.

- Being able to run nonlinear material models, optimisation routines and objective functions created by the user for an arbitrary 2D structure.
- Being able to run topology optimisation analysis with nonlinear material with a resolution high enough to clearly draw conclusions from the results. This should be done in a reasonable amount of time; the computational time will depend on the material model. Because of this, it is estimated that optimising a model with 1 000 elements in one hour or less will be satisfactory for the material models in this thesis.
- Being able to run a model with linear material of 10 000 elements in one hour or less should be adequate due to the reduced computational cost using a linear model.
- A well written and clear code to such extent that it can be used in future optimisation courses at Chalmers University of Technology or at other technical universities.
- Being able to perform topology optimisation considering body forces.

1.3 Limitations and Simplifications

During this Master's thesis some limitations will be present. These limitations will affect the results.

- A total of 1600 hours of work will be spent during 20 weeks.
- Computational power is limited to the available computers. See A.2 for specifications.
- The optimisation will assume plane strain and small deformations.
- The optimisation result will be in 2D with a constant thickness.
- Material models that contain state or are time dependent will not be considered.

2

Structural Problem

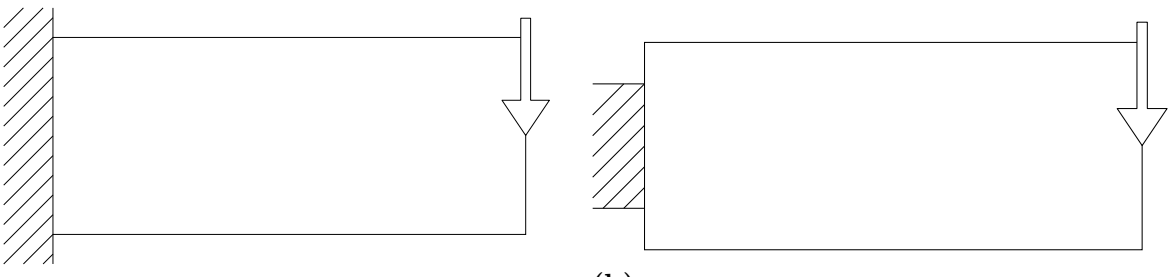
The structural problem, which is solved by a finite element analysis, is described in this chapter. The first step is to define design domains that will be used to evaluate the optimisation program. The Finite Element Method (FEM) is then explained for a general non-linear material model. Finally, the specific material models used are described.

2.1 Design domains

Design domains are the area accessible for the optimisation and are required in order to test the topology optimisation program. In this section will five different domains be presented, which several are inspired by examples in literature [15, 12].

2.1.1 Cantilever beam

The first example design domains are the cantilever beams in Figure 2.1. The cantilever beam is a classic engineering problem. Two versions of this problem often occurs in literature. In the first version, seen in Figure 2.1a, the design domain is restricted height wise to the same height as the fixed boundary [15]. The other version in Figure 2.1b allows the design domain to expand outside of the height of the fixed boundary.



(a) **(b)**

Figure 2.1: Two cantilever beam examples

A beam in compression could be considered as a third version of a cantilever beam. This version is very similar to the two versions above. The only difference is that the force is acting horizontally instead of vertically.

Figure 2.2: An engineering model of a beam fixed on the left boundary and a horizontal force acting on the right end.

2.1.2 L-Shape

An L-shape design domain is chosen as a fourth example. The L-shaped structure is fixed at the top and with a point-load applied at the rightmost upper corner, see Figure 2.3. The same design domain is used in [15] and is interesting due to the stress concentration at the re-entrant corner.

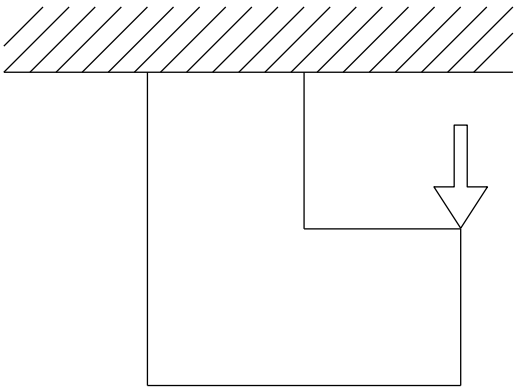


Figure 2.3: An engineering model of an L-shape fixed in one end.

2.1.3 Bridge

The last example is a 100 meter long bridge with two supports underneath, see Figure 2.4. This example is chosen to evaluate the program on a large-scale structures such as a bridge. The bridge is assumed to have cars on it and therefore a line load is applied on the upper edge of the structure. Heightwise, the structure is limited to 25 meters and is clamped on both sides.

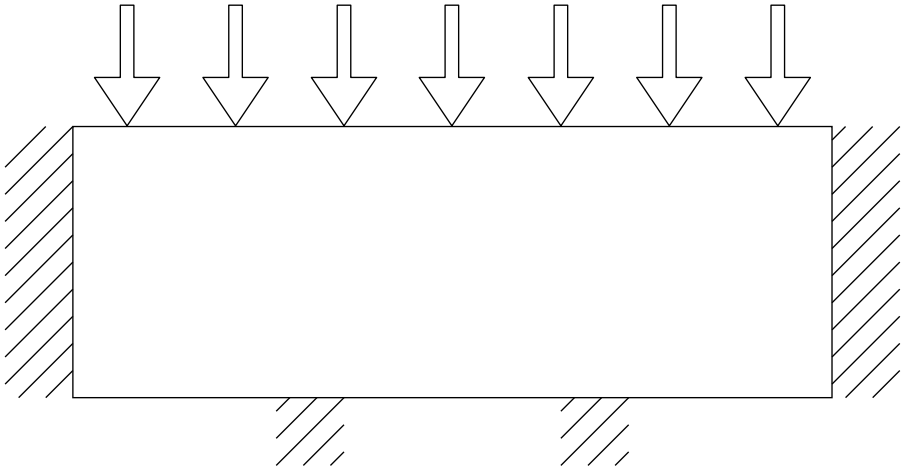


Figure 2.4: A hundred meters long bridge with two supports underneath and a line load on top.

2.2 FE analysis

The finite element method is an essential part in the optimisation process. By discretisation of the area into elements it is easy to apply the design variables to each element and to calculate the desired data, such as the displacements. In 2D, the most common element types are linear triangles and quadrilateral elements, both of which are implemented in the program. The strong form of the structural problem is expressed as

$$-\boldsymbol{\nabla}^{\mathrm{T}}\boldsymbol{\sigma} = \mathbf{b} \tag{2.1}$$

where $\boldsymbol{\sigma}$ is the stress matrix and $\mathbf{b}$ is the body load vector. The FE problem will be solved according to the methods described in *Introduction to the Finite Element Method* by Ottosen and Petersson [4]. The sought displacements, $\mathbf{u}$, is approximated with the shape-functions, $\mathbf{N}$, and the nodal displacements, $\mathbf{a}$ as

$$\mathbf{u} \approx \mathbf{N}\mathbf{a}. \tag{2.2}$$

To find the nonlinear solution of the displacements, iterations according to the Newton-Raphson iteration method are necessary. The residual, which is needed for the iteration method, is calculated as

$$\mathbf{R} = \boldsymbol{f}^{\mathrm{int}}(\mathbf{u}) - \boldsymbol{f}^{\mathrm{ext}}. \tag{2.3}$$

Where the force vectors are assembled as

$$\boldsymbol{f}^{\mathrm{int}} = \sum_{e=1}^{\mathrm{N}} x_e \boldsymbol{f}_e^{\mathrm{int}} = \sum_{e=1}^{\mathrm{N}} x_e \int_V \mathbf{B}_e^{\mathrm{T}} \boldsymbol{\sigma} \mathrm{d}V, \tag{2.4}$$

$$\boldsymbol{f}^{\mathrm{ext}} = \boldsymbol{f}^{\mathrm{pre}} + \sum_{e=1}^{\mathrm{N}} x_e \boldsymbol{f}_e^{\mathrm{ext}} = \boldsymbol{f}^{\mathrm{pre}} + \sum_{e=1}^{\mathrm{N}} x_e \int_V \mathbf{N}_e^{\mathrm{T}} \mathbf{b} \mathrm{d}V. \tag{2.5}$$

Where $\boldsymbol{f}^{\text{pre}}$ are the prescribed forces and the notations int and ext stands for internal and external respectively. $\mathbf{B}$ is the gradients of the shape-functions $\mathbf{N}$. x_e is the design variable for element e. The number of elements are denoted as N. The prescribed forces are applied node-wise and are not dependent on the design variable, also known as the element density. Body forces are scaled linearly with the density of the element. For example, gravity will affect elements with high density more than those with a low density. The stiffness matrix is defined as

$$\mathbf{K}^{\text{FF}} = \frac{\partial \mathbf{R}^{\text{F}}}{\partial \mathbf{u}^{\text{F}}} \tag{2.6}$$

and will for every Newton iteration solve for a new guess of the free displacements as

$$\mathbf{u}^{\text{F}}_{new} = \mathbf{u}^{\text{F}}_{old} - \mathbf{K}^{\text{FF}^{-1}}\mathbf{R}^{\text{F}}. \tag{2.7}$$

The iterations will continue until the residual is below a predefined tolerance. The notation F and C signifies the free respectively constrained degrees of freedom.

In the linear case, the internal forces are not needed to be calculated because the global stiffness can be assembled directly as

$$\mathbf{K} = \sum_{e=1}^{N} \mathbf{K}_e x_e. \tag{2.8}$$

The unknown displacements can instantly be determined by inverting the free part of the global stiffness matrix and multiplying with the global external force vector. The FE-equations can be written as

$$\mathbf{u}^{\text{F}} = \mathbf{K}^{\text{FF}^{-1}}\boldsymbol{f}^{\text{ext,F}} \tag{2.9}$$

$$\begin{bmatrix} \mathbf{K}^{\text{FF}} & \mathbf{K}^{\text{FC}} \\ \mathbf{K}^{\text{CF}} & \mathbf{K}^{\text{CC}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\text{F}} \\ \mathbf{u}^{\text{C}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^{\text{ext,F}} \\ \boldsymbol{f}^{\text{ext,C}} \end{bmatrix}. \tag{2.10}$$

2.3 Material models

A material's behaviour is predicted with the help of a material model. More specifically, given a strain ε, the material model gives the stress σ. The program will have three different material models that can be chosen by the user. The one most commonly used in literature and in commercial software is the linear elastic material model.

The linear elastic material model is trivial and follows Hooke's law,

$$\boldsymbol{\sigma} = 2G\boldsymbol{\varepsilon}_{dev} + K\mathbf{I}\text{tr}(\boldsymbol{\varepsilon}). \tag{2.11}$$

Here is $\text{tr}(\boldsymbol{\varepsilon})$ the trace of the strain tensor and equals three times the hydrostactic strain ε_h. $\boldsymbol{\varepsilon}_{dev}$ is the deviatoric part of the strain tensor. G and K are the shear modulus and bulk modulus respectively. In the process of material model development an experimental model was used, called Modified Hooke's law in this report. This remains in the program but will not

be investigated further in the report.

A new bilinear asymmetric material model is created in this thesis and is inspired by the behaviour of concrete [16]. Concrete is stronger in compression than in tension which is reflected in the material model. The material model utilises the hydrostatic strain to determine if the material is in compression or tension. When a limit for the hydrostatic strain ε_y has been reached, the material's Young's modulus is reduced. In Figure 2.5 is the hydrostatic yielding limit $\varepsilon_y = 0$. After this limit has been reached, the Young's modulus becomes 5% of its original value.

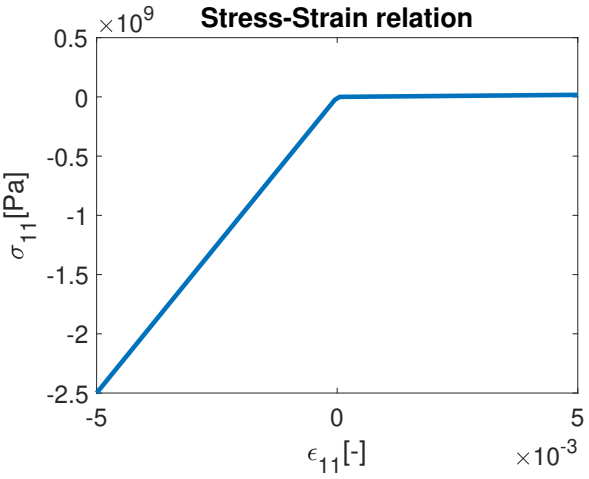


Figure 2.5: Stress-Strain response for the bilinear asymmetric material in axial compression/tension. The strain is controlled with ε_{22} =ε_{33} =0.

The material model is created by dividing the strain $\boldsymbol{\varepsilon}$ into two parts so that

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2 \tag{2.12}$$

$$\boldsymbol{\varepsilon}_1 = \boldsymbol{\varepsilon}(1-k) \tag{2.13}$$

$$\boldsymbol{\varepsilon}_2 = \boldsymbol{\varepsilon}k. \tag{2.14}$$

Where the factor k is determined as follows

$$k = \begin{cases} 1 - \frac{\boldsymbol{\varepsilon}_y}{\mathrm{tr}(\boldsymbol{\varepsilon})} & \text{if } \mathrm{tr}(\varepsilon) \geq \varepsilon_y \\ 0 & \text{if } \mathrm{tr}(\boldsymbol{\varepsilon}) < \varepsilon_y \end{cases}. \tag{2.15}$$

This means that $\boldsymbol{\varepsilon_1}$ would only represent the strain required to reach ε_y if $\mathrm{tr}(\boldsymbol{\varepsilon}) \geq \varepsilon_y$. All strain above the yielding limit will be included in $\boldsymbol{\varepsilon_2}$. Each part of $\boldsymbol{\varepsilon}$ will have a shear modulus and a bulk modulus such that

$$\boldsymbol{\sigma} = 2G_1\boldsymbol{\varepsilon}_{dev1} + K_1\mathbf{I}\ \mathrm{tr}(\boldsymbol{\varepsilon}_1) + 2G_2\boldsymbol{\varepsilon}_{dev2} + K_2\mathbf{I}\ \mathrm{tr}(\boldsymbol{\varepsilon}_2). \tag{2.16}$$

To be able to derive the tangent material stiffness matrix, $\mathbf{D}$, when $k > 0$ the stress must be considered in Einstein notation as

$$\sigma_{ij} = E_{ijkl}\varepsilon_{kl} = E_{ijkl_1}\varepsilon_{kl_1} + E_{ijkl_2}\varepsilon_{kl_2}. \tag{2.17}$$

Where E_{ijkl} is the fourth order material stiffness tensor which can be expressed as

$$E_{ijkl_1} = 2G_1(\mathrm{I}_{ijkl} - \frac{\delta_{ij}\delta_{kl}}{3}) + K_1\delta_{ij}\delta_{kl} \tag{2.18}$$

and

$$E_{ijkl_2} = 2G_2(\mathrm{I}_{ijkl} - \frac{\delta_{ij}\delta_{kl}}{3}) + K_2\delta_{ij}\delta_{kl}. \tag{2.19}$$

I_{ijkl} is the fourth order identity tensor and δ_{ij} is the second order identity tensor. With this relation the tangent material stiffness matrix can be derived as

$$\begin{aligned} D_{ijmn} &= \frac{\mathrm{d}\sigma_{ij}}{\mathrm{d}\varepsilon_{mn}} \\ &= \frac{\partial}{\partial\varepsilon_{mn}}\left(E_{ijkl_1}\varepsilon_{kl}(1-k)\right) + \frac{\partial}{\partial\varepsilon_{mn}}\left(E_{ijkl_2}\varepsilon_{kl}(k)\right) \end{aligned} \tag{2.20}$$

The first term in 2.20 can then be expanded as

$$\begin{aligned} \frac{\partial}{\partial\varepsilon_{mn}}\left(E_{ijkl_1}\varepsilon_{kl}(1-k)\right) &= E_{ijkl_1}\frac{\partial}{\partial\varepsilon_{mn}}(\varepsilon_{kl}(1-k)) \\ &= E_{ijkl_1}\left(\frac{\partial\varepsilon_{kl}}{\partial\varepsilon_{mn}}(1-k) + \varepsilon_{kl}\frac{\partial(1-k)}{\partial\varepsilon_{mn}}\right) \end{aligned} \tag{2.21}$$

The two terms inside the large parentheses in 2.21 can be expressed as

$$\frac{\partial\varepsilon_{kl}}{\partial\varepsilon_{mn}}(1-k) = \delta_{km}\delta_{ln}(1-k) \tag{2.22}$$

$$\varepsilon_{kl}\frac{\partial(1-k)}{\partial\varepsilon_{mn}} = -\varepsilon_{kl}\frac{\partial k}{\partial\varepsilon_{mn}} \tag{2.23}$$

$$\begin{aligned} \frac{\partial k}{\partial\varepsilon_{mn}} &= \frac{\partial(1-\frac{3\varepsilon_y}{\varepsilon_{oo}})}{\partial\varepsilon_{mn}} \\ &= 3\varepsilon_y\varepsilon_{oo}^{-2}\delta_{mn} \end{aligned} \tag{2.24}$$

Similar derivation will occur for the second term in 2.20. By inserting 2.21-2.24 in 2.20 the tangent material stiffness matrix is given as

$$D_{ijmn} = \begin{cases} E_{ijmn_1}(1-k) + E_{ijmn_2}k + (E_{ijkl_2} - E_{ijkl_1})(3\varepsilon_{kl}\varepsilon_y\varepsilon_{oo}^{-2}\delta_{mn}) & \text{if } \mathrm{tr}(\boldsymbol{\varepsilon}) \geq \varepsilon_y \\ E_{ijmn_1} & \text{if } \mathrm{tr}(\boldsymbol{\varepsilon}) < \varepsilon_y \end{cases}. \tag{2.25}$$

It can be noted that a negative stiffness can occur if the material is exposed to high shear strains compared to hydrostatic strains under the condition that $\mathrm{tr}(\boldsymbol{\varepsilon}) \geq \varepsilon_y$. To check this, let's consider the stress difference at a stiffness where $\mathrm{tr}(\boldsymbol{\varepsilon}) \geq \varepsilon_y$ is fulfilled, such that

$$\Delta\sigma_{ij} = D_{ijmn}[\varepsilon_{kl}]\Delta\varepsilon_{mn}. \tag{2.26}$$

Consider now ε_{kl} as a combination of a uniaxial strain tensor, $\varepsilon_{kl_{uni}}$, and a shear strain, γ_{kl}. The uniaxial strain, α, is in the 11-direction and is a small number. The shear strain is considered in the 12-direction. This means that

$$\varepsilon_{kl} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \gamma_{12} & 0 \\ \gamma_{12} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \varepsilon_{kl_{uni}} + \gamma_{kl}, \tag{2.27}$$

$$\varepsilon_{oo} = \varepsilon_{oo_{uni}} + \gamma_{oo} = \frac{\alpha}{3} > \varepsilon_y \tag{2.28}$$

and

$$D_{ijkl}[\varepsilon_{kl}] = E_{ijmn_1}(1-k) + E_{ijmn_2}k + (E_{ijkl_2} - E_{ijkl_1})\frac{(27(\varepsilon_{kl_{uni}} + \gamma_{kl})\varepsilon_y\delta_{mn})}{\alpha^2}. \tag{2.29}$$

From Equation 2.29 one can notice that it is only the last term that can inflict negative stiffness. It can also be observed that any shear strain multiplied with this term will be zero due to multiplication with δ_{mn}. Let $\Delta\varepsilon_{mn}$ in Equation 2.26 be a small strain, $e\delta_{mn}$ where $0 < e << 1$, such that

$$\Delta\varepsilon_{mn} = e\delta_{mn}. \tag{2.30}$$

Further, insert Equations 2.18 and 2.19 to the term of interest from Equation 2.29. Combining this with the Equations 2.26 and 2.30 gives

$$\begin{aligned} \Delta\sigma_{ij} = \Bigg(& E_{ijmn_1}(1-k) + E_{ijmn_2}k \\ & + (2(G_2 - G_1)(\mathrm{I}_{ijkl} - \frac{\delta_{ij}\delta_{kl}}{3}) + (K_2 - K_1)\delta_{ij}\delta_{kl})\frac{(27(\varepsilon_{kl_{uni}} + \gamma_{kl})\varepsilon_y\delta_{mn})}{\alpha^2} \Bigg) e\delta_{mn} \\ = & (E_{ijmn_1}(1-k) + E_{ijmn_2}k)e\delta_{mn} + \left(\frac{2(G_2 - G_1)(\varepsilon_{ij_{uni}} - \frac{\delta_{ij}\alpha}{3} + \gamma_{ij})}{\alpha} + (K_2 - K_1)\delta_{ij} \right) \frac{81\varepsilon_y e}{\alpha}. \end{aligned} \tag{2.31}$$

The last term in Equation 2.31 will be large as α goes towards zero. Especially will the part involving G_1 and G_2 be considerably large. Anisotropy could also be induced since a hydrostatic strain can influence shear stresses. If $G_1 > G_2$ this could also inflict a negative value of $\Delta\sigma_{ij}$ and that is not reasonable. This issue is addressed by choosing $G_1 = G_2$.

3
Topology Optimisation

The topology optimisation is the process of finding the optimal density distribution in a FE model. This chapter describes the topology optimisation process. The chapter begins with defining the objective function that should be minimised. After this, the derivative of this objective function, called sensitivity, will be derived. Finally, the expansion to include body forces and different objective functions will be covered.

3.1 Objective function

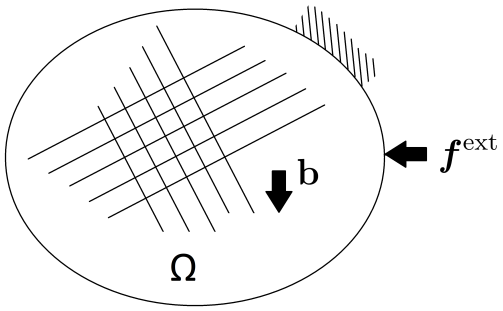


Figure 3.1: Arbitrary design domain with boundary condition and forces.

The mathematical expression of the optimisation problem is stated as

$$\begin{aligned} &\text{minimise} && G_0 = G_0(p_i(\mathbf{x}), \mathbf{x}) = \int_\Omega g_0(p_i(x), x)\mathrm{d}A \\ &\text{subject to} && C_j(p_i(\mathbf{x}), \mathbf{x}) \leq 0 \text{ with } j = 0, 1, ..., m \\ & && \mathbf{R} = \mathbf{0} \\ &\text{where} && p_i \text{ with } i = 0, 1, ..., n. \end{aligned} \tag{3.1}$$

$G_0(p_i(\mathbf{x}), \mathbf{x})$ is the *Objective Function* that will be minimised. The design variables $\mathbf{x}$ are in this case the element densities. Moreover, the parameters $p_i(\mathbf{x})$ are dependent on the design variables. For instance could such a parameter be the nodal displacements, $\mathbf{u}(\mathbf{x})$, of the finite elements. The total area available to the optimiser is Ω, see Figure 3.1. $\mathbf{R}$ is the residual from the FE analysis. The constraints are labelled as C_j where C_0 is a material constraint that specifies the amount of material that can be used. A material constraint is one of the most common constraints and can be written as

$$C_0(\mathbf{x}) = \int_\Omega x \mathrm{d}A - A_0 \leq 0. \tag{3.2}$$

Where A_0 is the maximum allowed area.

Minimisation of energy is widely used as objective function because of its simplicity [6, 15, 8]. Because of this reason the minimisation of energy will be the main objective function in this thesis. The energy is calculated by multiplying the external force vector with the displacement vector as

$$G_0 = \boldsymbol{f}^{\mathrm{ext}}(x_j)^{\mathrm{T}}\mathbf{u}(x_j). \tag{3.3}$$

In practice, this results in a minimisation of compliance. Therefore, this objective function will be called compliance in this thesis.
Two ways of implementing the objective function are described in [8]. One of the methods, minimisation of complementary work, involves load stepping and calculating the equilibrium at every load step. This method is computationally costly, but yields the best result [8].

The other method is the minimisation of end compliance. In this case a single load step is required and it is only when the total force has been applied that an equilibrium is calculated. The drawback of this method is that one can not be certain that the structure has not collapsed before the total load was applied [8]. This thesis will focus on this method due to the simplicity and speed of calculating a single load step.

3.2 Penalties and Filters

Solid Isotropic Material with Penalization (SIMP) is a method making it less favourable for elements to have a density between zero and one, since a material with density between zero and one is impossible to manufacture. As an example, when an element has a density of 0.5, it can not be said if the element should have material or not. This is visible as grey elements in simulation results. The method consist of raising the density to a power of c. This makes it inefficient for the optimiser to use material that is not zero or one. In Figure 3.2, is the SIMP method visualised for three different values of the SIMP penalty parameter c. SIMP is easily implemented into the FEM equations 2.4 and 2.8 as

$$\boldsymbol{f}^{\text{int}} = \sum_{e=1}^{N} x_e^c \int_V \mathbf{B}_e^{\text{T}} \boldsymbol{\sigma} \mathrm{d}V \tag{3.4}$$

and

$$\mathbf{K} = \sum_{e=1}^{N} \mathbf{K_e} x_e^c. \tag{3.5}$$

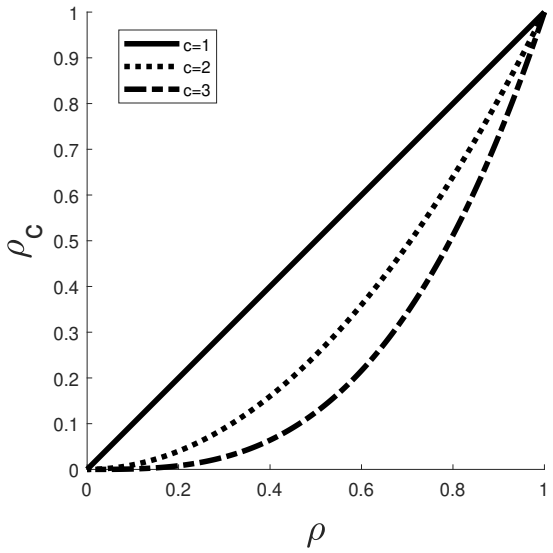
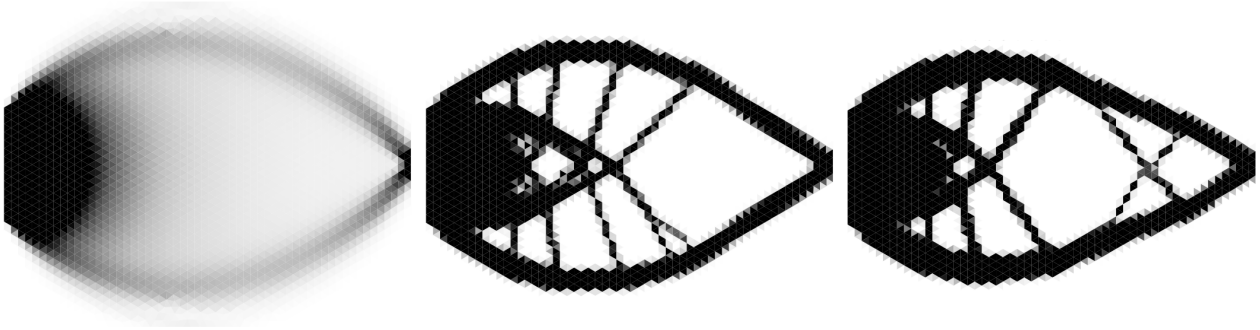


Figure 3.2: An illustration of how the SIMP parameter c affects the density in the SIMP method.

Simulations with the settings shown in Table 3.1 demonstrate the difference in results obtained for three different SIMP parameters in the SIMP method, see Figure 3.3. The first of the three parameters is $c = 1$ which means that SIMP is inactive. SIMP parameter equal to three is recommended by Bendsøs as the best value for c [6]. $c = 5$ is chosen to visualise the effects of a clearly higher c than recommended.

Table 3.1: Settings for simulations describing the influence of SIMP.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.02
Volume Fraction	0.3
rMin	0.014
SIMP	1/3/5
Linear	True
Optimiser	OC



(a) No SIMP method active.

(b) Intermediate value of SIMP parameter.

(c) High value of SIMP parameter.

Figure 3.3: Effect of different SIMP parameters.

When having a large SIMP parameter, the iterations necessary to find the optimal solution increases drastically and the optimisation is much more prone to have convergence issues. So even though the result is similar, the computational time makes a high SIMP parameter inefficient to use.

Checkerboard patterns have an artificially high stiffness compared to a uniformly distributed density [7]. This problem is also increased by the SIMP method. Instead of using elements between zero and one, the optimiser tends to create a checkerboard pattern, leaving every other element full and every other empty, which is also difficult to manufacture. To reduce this problem [10] chooses to implement a filter algorithm to create a mesh independent solution by modifying the sensitivity of the objective function. The sensitivities $\frac{\partial G_0}{\partial x_f}$, are a vital part of the optimisation routine as they describe the direction of the optimisation. The sensitivity analysis is described in Section 3.3 and the filter modifies the sensitivities as

$$\frac{\partial G_0}{\partial x_e}_{\text{mod}} = \frac{1}{x_e \sum_{f=1}^{N} H_f} \sum_{f=1}^{N} H_f x_f \frac{\partial G_0}{\partial x_f} \tag{3.6}$$

$$H_f = \max(r_{\min} - \text{dist}(e, f), 0). \tag{3.7}$$

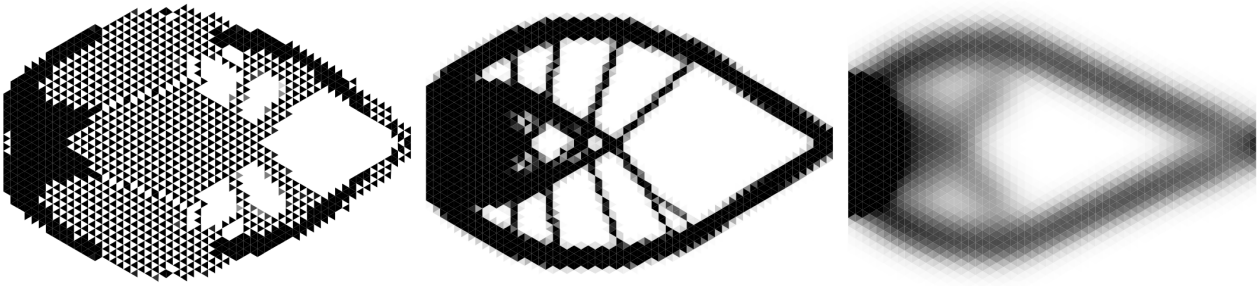
H_f is a weight factor that decreases linearly with the distance from the element's centre to the centres of nearby elements, up to the radius $r_{\min}$. The function 'dist' calculates the distance

from element e to element f.

Three simulations are done showing the effects of the filter in Figure 3.4. In Table 3.2 the settings for the simulation can be seen.

Table 3.2: Settings for simulations describing the influence of filters.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.02
Volume Fraction	0.3
rMin	0.002/0.014/0.08
SIMP	3
Linear	True
Optimiser	OC



(a) Low value for $r_{\min}$ **(b)** Intermediate value for $r_{\min}$ **(c)** High value for $r_{\min}$

Figure 3.4: Effect of different $r_{\min}$ constants

Checkerboarding occurs as predicted for a low value of $r_{\min}$. Although a too high $r_{\min}$ causes a blurry result and conclusions about the structure is hard to make.

3.3 Sensitivity Analysis

The sensitivity analysis is vital to find an optimal solution as the sensitivity analysis describes how the objective function depends on the design variables. The sensitivity analysis will be based on Tyler Bruns [9] approach in this thesis. Only a single element j will be looked at in this section. A general objective function is denoted as

$$G_0(\mathbf{u}^{\mathrm{C}}(x_j), \mathbf{u}^{\mathrm{F}}(x_j), \boldsymbol{f}^{\mathrm{ext,C}}(x_j), \boldsymbol{f}^{\mathrm{ext,F}}(x_j), \boldsymbol{f}^{\mathrm{int,C}}(x_j), \boldsymbol{f}^{\mathrm{int,F}}(x_j), x_j) \tag{3.8}$$

These quantities are defined in Section 2.2. The first step is to add Lagrangian multipliers, $\boldsymbol{\lambda}$. Adding the Lagrangian multipliers and the residual gives

$$\tilde{G}_0(x_j) = G_0 + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{R}, \tag{3.9}$$

with the condition that

$$\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{R} = 0. \tag{3.10}$$

Where $\mathbf{R}$ is the residual from the FE solution, see Equation 2.3. The residual can be divided into a free part and a constrained part. By expanding the constrained part, the objective function can be rewritten as

$$\begin{aligned}\tilde{G_0}(x_j) = G_0(\mathbf{u}^{\mathrm{C}}(x_j), \mathbf{u}^{\mathrm{F}}(x_j), \boldsymbol{f}^{\mathrm{ext,C}}(x_j), \boldsymbol{f}^{\mathrm{ext,F}}(x_j), \boldsymbol{f}^{\mathrm{int,C}}(x_j), \boldsymbol{f}^{\mathrm{int,F}}(x_j), x_j) \\ + \boldsymbol{\lambda}^{\mathrm{F}}\mathbf{R}^{\mathrm{F}}(\mathbf{u}^{\mathrm{C}}(x_j), \mathbf{u}^{\mathrm{F}}(x_j), x_j) + \boldsymbol{\lambda}^{\mathrm{C}}(\boldsymbol{f}^{\mathrm{int,C}}(\mathbf{u}^{\mathrm{C}}(x_j), \mathbf{u}^{\mathrm{F}}(x_j), x_j) - \boldsymbol{f}^{\mathrm{ext,C}}(x_j)).\end{aligned} \tag{3.11}$$

The equation is then differentiated such that

$$\begin{aligned}\frac{\mathrm{d}\tilde{G_0}}{\mathrm{d}x_j} = &\frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j}\left(\frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{C}}} + \boldsymbol{\lambda}^{\mathrm{F}}\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{C}}} + \boldsymbol{\lambda}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial \mathbf{u}^{\mathrm{C}}}\right) + \frac{\mathrm{d}\mathbf{u}^{\mathrm{F}}}{\mathrm{d}x_j}\left(\frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{F}}} + \boldsymbol{\lambda}^{\mathrm{F}}\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{F}}} + \boldsymbol{\lambda}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial \mathbf{u}^{\mathrm{F}}}\right) \\ &+ \frac{\mathrm{d}\boldsymbol{f}^{\mathrm{ext,C}}}{\mathrm{d}x_j}\left(\frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,C}}} - \boldsymbol{\lambda}^{\mathrm{C}}\right) + \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,F}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{ext,F}}}{\mathrm{d}x_j} + \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{int,C}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{int,C}}}{\mathrm{d}x_j} \\ &+ \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{int,F}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{int,F}}}{\mathrm{d}x_j} + \frac{\partial G_0}{\partial x_j} + \boldsymbol{\lambda}^{\mathrm{F}}\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j} + \boldsymbol{\lambda}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial x_j}.\end{aligned} \tag{3.12}$$

The Lagrangian multipliers can at this point be chosen arbitrarily. However, in the current expression, $\frac{d\mathbf{u}^{\mathrm{F}}}{dx_j}$ and $\frac{d\boldsymbol{f}^{\mathrm{ext,C}}}{dx_j}$ are particularly expensive to solve and would be preferably eliminated to increase efficiency [9]. These expressions do not have an analytical expression and requires an FE analysis in order to be solved. These terms can be eliminated by choosing the Lagrangian multiplier as

$$\boldsymbol{\lambda}^{\mathrm{C}} = \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,C}}} \tag{3.13}$$

$$\boldsymbol{\lambda}^{\mathrm{F}}\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{F}}} = -\left(\frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{F}}} + \boldsymbol{\lambda}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial \mathbf{u}^{\mathrm{F}}}\right). \tag{3.14}$$

The expression for the sensitivity will then be

$$\begin{aligned}\frac{\mathrm{d}\tilde{G_0}}{\mathrm{d}x_j} = &\frac{\partial G_0}{\partial x_j} + \frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{C}}}\frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j} + \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,F}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{ext,F}}}{\mathrm{d}x_j} + \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{int,C}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{int,C}}}{\mathrm{d}x_j} + \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{int,F}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{int,F}}}{\mathrm{d}x_j} \\ &+ \boldsymbol{\lambda}^{\mathrm{F}}\left(\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{C}}}\frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j} + \frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j}\right) + \boldsymbol{\lambda}^{\mathrm{C}}\left(\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial \mathbf{u}^{\mathrm{C}}}\frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j} + \frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial x_j}\right).\end{aligned} \tag{3.15}$$

The chosen objective function of compliance can now be inserted as

$$G_0(x_j) = \boldsymbol{f}^{\mathrm{ext}}(x_j)^{\mathrm{T}}\mathbf{u}(x_j) \tag{3.16}$$

and terms that are zero for this specific objective function are eliminated. I.e. $\frac{d\mathbf{u}^{\mathrm{C}}}{dx_j} = 0$, $\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial x_j} = 0$, $\frac{\partial G_o}{\partial \boldsymbol{f}^{\mathrm{int,C}}} = 0$, $\frac{\partial G_o}{\partial \boldsymbol{f}^{\mathrm{int,F}}} = 0$, and $\frac{\partial G_0}{\partial x_j} = 0$, resulting in

$$\frac{\mathrm{d}\tilde{G_0}}{\mathrm{d}x_j} = \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,F}}}\frac{\mathrm{d}\boldsymbol{f}^{\mathrm{ext,F}}}{\mathrm{d}x_j} + \boldsymbol{\lambda}^{\mathrm{F}}\left(\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j}\right) \tag{3.17}$$

3.3.1 Lagrangian multipliers

The terms in Equation 3.14 can with the objective function from Equation 3.16 be written as

$$\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{F}}} = \mathbf{K}^{\mathrm{FF}} \tag{3.18}$$

$$\frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{F}}} = \boldsymbol{f}^{\mathrm{ext,F}} \tag{3.19}$$

$$\boldsymbol{\lambda}^{\mathrm{C}} = \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,C}}} = \mathbf{u}^{\mathrm{C}} \tag{3.20}$$

$$\boldsymbol{\lambda}^{\boldsymbol{F}} = -\mathbf{K}^{FF^{-1}}(\boldsymbol{f}^{\mathrm{ext,F}} + \mathbf{u}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,F}}}{\partial \mathbf{u}^{\mathrm{F}}}) \tag{3.21}$$

In a linear case is $\boldsymbol{\lambda}^{\boldsymbol{F}} = -\mathbf{u}^{\mathrm{F}} - \mathbf{K}^{FF^{-1}}\mathbf{u}^{\mathrm{C}}\frac{\partial \boldsymbol{f}^{\mathrm{int,F}}}{\partial \mathbf{u}^{\mathrm{F}}}$. In both the linear and nonlinear cases $\mathbf{u}^{\mathrm{C}}$ is equal to zero since there are no prescribed displacements in the structure.

3.3.2 Differentiation of the residual

The differential of the residual with respect to the design variables will be derived as

$$\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\boldsymbol{f}^{\mathrm{int,F}} - \boldsymbol{f}^{\mathrm{ext,F}}\right), \tag{3.22}$$

and will with Equation 2.5 and 3.4 be

$$\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j} = \mathrm{c}x_j^{c-1}\int_V \mathbf{B}_j^{\mathrm{T,F}}\boldsymbol{\sigma}\mathrm{d}V - \int_V \mathbf{N}_j^{\mathrm{T,F}}\mathbf{b}\mathrm{d}V. \tag{3.23}$$

Note that the constant c is the SIMP coefficient.

3.3.3 Final expression

The first factor in Equation 3.17 is differentiated as

$$\frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,F}}} = \mathbf{u}^{\mathrm{F}}(x_j). \tag{3.24}$$

Finally the whole expression for the nonlinear sensitivity analysis can be expressed by using equations 3.21, 3.23 and 3.24. Which gives the expression

$$\frac{\mathrm{d}\tilde{G}_0}{\mathrm{d}x_j} = \mathbf{u}^{\mathrm{F}}(x_j)\int_V \mathbf{N}_j^{\mathrm{T,F}}\mathbf{b}\mathrm{d}V - \mathbf{K}^{FF^{-1}}\boldsymbol{f}^{\mathrm{ext,F}}\left(cx_j^{c-1}\int_V \mathbf{B}_j^{\mathrm{T,F}}\boldsymbol{\sigma}\mathrm{d}V - cx_j^{c-1}\int_V \mathbf{N}_j^{\mathrm{T,F}}\mathbf{b}\mathrm{d}V\right). \tag{3.25}$$

When body forces are not present, $\frac{\partial \boldsymbol{f}^{\mathrm{ext}}}{\partial x_j} = 0$, the final expression can be calculated as

$$\frac{\mathrm{d}\tilde{G}_0}{\mathrm{d}x_j} = -\mathbf{K}^{FF^{-1}}\boldsymbol{f}^{\mathrm{ext,F}}\left(cx_j^{c-1}\int_V \mathbf{B}_j^{\mathrm{T,F}}\boldsymbol{\sigma}\mathrm{d}V\right). \tag{3.26}$$

This final expression is the sensitivity needed for the optimisation.

3.4 Verification of sensitivity analysis using numerical differentiation

A numerical differentiation to the sensitivity analysis can be useful for verifying the analytical expressions. $\mathbf{\Delta h}_j$ is a zero vector with a small numerical perturbation at a single location j in the vector $\Delta \mathbf{h}_j$. A central difference scheme gives

$$\frac{\mathrm{d}G_0}{\mathrm{d}x_j}_{\text{ Num.}} = \frac{G_0(\mathbf{x} + \mathbf{\Delta h}_j) - G_0(\mathbf{x} - \mathbf{\Delta h}_j)}{2\mathbf{\Delta h}_j} \tag{3.27}$$

The error is calculated as

$$\text{Error} = \left(1 - \frac{\frac{\mathrm{d}G_0}{\mathrm{d}x_j \text{ Num.}}}{\frac{\mathrm{d}G_0}{\mathrm{d}x_j \text{ Anal.}}}\right) \tag{3.28}$$

3.5 Examples of other objective functions

The objective function could be chosen as many different functions depending on what type of optimisation is desired. Some objective functions could be:

- Minimisation of compliance
- Minimisation of displacements
- Minimisation of stress
- Minimisation of mass

Displacements could be used instead of the compliance as an objective function to determine an optimal structure. The objective function may, for example, be the square of the displacements. The objective function is then defined as

$$G_0(x_j) = ||\mathbf{u}||^2 \tag{3.29}$$

Recall Equation 3.15 from the sensitivity analysis. Every term where the displacement is not included will disappear and the only parts that remain are

$$\frac{\mathrm{d}\tilde{G}_0}{\mathrm{d}x_j} = \boldsymbol{\lambda}^{\mathrm{F}} \left(\frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{C}}} \frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j} + \frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j}\right) + \boldsymbol{\lambda}^{\mathrm{C}} \left(\frac{\partial \boldsymbol{f}^{\mathrm{int,C}}}{\partial \mathbf{u}^{\mathrm{C}}} \frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j}\right). \tag{3.30}$$

The Lagrangian's multipliers are chosen as

$$\boldsymbol{\lambda}^{\mathrm{F}} \frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial \mathbf{u}^{\mathrm{F}}} = -\left(\frac{\partial G_0}{\partial \mathbf{u}^{\mathrm{F}}} + \boldsymbol{\lambda}^{\mathrm{C}} \frac{\partial \boldsymbol{f}^{\mathrm{int,F}}}{\partial \mathbf{u}^{\mathrm{F}}}\right) \tag{3.31}$$

$$\boldsymbol{\lambda}^{\mathrm{C}} = \frac{\partial G_0}{\partial \boldsymbol{f}^{\mathrm{ext,C}}} = 0. \tag{3.32}$$

With Equation 3.18 this gives

$$\boldsymbol{\lambda}^{\mathrm{F}} = -2\mathbf{K}^{FF^{-1}}\mathbf{u}^{\mathrm{F}}. \tag{3.33}$$

The sensitivity will, due to the fact that $\frac{\mathrm{d}\mathbf{u}^{\mathrm{C}}}{\mathrm{d}x_j} = 0$, be

$$\frac{\mathrm{d}\tilde{G}_0}{\mathrm{d}x_j} = -2\mathbf{K}^{FF^{-1}}\mathbf{u}^{\mathrm{F}} \frac{\partial \mathbf{R}^{\mathrm{F}}}{\partial x_j}. \tag{3.34}$$

Where the last factor will take the same expression as in Equation 3.23. This objective function is implemented within the program. However, this is an initial implementation and due to inconclusive results are the results left out of the thesis.

3.6 Optimisation Algorithms

To find the optimal solution an optimisation algorithm is used. Optimisation functions use the sensitivities of the objective function to iterate towards a minimum. Since the optimisation involves constraints, not all optimisation methods are suitable. The most popular methods in topology optimisation are *Optimality Criteria* (OC) and *Method of Moving Asymptotes* (MMA)[14, 17]. Both of these algorithms will be explored.

3.6.1 Optimality Criteria

Optimality Criteria (OC) was described in 1995 by Martin Bendsøe and Ole Sigmund [6]. However, OC is inefficient if multiple constraints are active and is usually only used for single constraint optimisation [14]. OC was also used in "A 99 lines of topology optimization code written in Matlab" published by DTU [10] which is a great source of information on how to apply the optimisation method. The OC can be summarised as:

$$x_{k+1} = \begin{cases} \max((1-\zeta)x_k, 0) & \text{if } x_k B_k^\eta \leq \max((1-\zeta)x_k, 0) \\ x_k B_k^\eta & \text{if } \max((1-\zeta)x_k, 0) \leq B_k^\eta \leq \min((1+\zeta)x_k, 1) \\ \min((1+\zeta)x_k, 1) & \text{if } \min((1+\zeta)x_k, 1) \leq x_k B_k^\eta \end{cases} \tag{3.35}$$

Where ζ is a step length, η is a damping coefficient and

$$B_k^\eta = \left(\frac{-\frac{\partial G_0}{\partial x_k}}{\lambda} \right)^\eta . \tag{3.36}$$

λ is here the Lagrangian multiplier that makes sure that the volume constraint is not broken. However, Equation 3.36 implies that $\frac{\partial G_0}{\partial x_k}<0$ for all k. When body forces or varying loads are implemented, $\frac{\partial G_0}{\partial x_k}<0$ can not be assured for all k. Since the damping coefficient is less than one this would cause imaginary numbers [14]. To account for non-constant loads or body forces B_k^η will be altered as

$$B_k^\eta = -\text{sign}\left(\frac{\partial G_0}{\partial x_k} \right) \left(\frac{|\frac{\partial G_0}{\partial x_k}|}{\lambda} \right)^\eta . \tag{3.37}$$

This is done using a bi-sectioning algorithm in [10] as

$$\lambda = \frac{\lambda_1 + \lambda_2}{2} \tag{3.38}$$

$$\text{if } \int_\Omega x > \int_\Omega A_0 \text{ then } \lambda_1 = \lambda \text{ else } \lambda_2 = \lambda \tag{3.39}$$

An optimal solution is found by repeating the process above until the difference between the both limits are below a predefined threshold.

3.6.2 Methods of Moving Asymptotes

The method of moving asymptotes (MMA) by Krister Svanberg was first introduced 1987 [2] and is still a well accepted optimisation method for structural designs [10]. MMA utilises asymptotes to keep the optimisation within the constraints. In contrast to OC, MMA can directly handle multiple global constraints but with the cost of being more complex to implement [15]. In addition, it is also shown that OC gives a better result considering final compliance than MMA and that the two methods can converge to different optima if the SIMP parameter is larger than one [14]. However, this is under the condition that only one constraint is active. This is a very short and simplified description of MMA. For a more in depth explanation how the MMA works, the reader is recommended to read [2] by Krister Svanberg. The MMA iteration process is according to [2] as follows

1. Choose a starting point $\mathbf{x}$ and let iteration variable $k = 0$.
2. Calculate $G_0(\mathbf{x}^k)$ and $\nabla G_0(\mathbf{x}^k)$
3. Create a subproblem P^k with approximations of G_0 from previous step.
4. Solve the subproblem P^k and let the optimal solution be next starting point $\mathbf{x}^{(k+1)}$
5. $k = k + 1$ and go to step 2.

3.7 Topology Optimisation in ANSYS

Several commercial topology optimisation software exist [19][22][20]. In this thesis, ANSYS [21] will be used to verify the result from our topology optimisation code written in Python.

In ANSYS, the available optimisation algorithms are Optimality Criteria and Sequential Convex Programming. Sequential convex programming is an extension of MMA [21]. The built in objective functions in ANSYS are mass, volume and compliance. The program also has the possibility to implement a custom objective function. Constraints that can be used are local and global Von Mises stress, mass, volume, displacement and reaction force. A custom criterion can be implemented here as well. There are also ways to alter the penalty factor, the maximum number of iterations and a minimum element density. The penalty factor is the same as the Solid Isotropic Material with Penalization (SIMP) constant [21]. ANSYS is using a filter algorithm for the optimisation but the user can not change the settings for the filer more than if the filter will be linear or nonlinear.

4
Program Structure

This chapter will go though the application of the earlier explained theory in a topology optimisation program. An example will be demonstrated and lastly each module has its own section where it is explained in more detail.
The program, is from now on called PyTOpt, is divided into several modules. To download PyTOpt go to `https://github.com/ErikSaterskog/PyTOpt`. Mainly three Python libraries were used in creating this program. All three are listed below.

- Numpy - Basic Library for maths in Python.
- Scipy - Library with linear algebra functionality
- Calfem - FEM Library

Both *Numpy* and *Scipy* are frequently used libraries which contributes with algebraic and matrix functionality. *Scipy*'s sparse-matrices made it possible to use denser meshes for the problems studied. CALFEM is created at Lund University and adds the possibility to create meshes and geometries with the help of GMSH [24].

4.1 A Short Topology Optimisation Example

An example will be explained briefly in this section. The example will contain all the necessities needed from a potential end user. For a more in-depth explanation please view the Appendix A.1. The example problem can be read in Listing 4.1 and consists of these six main parts:

1. Importing Modules.
    - Import all the important modules for creating a geometry and calling the optimisation.
2. Creating geometry.
    - Set up the domain that will be optimised with the help of CALFEM. The example demonstrates the design domain for a cantilever beam.
3. Forces and boundary conditions.
    - Mark fixed boundaries and apply loads on the geometry.
4. Material parameters
    - Determine the material model for the optimisation and its required parameters. Material models available are:
        - Elastic
        - Bilinear
5. Settings
    - Select an objective Function for the optimisation. The two available objective functions are:
        - Compliance

  - Displacement
- Choose an Optimisation algorithm for the topology optimisation. PyTOpt have these two to choose from:
  - OC
  - MMA
- Denote additional optimisations settings. These are for instance the SIMP parameter and mesh size.

6. Calling PyTOpt
   - The optimisation program PyTOpt is called with the parameters and settings assigned in 1-5 as input.

Listing 4.1: Example of a program written for topology optimisation with PyTOpt.

```
# Importing Modules
import calfem.geometry as cfg
import Pytopt.PyTOpt as PyTOpt
from Pytopt import Material_Routine_Selection as mrs
from Pytopt import Object_Func_Selection as ofs
from Pytopt import Optimisation as opt

# Creating geometry
g = cfg.Geometry()

g.point([0,0])
g.point([1,0])
g.point([1,0.4],marker=9)
g.point([1,0.8])
g.point([0,0.8])
g.point([0,0.5])
g.point([0,0.3])


g.line([0, 1],marker=0)
g.line([1, 2],marker=1)
g.line([2, 3],marker=2)
g.line([3, 4],marker=3)
g.line([4, 5],marker=4)
g.line([5, 6],marker=5)
g.line([6, 0],marker=6)

g.surface([0, 1, 2, 3, 4,5,6])

# Forces and boundary conditions
force = [-1e6,9,2]
bmarker = 5
eq=[0,0]

# Material parameters
mp = {'E':210e9,'nu':0.3,'eps_y':0 }
materialFun = mrs.Bilinear

# Settings, Objective function and Optimisation routine
ep=[1,True,2]
settings = {'volFrac':0.3,'meshSize':0.03,'rmin':0.03*0.7,'changeLimit': 0.0,'SIMP_const':3}
ObjectFun = ofs.Displacement
OptFun = opt.OC

# Calling the optimisation
PyTOpt.Main(g, force, bmarker, mp, ep, materialFun, ObjectFun, OptFun,settings,eq,maxiter=150)
```

4.2 PyTOpt

The main module *PyTOpt.py* connects all the other modules. The module requires inputs such as geometry, material model and optimisation settings. If settings are missing, PyTOpt will start by auto-filling the missing settings and tries to do an optimisation based on a mixture of user defined settings and default settings. Both the definition of *PyTOpt.Main* and the default settings can be seen in Listing 4.2.

Listing 4.2: The beginning of the PyTOpt module showing the inputs and default settings.

```
_defaultSettings = {'volFrac':0.5,'meshSize':0.1,'rmin':0.1*0.7,'changeLimit': 0.01,'SIMP_const':3,'Debug':False}

def Main(g, force, bmarker, mp, ep, materialFun, ObjectFun, OptFun, settingsdict={}, eq=None, maxiter=30):
```

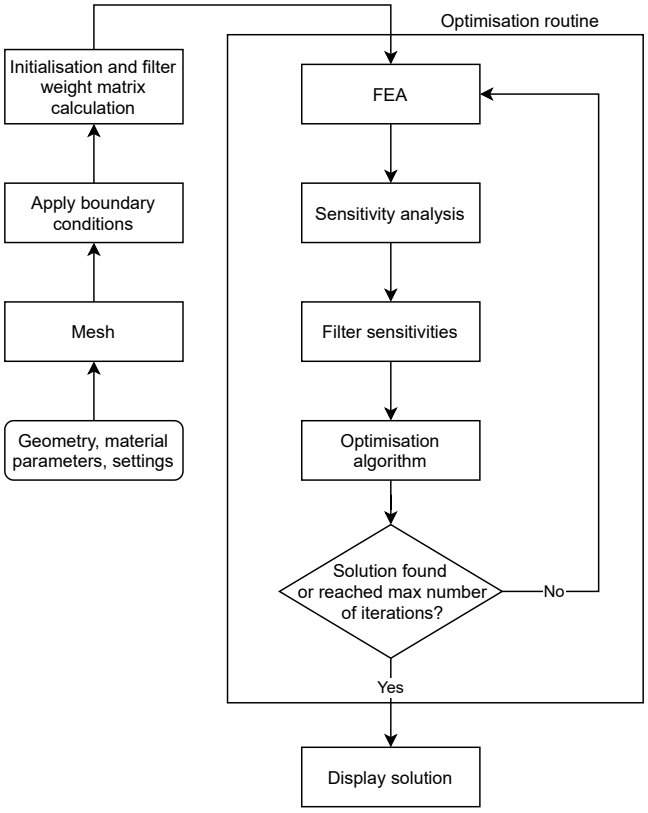


Figure 4.1: Flowchart of the programflow of *PyTOpt.py*.

A simplified flowchart of the module can be seen in Figure 4.1. The first step is to mesh the geometry with help of the module *mesh.py*. In the next step, the main module defines the degrees of freedom of the problem and interpret the boundary conditions set by the user. An uniform starting guess for the density distribution is assumed based on the allowed volume fraction. Initialisation of the FEM-problem is done next, as well as calculating the weight matrix used in the filter module. The last part of the module is the optimisation loop. The loop contains the finite element analysis, sensitivity analysis and the filtering step. A plot of the current optimised structure will be shown every fifth optimisation iteration. The optimisation will run until it satisfies the conditions set by the user; either the optimisation tolerance or a limit on the maximum number of iterations. When finished, the result will be shown in a plot

over the final optimised structure. In addition, a plot over the hydrostatic strain distribution will be presented to easier interpret the results from the optimisation, especially when the bilinear asymmetric material model is used.

4.3 Mesh

The module *Mesh.py* meshes the geometry. The module receives the geometry, approximate element size and the type of element as input. With help from the CALFEM library the geometry is constructed as a 2D mesh with two translational degrees of freedom for each node. GMSH is used as a mesh generator [24] and is called within CALFEM routines. The mesh can handle two different types of elements, linear triangles and linear quadrilaterals.

4.4 Finite Element Analysis

The Finite Element Analysis module *FE.py* solves the finite element problem. It requires the mesh, the boundary conditions, the prescribed forces and the design parameters. Depending on the settings, *FE.py* either uses a linear solver or a nonlinear solver. The linear solver constructs the global stiffness matrix and solves the displacements according to equation 2.9. The nonlinear solver utilises a nonlinear material model and uses Newton-Raphsons iteration method to find the displacements.

4.5 Filter

The module *Filter.py* utilises a low pass filter to smooth the distribution of objective function sensitivities. The inputs are the filter weight matrix calculated in *PyTOpt.py*, the sensitivities and the design variables. *Filter.py* assigns the weighted average of nearby elements to the each element. The weight is stored in the weight matrix and is declining linearly with the distance from the element to a maximum radius of $r_{\min}$. Output from the module is the updated sensitivities.

4.6 Optimise

The optimisation module *Optimisation.py* will contain the optimisation algorithms OC and MMA. Both the optimisation algorithms are callable separately with design variables, sensitivities, element areas, volume fraction, optimisation iteration number and present number of the objective function. With this input, the optimisation returns updated design variables. The Method of Moving Asymptotes algorithm is based on the functions written by Arjen Deetman [23]. Deetman's code is a translation of Krister Svanbergs MMA-GCMMA code written in MATLAB [13]. The Optimality Criteria optimisation method follows the routine described in the theory chapter. It is a modified version of the code written in the 99 lines topology optimisation code [10].

5
Simulations and Results

This chapter presents the results provided by simulations with PyTOpt. Before running the simulations, an investigation was performed to find the parameter values in the optimisation functions which gives the lowest compliance. Results from this investigation is first presented. The chapter then continues with presenting simulation results from using the two material models. Body forces and the dependency of the initial density distribution, $\boldsymbol{x}_0$, were also examined in this chapter.

5.1 Optimisation Parameters and Convergence

The optimisation depends largely on the settings used. Bendsøe and Sigmund recommends a step length of 0.2 [6] which represents the maximum change of the density for a element for one optimisation iteration. In this section a brief investigation into the optimisation algorithm's parameters will be done. All simulations in this section will be done on the cantilever beam seen in Figure 5.1.

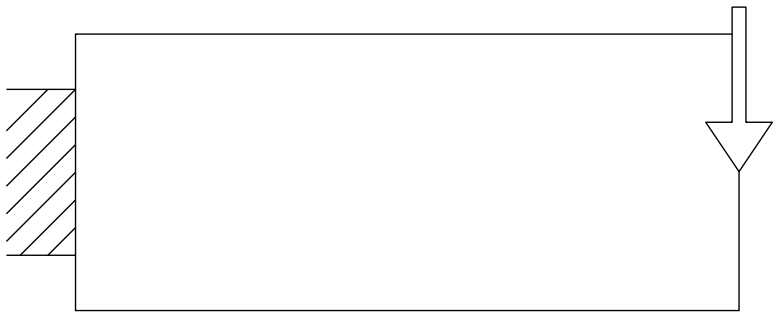
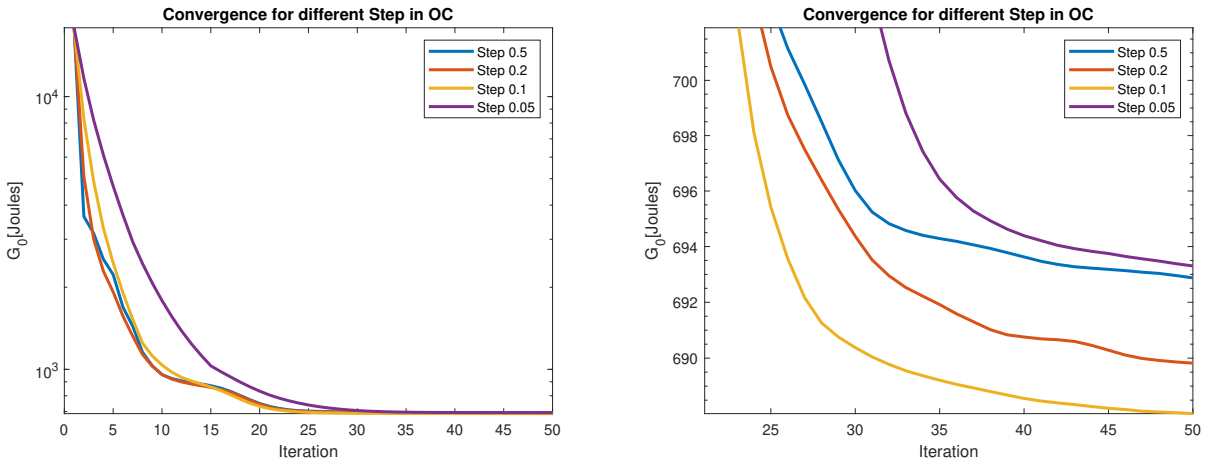


Figure 5.1: The second cantilever beam example from Chapter 2.

5.1.1 Optimality Criteria

The parameter *Step* is changed between four different cases and the convergence behaviour is observed in Figures 5.2a and 5.2b. The parameter *step* describes how large step the OC optimisation can take in a direction. Settings for the simulations can be found in Table B.1.



(a) Convergence of compliance with different values of *step*.

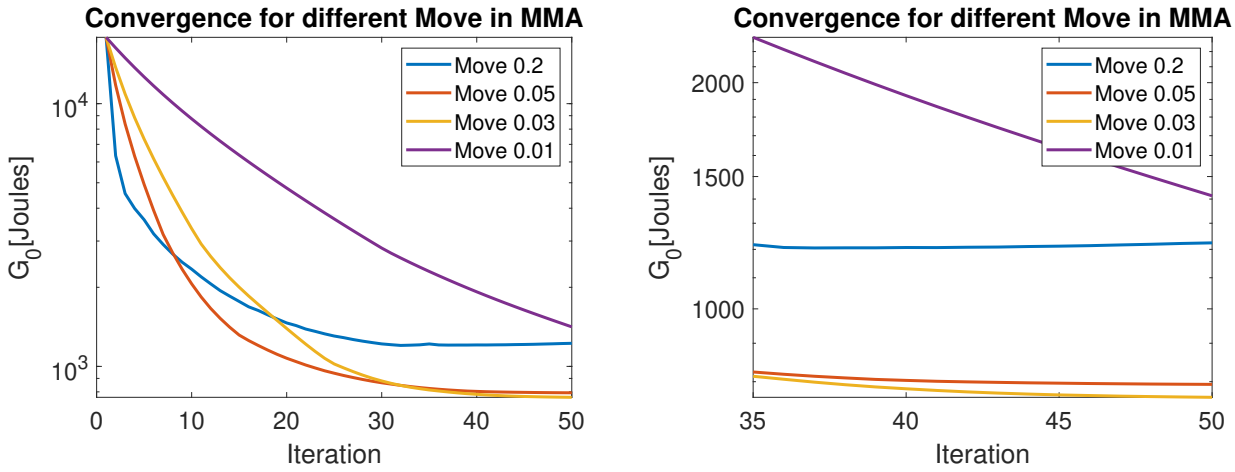
(b) A close-up on the convergence curves.

Figure 5.2: Impact of the OC parameter *step*.

It can be seen that a sufficiently good setting is *step*=0.1 as this solution finds the lowest minimum and the convergence rate is on par with the ones with a higher *step*-value.

5.1.2 Method of Moving Asymptotes

Arjen Deetmans sets the *move* parameter in MMA to 0.2 in his topology optimisation code [23]. The parameter *move* has roughly the same function as *step* has in OC. However, 0.2 is not the optimum value according to our observations. Simulations were run with different values of the parameter *move* and the convergence in the different cases can be seen in Figure 5.3. The settings for the simulations are in Table B.2. It can be observed in Figure 5.3a that a higher *move* parameter gives a quicker convergence. By studying Figure 5.3b one can notice that the *move* parameter suggested by Deetmans gives the second highest end compliance. Furthermore, a lower *move* parameter gives a lower end compliance when an ample amount of iterations are performed. A low *move* parameter on the other hand converges slower. It can be seen that the simulation with a *move* parameter equal to 0.01 has not yet converged after 50 iterations.

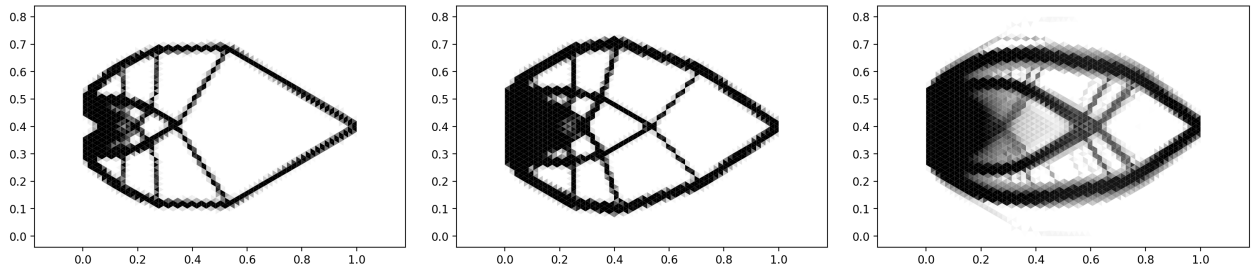


(a) Convergence of Compliance for 4 different values of the parameter *move*.

(b) A close-up on the convergence curves.

Figure 5.3: Impact of MMA parameter *move* on the optimisation.

Additionally the optimised structures can be seen in Figure 5.4. Note that the optimised structure, when having $move = 0.01$, is not fully converged as there are a lot of grey elements left. This confirms the conclusions drawn from the convergence plot. Further it can be seen that the simulation result with $move = 0.2$ utilises less material than the optimised structure with $move = 0.05$. More material usually means a stiffer structure, which is confirmed by the convergence plot.



(a) Simulation with the parameter $move = 0.2$.

(b) Simulation with the parameter $move = 0.05$.

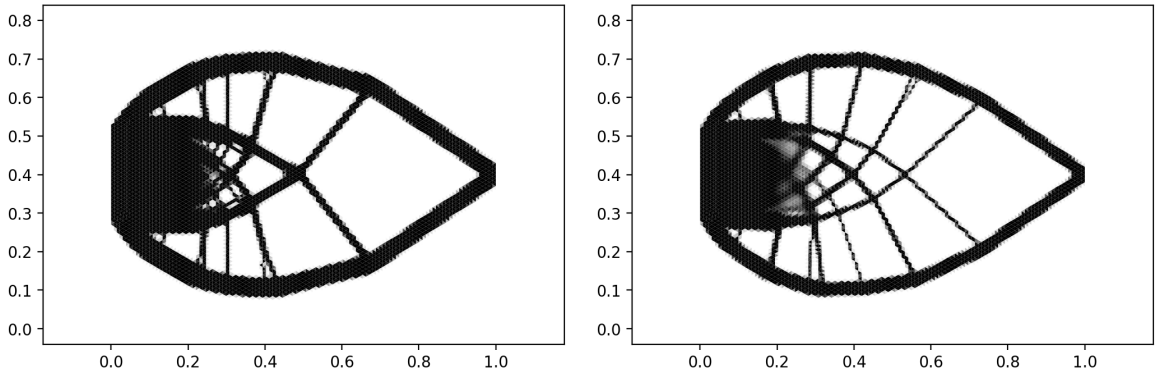
(c) Simulation with the parameter $move = 0.01$.

Figure 5.4: Results from three simulations with different values for the parameter *move*.

5.2 Linear Topology Optimisation

In this section an optimisation of a cantilever beam with the elastic material model will be executed. A total of 18 618 elements were used in the optimisation for both the optimisation algorithms, Method of Moving Asymptotes (MMA) and Optimality Criteria (OC). The settings used can be seen in Table B.3. The same example is presented in *Topology Optimization: Theory, Methods, and Applications*, [15].

Results from the optimisation can be seen in Figure 5.5. Neither of the optimised structures are completely symmetrical and which can depend on mesh irregularities. Moreover, there are differences between the final structures for MMA and OC. MMA has a thinner outer frame and more bars going out from the centre. OC has a thicker outer frame with fewer but thicker bars going out from the centre. The results in *Topology Optimization: Theory, Methods, and Applications* are also fairly similar to the result presented here. The results also confirms Groenwolds and Etmans observation that MMA and OC can find different optimums if the SIMP-constant is larger than one [14]. Compliance for the optimised structures are approximately 697.67 J for OC and 810.32 J for MMA.

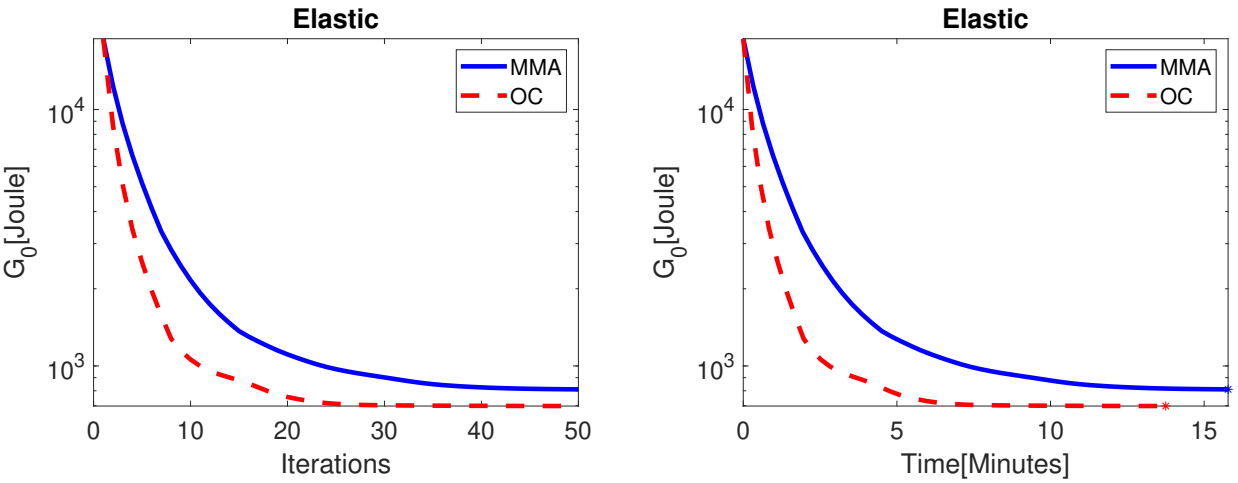


(a) Final compliance for the OC optimisation is 697.67 J and it took 13 minutes and 43 seconds to complete.

(b) Final compliance for the MMA optimisation is 810.32 J and it took 15 minutes and 49 seconds to complete.

Figure 5.5: OC and MMA density distribution comparison for a Cantilever beam.

It can be seen in Figure 5.6a and 5.6b that OC is converging faster than MMA. It converges faster both in matter of iterations and time. This is anticipated when only one constraint is active [14], which is the case here.



(a) The compliance convergence over iterations.

(b) The compliance convergence over time.

Figure 5.6: Convergence for OC and MMA regarding time and iterations.

5.3 Bilinear Topology Optimisation

The bilinear topology optimisation uses a material that is weaker in tension as described in section 2.3. The optimised structure can be seen in Figure 5.7. The structure consists of 19 000 elements and has a total computational time of roughly 8 hours and 15 minutes and a final compliance of 1084.38 J. The optimised structure has potentially a honeycomb pattern in the grey area, see Figure 5.7. The honeycomb pattern is much more prone to appear for a lower value of the filter constant r_{min}. The settings for the simulation can be seen in Table B.4.

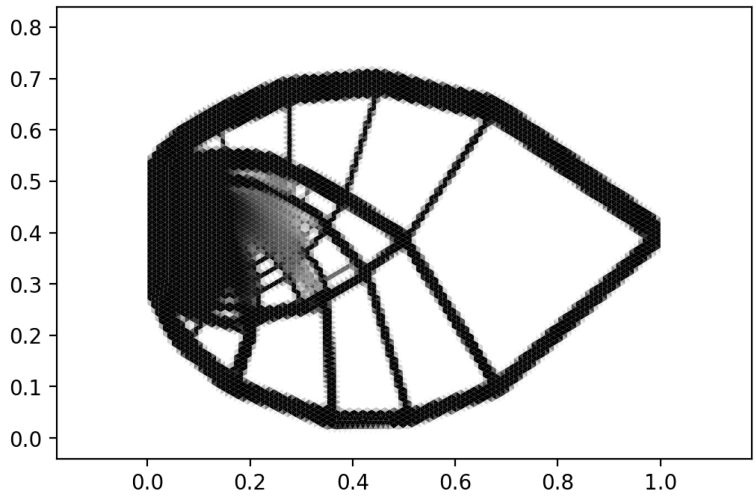


Figure 5.7: Result from the optimisation with bilinear material model with Optimality Criteria as optimisation algorithm.

The hydrostatic strain over the optimised cantilever beam, see Figure 5.8, is indicating that the beam is more in tension than compression. As a result of the weakness in tension the optimiser compensates with more material.

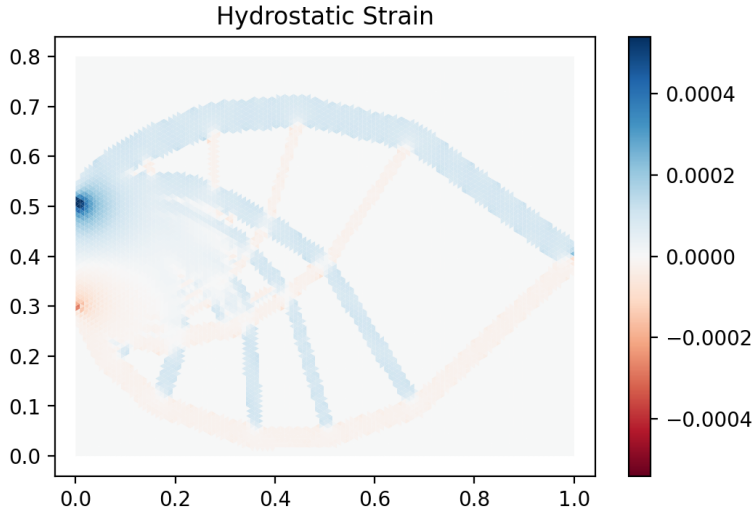


Figure 5.8: The hydrostatic strain over the optimised structure. Note that the colourbar is capped at half the absolute maximum hydrostatic strain.

5.4 Optimisation with line load and multiple boundaries

The program can also handle line load and multiple defined boundaries. In the following example an optimisation is presented on the bridge design domain, earlier described in section 2.1.3. The line load symbolises cars on a carriageway. The settings for the simulation can be found in Table B.5. Furthermore, in this simulation quadratic elements were used to show the utility for these. The optimised bridge can be seen in Figure 5.9.

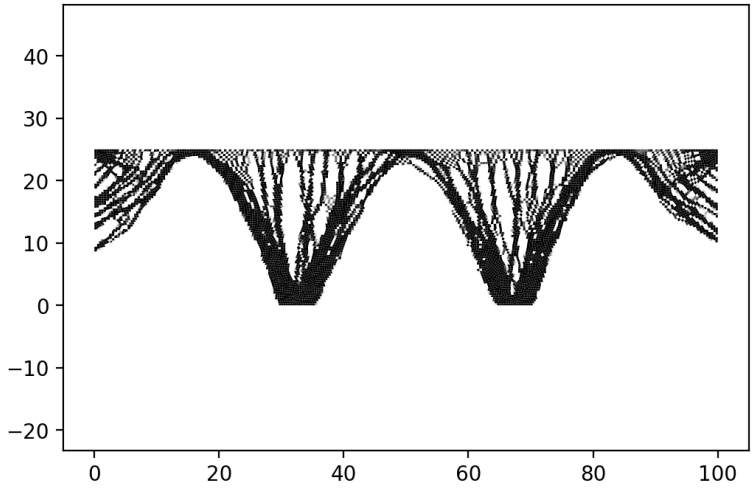
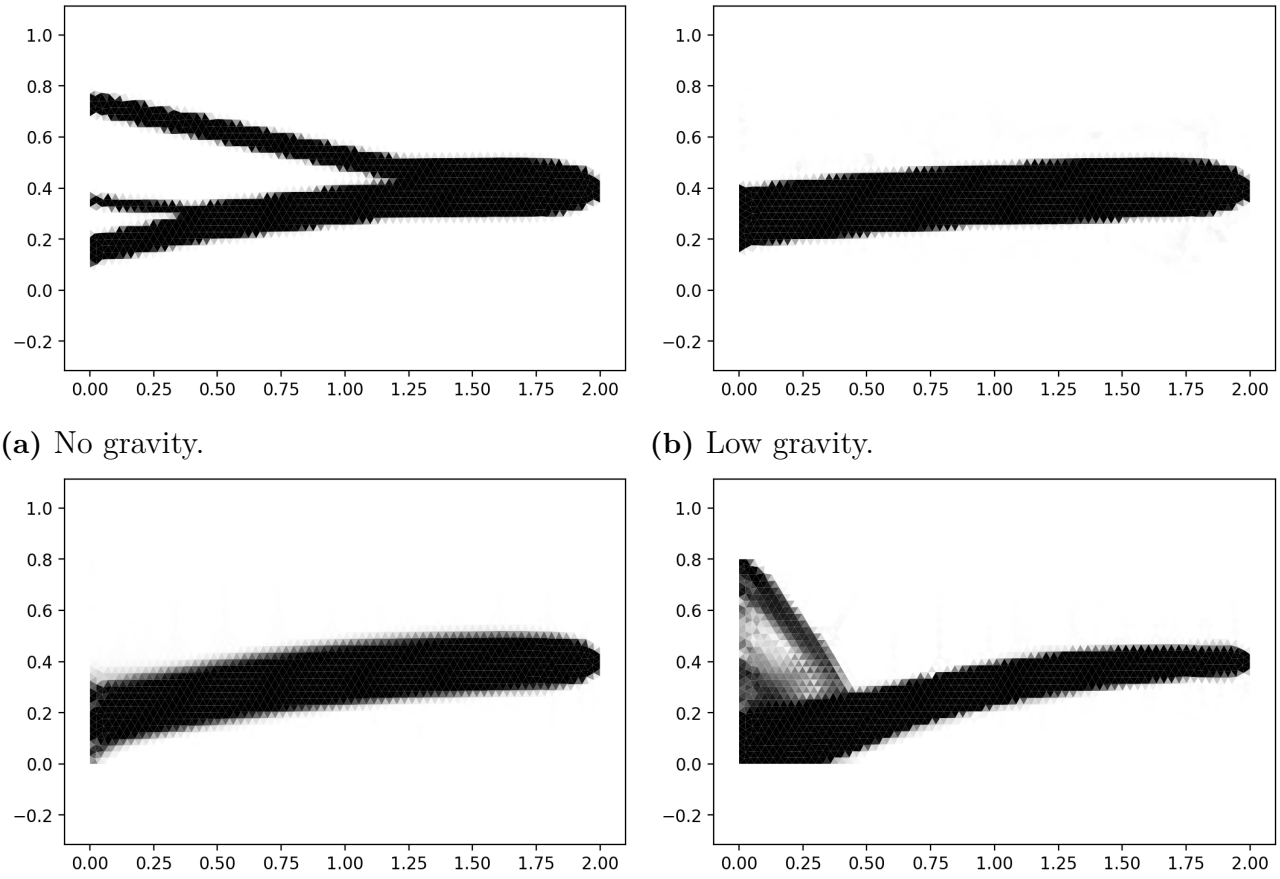


Figure 5.9: An optimised structure of a bridge with line load with Optimality Criteria as optimisation algorithm.

5.5 Body Forces

Applying the body forces causes some interesting changes to the optimised topologies. By applying vertical gravity to the beam in compression the following topologies were obtained.



(a) No gravity.

(b) Low gravity.

(c) Medium gravity.

(d) High gravity.

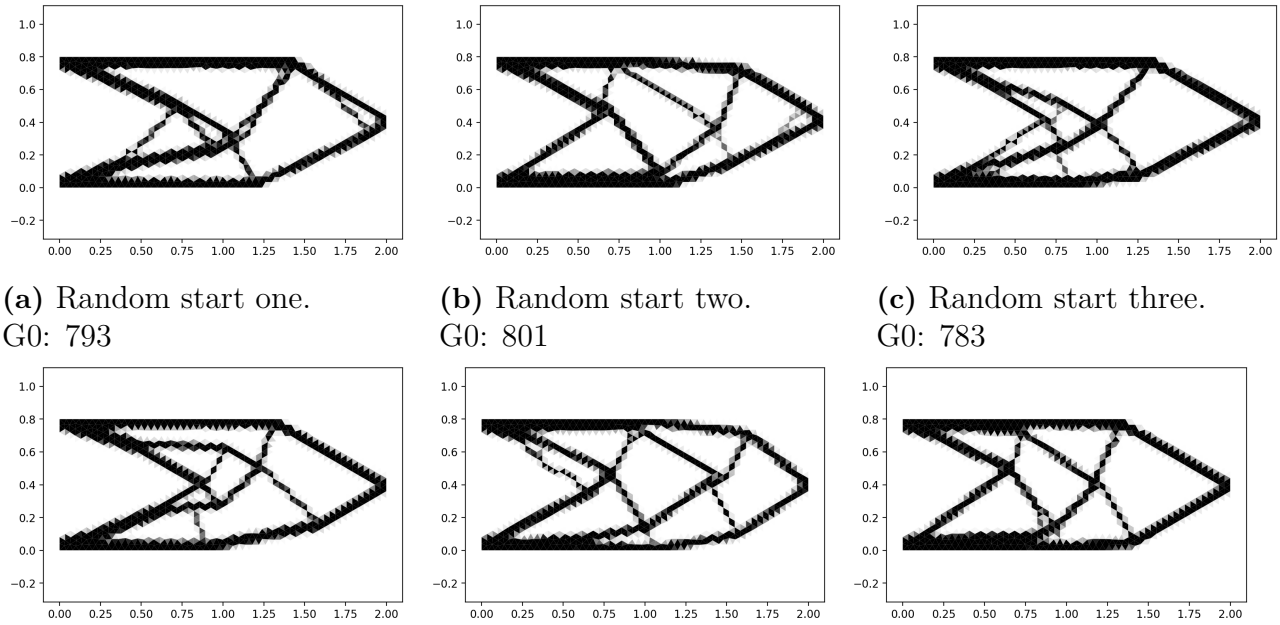
Figure 5.10: Evolution of gravity induced topology changes. Low gravity implies a structure with low material density.

It can be seen from Figure 5.10 that the structure starts to curve from the bottom left towards the applied force as the gravity increases. The gravity inflicts a moment at the bottom left corner. This moment is counteracted by the force, thanks to the curvature of the beam. The settings for these simulations can be found in Table B.6.

5.6 Dependency of Initial Density Distribution

The optimisation starts with an uniform distribution of density in the beginning of the optimisation. However, when using a randomised start, different results can be obtained. This dependency of the initial density distribution on the cantilever beam is more thoroughly investigated in this section. Different outcomes can be seen in Figure 5.11 when random starts were assigned. Running the same example but starting with an uniform distribution gave the result

shown in Figure 5.11f. The settings for this simulation can be seen in Table B.7. The simulation was done in 200 iterations, which is well enough to find a converged solution according to Figure 5.12.



(a) Random start one. G0: 793

(b) Random start two. G0: 801

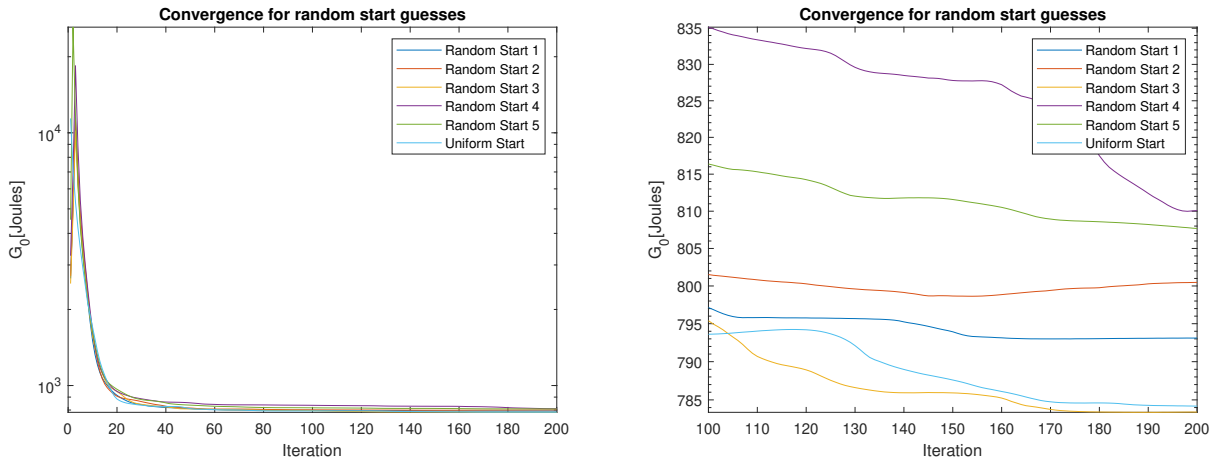
(c) Random start three. G0: 783

(d) Random start four. G0: 810

(e) Random start five. G0: 808

(f) Uniform start. G0: 784

Figure 5.11: Results from five random starts and an uniform start.



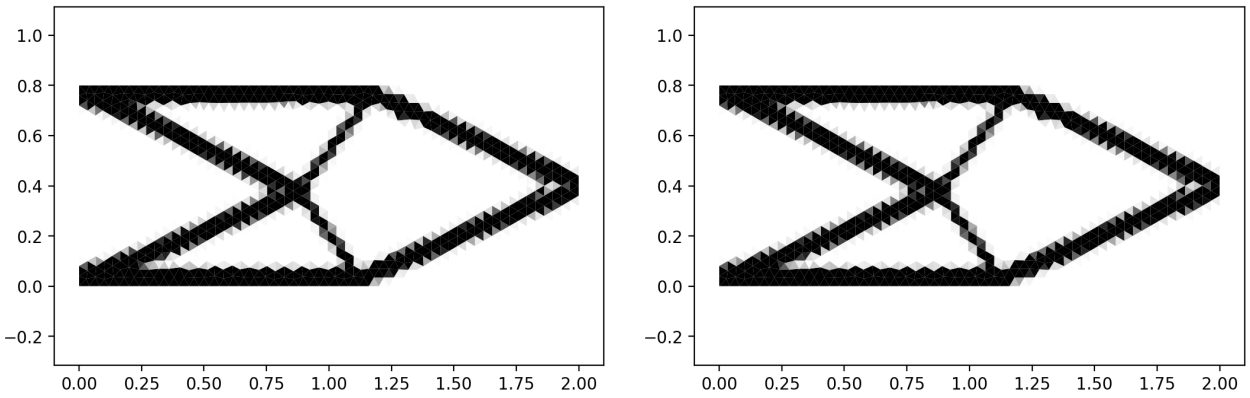
(a) Convergence curves for the random and uniform starts.

(b) Close-up on the convergence curves.

Figure 5.12: The development of compliance for the random and uniform start.

Repeating a simulation with a uniform starting guess will result in the same optimised structure

every time. However, it is certain that this is not the optimal solution since one of the randomised starts outperformed the one with uniform distribution. A varying $r_{\min}$ can be used in an attempt to reduce the dependency on the initial density distribution. By starting with a high $r_{\min}$ value and making it decrease towards the standard setting throughout the optimisation a result independent from starting position can be obtained as seen in Figure 5.13.



(a) Result from random start with end compliance of 776.78 J.

(b) Result from random start with end compliance of 776.78 J.

Figure 5.13: Results with adapting r_{Min} and random starts.

These topologies started with a random start but converged into the same solution, see Figure 5.14. Observe that this method outperformed the uniform start, it is still not certain that a global minimum has been found.

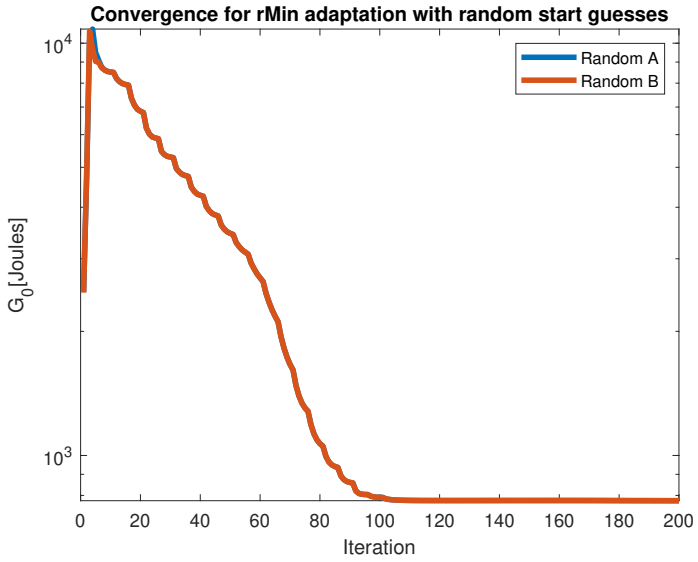
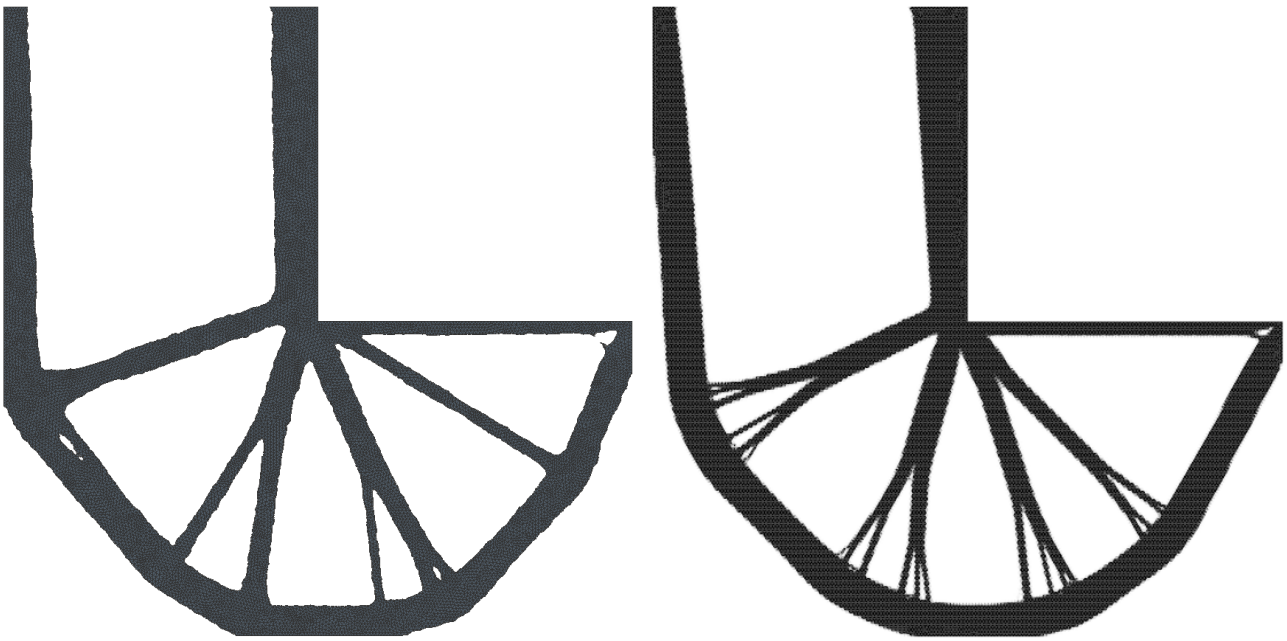


Figure 5.14: Convergence for two random start guesses with an adapting filter.

5.7 Ansys Comparisons

In this section the L-Beam example will be compared between PyTOpt and ANSYS. Figure 5.15a shows the result from the ANSYS optimisation. The ANSYS optimisation was done by constructing the design space in CATIA V5. After inserting the correct supports and forces could the optimisation be done with ANSYS. The settings from the simulation on the L-beam design domain in PyTOpt can found in Table B.8. There are only minor differences between the results. The small differences between the two solutions can depend on the filter's impact. In PyTOpt the bars creates delta-like formations at the bottom which is not occurring in the ANSYS result. The ANSYS optimisation takes around 7 minutes to complete and the PyTOpt optimisation takes 66 minutes to complete, more than 9 times longer computational time.



(a) Results from ANSYS **(b)** Results from PyTOpt

Figure 5.15: A topology optimisation comparison between ANSYS and PyTOpt.

There is not a setting for adjusting the filter in ANSYS so it can not be assured that the same filter setting was used in the two optimisations [21]. Another factor leads to differences in the results are the meshes. It is not possible to make sure that both of the programs uses the same mesh without extensive further development of PyTOpt.

5.8 Numerical Sensitivity Analysis

A sensitivity comparison is done on the cantilever beam to confirm the nonlinear sensitivity analysis. The Cantilever Beam example is used with the settings shown in Table B.9. The error depending on the perturbation in the numerical approach can be seen in Figure 5.16 and is calculated according to Equation 3.28. The standard deviation can be seen in the error bars and the mean value is seen as the line graph. At high perturbation values the error is increasing linearly. At low values numerical instability occurs causing errors in the mean as well as in the spread of the values.

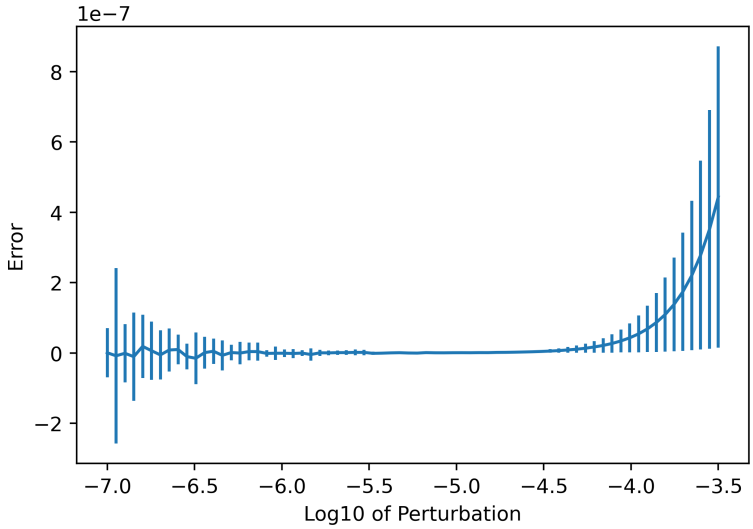


Figure 5.16: The error between a numerical and an analytical nonlinear sensitivity analysis.

6
Discussion and Conclusion

This section will discuss the results found in the simulation chapter. We will then go through the goals to see if they have been fulfilled or not.

6.1 Program

The final version of the program is easy to use. The end-user only needs to specify a geometry and where the forces and boundary conditions should be applied. The rest could easily be chosen from the already existing settings, material models and optimisation algorithms.

Another benefit of the program is its versatility. One of the advantages is the possibility to input almost any kind of geometry. While this makes the program run slower than if only a rectangular geometry is possible, it adds functionality which is important. The possibility of a future user to implement their own functions was prioritised when developing the program. Because of this, it is simple to use another material function, objective function, element routine or optimisation routine than what is included in the original program. One of the disadvantages with this program is that it is slower than other software available. The ANSYS optimisation with the same amount of elements took about 10 % of the time when using a linear material model.

It would be interesting to continue the program with the addition of another problem, such as heat exchange instead of a structural FE-problem. It should be fairly straightforward to implement with the current interface for the objective function. Displacement is implemented into the program as an additional objective function. However, the suitability for this objective function is limited as the results are difficult to draw any conclusions from. Another feature that could be implemented is the ability to add areas in the geometry where full material is required as in [15] and [6].

The program's results are intended as an inspiration for further development rather than an exact solution to a loading case. Even though the result may not be easily manufactured, seeing how an optimal structure would look like might help in the development of a design possible to manufacture. Since there often occurs grey elements in the solution some critical thinking is necessary to draw conclusions from the result.

The program is stable and written well enough so that it may be used in a future course in topology optimisation. If not, it served as an excellent opportunity for us, the developers, to learn more about the topic and further extend our knowledge within FEA and programming in Python.

6.2 Settings

There are many settings that the user can adjust until a desired solution is found. Although all of these settings are not needed for a simple optimisation. In agreement with the literature found, a SIMP parameter of three seems to be giving the best results [6]. A higher SIMP parameter gives a very similar solution however the computation time is increased drastically. A lower SIMP parameter creates more elements with values between one and zero. There is a middle-ground where the solutions are not too slow and gives good results. An observation found that when the SIMP is turned off, checkerboarding is greatly reduced. This is interesting as SIMP is not the main reason checkerboarding occurs.

The main purpose of the filter is to reduce the mesh dependency of the solution. By smoothing out the solution, checkerboarding is removed which can have been created by the SIMP. However, in this thesis the filter is only used to remove checkerboarding. Since we are at an experimental stage, and we do not plan to manufacture any of the results, we want to keep all the fine intricate structures that emerge from the optimisation when using a fine mesh. Because of this, the filter's radius is usually set to $Meshsize \cdot 0.7$ in the optimisations presented in this thesis. The mesh size is a factor of this value since more elements will fit inside the filter's radius with a smaller mesh size. When using this value checkerboarding is avoided and not too many of the small intricate structures are removed. A higher radius will smooth things out more than necessary. When using a bilinear material model this value for $r_{\min}$ gives a honeycomb pattern instead of checkerboarding. So when blinear material is used $r_{\min}$ is chosen as $Meshsize \cdot 0.9$. The result may instead be a bit blurry but with a fair amount of iterations this is not a problem.

What value that is an acceptable $r_{\min}$ is highly dependent on the situation that one performs the optimisation in. With an additive manufacturing approach, a lower $r_{\min}$ might be suitable since more complicated geometries can be constructed. When such methods are not available a higher $r_{\min}$ can be used to create a geometry that is more easily constructed.

The best settings for an optimisation greatly depends on what situation the result is going to be used. Because of this, it is important to choose the settings best suited for a specific task. However, the SIMP parameter should be set to three independent of situation according to our findings.

6.3 Meshing

The meshing is done with the help of GMSH and CALFEM. One of the two main problems encountered when meshing is the sub-optimal solution of having elements of different sizes and shapes. By having a structured mesh, where every element has the same shape and size, the

computational time could be reduced by a lot. With a nonlinear material model however, each element must be evaluated separately even if the shape is the same. Because of this the computational time will not be reduced under those circumstances and the benefits of a structured mesh is decreased. Another problem with an unstructured mesh is meshing irregularities that affect the solution. Solutions may not be symmetrical even though the problem's geometry is.

6.4 Optimiser

MMA is often used in literature [13] and as main optimisation algorithm in software as ANSYS [21]. This is contradictory to our and Groenwolds findings which show that OC is better in both speed and in finding the optimal solution [14]. The fact that MMA is so widely used could be explained by the reasons that MMA is more versatile since it simplifies incorporation of multiple constraints.

6.5 Material Model

The material model used is very important during the optimisation. A linear elastic material model is by far the fastest since the newton iteration procedure can be dismissed. The Newton iterations are the main reason for the increased time needed for optimisation with a nonlinear material model. The bilinear material model converges rather fast, only 4 to 5 Newton iterations are needed for each optimisation iteration.

It is difficult to draw conclusions from the bilinear material model since it is unclear whether the optimiser tends to add more material in tension, or if it tries to avoid tension altogether. In the result we can notice that the bars are thicker in tension and have a higher absolute strain compared to the areas in compression. The example used may not be able to remove the tension and therefore makes the tension bars thicker.

It would be interesting to implement additional material models to observe what differences these would have on the end result. One example on such model could be a rubber material. Further could also time and state-dependent materials be of interest to implement, such as hardening in steels.

6.6 Goals and Future Work

We defined a set of goals in the introduction. These goals included being able to run the optimisation with a nonlinear material model in a reasonable time. The code produced should also be well documented and easily understood. Finally we had the goal to perform topology optimisation considering body forces.

We managed to fulfil all of the goals. The code can run an optimisation for a linear elastic material model with about 10 000 elements in a matter of minutes. The nonlinear optimisation is a lot slower but the optimisations can still run with 1 000 elements in a matter of minutes as well.

Although, this is done with the material models created within this thesis. The program does not ensure that material models created by an end user to run within minutes for 1 000 elements.

Body forces are implemented and working. However, compliance may not be a suitable objective function when body forces are applied. Compliance wants to minimise the displacements simultaneously as minimising the effect of the body forces. This makes the optimisation jump back and fourth between solutions. In future work, an objective function on displacement, see section 3.5, could be more suitable. Furthermore, implementing an optimisation that accounts for structural dynamic behaviour could also be a possibility for future work.

Bibliography

[1] Alexander Hrennikoff. "Solution of problems of elasticity by the framework method". In: *J. appl. Mech.* (1941).

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

[3] James SM Shilstone et al. "Concrete mixture optimization". In: *Concrete International* 12.6 (1990), pp. 33–39.

[4] Niels Saabye Ottosen and Hans Petersson. *Introduction to the Finite Element Method*. English. Prentice-Hall, 1992. ISBN: 0-13-473877-2.

[5] Richard Courant et al. "Variational methods for the solution of problems of equilibrium and vibrations". In: *Lecture notes in pure and applied mathematics* (1994), pp. 1–1.

[6] Martin P Bendsøe and Ole Sigmund. *Optimization of structural topology, shape, and material*. Vol. 414. Springer, 1995.

[7] Ole Sigmund and Joakim Petersson. "Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima". In: *Structural optimization* 16.1 (1998), pp. 68–75.

[8] Thomas Buhl, Claus BW Pedersen, and Ole Sigmund. "Stiffness design of geometrically nonlinear structures using topology optimization". In: *Structural and Multidisciplinary Optimization* 19.2 (2000), pp. 93–104.

[9] Tyler E. Bruns and Daniel A. Tortorelli. "Topology optimization of non-linear elastic structures and compliant mechanisms". In: *Computer Methods in Applied Mechanics and Engineering* 190.26 (2001), pp. 3443–3459. ISSN: 0045-7825. DOI: https://doi.org/10.1016/S0045-7825(00)00278-4. URL: https://www.sciencedirect.com/science/article/pii/S0045782500002784.

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

[11] Matěj Lepš and Michal Šejnoha. "New approach to optimization of reinforced concrete beams". In: *Computers & structures* 81.18-19 (2003), pp. 1957–1966.

[12] Daeyoon Jung and Hae Chang Gea. "Topology optimization of nonlinear structures". In: *Finite Elements in Analysis and Design* 40.11 (2004), pp. 1417–1427.

[13] Krister Svanberg. "MMA and GCMMA-two methods for nonlinear optimization". In: *vol* 1 (2007), pp. 1–15.

[14] Albert A. Groenwold and L. F. P. Etman. "On the equivalence of optimality criterion and sequential approximate optimization methods in the classical topology layout problem". In: *International Journal for Numerical Methods in Engineering* 73.3 (2008), pp. 297–316. DOI: https://doi.org/10.1002/nme.2071. eprint: https://onlinelibrary.wiley.

com/doi/pdf/10.1002/nme.2071. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2071.

[15] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

[16] Tehmina Ayub, Sadaqat Ullah Khan, and Fareed Ahmed Memon. "Mechanical characteristics of hardened concrete with different mineral admixtures: a review". In: *The Scientific World Journal* 2014 (2014).

[17] Lei Li and Kapil Khandelwal. "Two-point gradient-based MMA (TGMMA) algorithm for topology optimization". In: *Computers and Structures* 131 (2014), pp. 34–45. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2013.10.010. URL: https://www.sciencedirect.com/science/article/pii/S0045794913002812.

[18] Xiaojia Shelly Zhang, Glaucio H Paulino, and Adeildo S Ramos. "Multi-material topology optimization with multiple volume constraints: a general approach applied to ground structures with material nonlinearity". In: *Structural and Multidisciplinary Optimization* 57.1 (2018), pp. 161–182.

[19] *Altair OptiStruct™ Optimization-enabled Structural Analysis*. 2021. URL: https://www.cati.com/design-analysis/altair-simulation/hyperworks/altair-optistruct/ (visited on 05/03/2021).

[20] *Ansys Mechanical Topology Optimization (Self-paced Learning Available)*. 2021. URL: https://www.ansys.com/training-center/course-catalog/structures/ansys-mechanical-topology-optimization (visited on 05/03/2021).

[21] *Structural Optimization analysis*. 2021. URL: https://ansyshelp.ansys.com/account/secured?returnurl=/Views/Secured/corp/v211/en/wb_sim/ds_topology_optimization.html (visited on 05/06/2021).

[22] *TOSCA STRUCTURE OPTIMIZE WITH ABAQUS, ANSYS, OR MSC NASTRAN*. 2021. URL: https://www.3ds.com/products-services/simulia/products/tosca/structure/ (visited on 05/03/2021).

[23] Arjen Deetman. */arjendeetman/GCMMA-MMA-Python*. URL: https://github.com/arjendeetman/GCMMA-MMA-Python (visited on 04/09/2021).

[24] Christophe Geuzaine and Jean-François Remacle. *A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. URL: http://gmsh.info/ (visited on 03/12/2021).

A

A.1 A Detailed Topology Optimisation Example

A short example of how the program can be used is shown in Listing A.1 and will be thoroughly explained in the following section.

Listing A.1: Example program

```
# Importing Modules
import calfem.geometry as cfg
import Pytopt.PyTOpt as PyTOpt
from Pytopt import Material_Routine_Selection as mrs
from Pytopt import Object_Func_Selection as ofs
from Pytopt import Optimisation as opt

# Creating geometry
g = cfg.Geometry()

g.point([0,0])
g.point([1,0])
g.point([1,0.4],marker=9)
g.point([1,0.8])
g.point([0,0.8])
g.point([0,0.5])
g.point([0,0.3])


g.line([0, 1],marker=0)
g.line([1, 2],marker=1)
g.line([2, 3],marker=2)
g.line([3, 4],marker=3)
g.line([4, 5],marker=4)
g.line([5, 6],marker=5)
g.line([6, 0],marker=6)

g.surface([0, 1, 2, 3, 4,5,6])

# Forces and boundary conditions
force = [-1e6,9,2]
bmarker = 5
eq=[0,0]

# Material parameters
mp = {'E':210e9,'nu':0.3,'eps_y':0 }
materialFun = mrs.Bilinear

# Settings, Objective function and Optimisation routine
ep=[1,True,2]
settings = {'volFrac':0.3,'meshSize':0.03,'rmin':0.03*0.7,'changeLimit': 0.0,'SIMP_const':3}
ObjectFun = ofs.Displacement
OptFun = opt.OC

# Calling the optimisation
PyTOpt.Main(g, force, bmarker, mp, ep, materialFun, ObjectFun, OptFun,settings,eq,maxiter=150)
```

The first thing required is to import at least two of the five modules shown in Listing A.2. Only the first two imports are required if the user contributes with a self made material model, objective function and optimisation algorithm. How these are defined is described later in this appendix chapter.

Listing A.2: Needed imports for the optimisation.

```
# Importing Modules
import calfem.geometry as cfg
import Pytopt.PyTOpt as PyTOpt
from Pytopt import Material_Routine_Selection as mrs
from Pytopt import Object_Func_Selection as ofs
from Pytopt import Optimisation as opt
```

The next step is to define a design domain and apply forces and boundary conditions. In this example, the design domain of the cantilever beam is used. The geometry definition can be seen in Listing A.3. The geometry commands are described in detail at `https://calfem-for-python.readthedocs.io/en/latest/calfem_mesh_guide.html#defining-geometry` and will not be explained in detail here.

Listing A.3: The definition of the design domain sought to optimise.

```
# Creating geometry
g = cfg.Geometry()

g.point([0,0])
g.point([1,0])
g.point([1,0.4],marker=9)
g.point([1,0.8])
g.point([0,0.8])
g.point([0,0.5])
g.point([0,0.3])


g.line([0, 1],marker=0)
g.line([1, 2],marker=1)
g.line([2, 3],marker=2)
g.line([3, 4],marker=3)
g.line([4, 5],marker=4)
g.line([5, 6],marker=5)
g.line([6, 0],marker=6)

g.surface([0, 1, 2, 3, 4,5,6])

# Forces and boundary conditions
force = [-1e6,9,2]
bmarker = 5
eq=[0,0]
```

The user also have to define the boundary markers and the force marker. Both these types of markers are needed to define the forces and boundaries. As seen in Listing A.3, the force is applied at the third point, which is why that point has obtained a marker. In this case, that marker is 9. The force list describes the force that is going to be applied. The first value is the magnitude of the force. The marker that signifies at which location the force should be applied is the second input. The last input is the direction of the force. It is either defined with 1, which means x-direction, or 2 which means y-direction. The force and boundaries can be applied at a point or on a line. The degrees of freedom defined in `bmarker` will have a prescribed displacement of zero. This means that nodes along line six in this example will be fixed, denoted as marker 5. PyTOpt also supports body forces. The variable `eq` has two

parameters, the first is body forces in x-direction and the second is body forces in y-direction. The parameters have the unit $\frac{N}{m^3}$.

The third step is to define the material parameters and the material model to use, see Listing A.4. The dictionary `mp` requires Young's modulus and Poisson's ratio for the program to run. In this example the material model was chosen as *Bilinear* which also needs the yield strain as an additional material parameter in `mp`.

Listing A.4: The definition of material and material model.

```
# Material parameters
mp = {'E':210e9,'nu':0.3,'eps_y':0 }
materialFun = mrs.Bilinear
```

Finally, the settings, choice of objective function and optimisation algorithm are defined in Listing A.5 before calling PyTOpt. The list `ep` is the element parameters where the first parameter is the element thickness. The second parameter is controlling whether a linear or nonlinear FE-solver will be utilised. If *True* a linear solver will be used. The last parameter in `ep` is the choice of element type. Triangle elements are used if the setting is set to 2 and quadrilateral elements are used if it is set to 3.

Listing A.5: The settings for the optimisation algorithm and the calling of the optimisation.

```
# Settings, Objective function and Optimisation routine
ep=[1,True,2]
settings = {'volFrac':0.3,'meshSize':0.03,'rmin':0.03*0.7,'changeLimit': 0.0,'SIMP_const':3}
ObjectFun = ofs.Displacement
OptFun = opt.OC

# Calling the optimisation
PyTOpt.Main(g, force, bmarker, mp, ep, materialFun, ObjectFun, OptFun,settings,eq,maxiter=150)
```

A `settings` dictionary is recommended to define the appropriate settings for the example. If the settings are totally or partially omitted, a standard setting will be used which might be sub optimal depending on the problem at hand. Examples on settings available are shown in Table A.1.

volfrac	Maximum volume fraction
meshSize	average element size
rmin	the filter parameter
changeLimit	Optimisation breaks when this limit is reached
SIMP_const	SIMP constant

Table A.1: Available settings for the dictionary.

The next thing to define is the objective function, `ObjectFun`. The objective function can be chosen from the two included in the PyTOpt library or created by a user. A new objective function is limited to have these variables as input

- Number of elements
- Element parameters
- Element type
- Element coordinates
- Tangent stiffness matrix
- Body forces

- Displacements
- Element degrees of freedom matrix
- External Force vector due to body forces
- External Force vector
- SIMP constant
- Design variables
- Sensitivity of the objective function
- Sensitivity of the residual
- Free degrees of freedom
- Global stiffness matrix

The output should be the objective function value and its sensitivity.
Finally, PyTOpt needs to know what optimisation algorithm that is going to be used. `OptFun` can use either Optimality Criteria (OC) or Methods of Moving Asymptotes (MMA) from the 'Optimisation' module or defined by the user. Inputs for a optimisation algorithm must be design variables, maximum volume fraction usable of the structure, objective function value, sensitivity of the objective function, current iteration and scaled element areas. As output one should get the updated design variables. The final step is to call the program.

A.2 Computer Specifications

Processor: Intel(R) Core(TM) i5-6198DU CPU @ 2.30GHz 2.40 GHz RAM: 4GB Systemtype: 64-bits operative system, x64-based processor

B

Simulation settings

B.1 Optimisation Parameters and Convergence

B.1.1 OC

Table B.1: Settings during OC parameter *Step* investigation.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.02
Volume Fraction	0.3
rMin	0.014
SIMP	3
Element type	Triangles
Object Function	Compliance
Material Model	Elastic
Optimiser	OC(Step: 0.5=>0.05)
Force	10^6 N

B.1.2 MMA

Table B.2: Settings during MMA parameter *Move* investigation.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.02
Volume Fraction	0.3
rMin	0.014
SIMP	3
Element type	Triangles
Object Function	Compliance
Material Model	Elastic
Optimiser	MMA(Move: 0.2=>0.01)
Force	10^6 N

B.2 Linear Topology Optimisation

Table B.3: Settings for the optimisation with linear elastic material model.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.01
Volume Fraction	0.3
rMin	0.007
SIMP	3
Element type	Triangles
Object Function	Compliance
Material Model	Elastic
Optimiser	OC(Step:0.1, Damping:0.5)
	MMA(Move: 0.05)
Force	10^6 N

B.3 Bilinear Topology Optimisation

Table B.4: Settings for the optimisation with bilinear material model.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.01
Volume Fraction	0.3
rMin	0.009
SIMP	3
Element type	Triangles
Object Function	Compliance
Material Model	Bilinear
Optimiser	OC
Force	10^6 N
Yielding Strain ε_y	0

B.4 Optimisation with line load and multiple boundaries

Table B.5: Settings during the bridge optimisation.

Simulation Settings	
Design domain	Bridge
Mesh Size	0.4
Volume Fraction	0.3
rMin	0.36
SIMP	3
Element type	Quadrilateral
Object Function	Compliance
Material Model	Elastic
Optimiser	OC (Step: 0.1 Damping: 0.5)
Line Load	$7 \cdot 10^4$ N/m

B.5 Body Forces

Table B.6: Settings during the body force investigation.

Simulation Settings	Figures 5.10a and 5.10b	Figures 5.10c and 5.10d
Design domain	Beam in Compression	
Mesh Size	0.03	
Volume Fraction	0.3	
rMin	0.021	
SIMP	3	
Material Model	Elastic	
Optimiser	OC (Step: 0.1 Damping: 0.5)	OC (Step: 0.02 Damping: 0.05)
Force	$4 \cdot 10^5$ N	

B.6 Dependency of Initial Density Distribution

Table B.7: Settings during the dependency of initial density distribution investigation.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.04
Volume Fraction	0.3
rMin	0.028
SIMP	3
Material Model	Elastic
Optimiser	OC(Step: 0.1, Damping: 0.5)
Force	10^6 N

B.7 ANSYS comparison

Table B.8: Settings during the start dependency investigation.

Simulation Settings	
Design domain	L-Beam
Mesh Size	0.005
Volume Fraction	0.3
rMin	0.0035
SIMP	3
Material Model	Elastic
Object Function	Compliance
Optimiser	OC(Step: 0.1, Damping: 0.5)
Force	10^6 N

B.8 Numerical Sensitivity Analysis

Table B.9: Optimisation settings during the numerical sensitivity analysis.

Simulation Settings	
Design domain	Cantilever Beam
Mesh Size	0.4
Volume Fraction	0.5
rMin	0.105
SIMP	3
Material Model	Elastic (With nonlinear solver)
Optimiser	OC(Step: 0.1, Damping 0.5)

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY