CHALMERS | GÖTEBORG UNIVERSITY

MASTER'S THESIS

Construction of an Optimal Linkage Arm Using Structural Optimization

Einar Guðfinnsson and Nikulás Árni Sigfússon

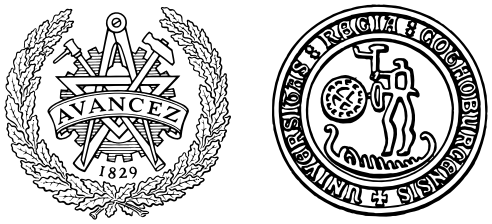
Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg Sweden 2006

Thesis for the Degree of Master of Science

Construction of an Optimal Linkage Arm Using Structural Optimization

Einar Guðfinnsson and Nikulás Árni Sigfússon

CHALMERS | GÖTEBORG UNIVERSITY



Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, May 2006

Matematiskt centrum
Göteborg 2006

Abstract

We use structural optimization methods to determine the design of a linkage arm such that its maximal stress is minimized. The methods include topology optimization (TO) and shape optimization (SO). In the TO part, we used a software program called TO++. By using approximating boundary conditions, we were able to solve the Minimum Compliance Problem and to develop ideas for improving the SO model. In the SO part, the software program Comsol Multiphysics (Femlab) was used, along with the optimization programs NEWUOA and MultiOb. Various approaches were used to determine the parameters of the SO model. On one hand, the original linkage arm with some simplifications was used as a model, and on the other hand, our results from the TO part were used to motivate a new construction of a linkage arm.

A major problem that we encountered was that the objective function in question is numerically very sensitive, which often resulted in poor estimates and in mesh dependent solutions. Furthermore, applying simplified boundary conditions can result in the lack of existence of a solution to the problem posed. Thus, we suggest the use of another objective function instead. A natural choice is to minimize the compliance of the structure.

Acknowledgments

First, we would like to give special thanks to our supervisors, Professor Michael Patriksson, and Ph.D. Ann-Brith Strömberg for frequent meetings and constant support. We would also like to give special thanks to Erik Höök at Fraunhofer-Chalmers Research Centre for Industrial Mathematics (FCC) for some great advice and help on the mechanical part of the project. The author of TO++, Ph.D. Thomas Borrvall, deserves extra credit for all his patience and assistance. His help made it possible for us to run both the 2D and the 3D version of his program. We would like to thank Professor Anders Klarbring, Linköping University, for the free copy of notes on structural optimization, which proved to be an excellent preparation for the project. We would like to thank Gunnar Sjödin for feedback on the original linkage arm. Many thanks go to Professor Ivar Gustavsson, Director of the International Masters Programme in Engineering Mathematics and Professor Leif Lundkvist, Director of studies at the Electrical Engineering department at Chalmers Chalmers University. We have enjoyed the facility at FCC and the company of Uno Nävert and all the staff here at FCC. Everyone has been more than willing to help us with various technical problems we encountered; Jenny Ekenberg, Anders Ålund, Henning Schmidt and Frederik Edelvik have all been very helpful. We want to thank our spouses, Ásta Herdís Hall and Sigrún Steingrímsdóttir, for all their love and patience and our friends and families in Iceland for all their support. Finally, we want to thank Sigurveig Einarsdóttir, who was born in March and was kind enough to let us finish our work here in Göteborg before heading back to Iceland.

Contents

1 Introduction

The search for better designs and more efficient ways to produce goods in the industry has a large potential. Production time, cost, and quality of the products play a vital role in how successful the manufacturers are. With constantly increasing computer power, various new virtual models have been developed, resulting in a more efficient production process compared with time-consuming and often expensive physical testing. International competition requires products at lower prices and using computer simulation to virtually test products can save both time and money. It is of great importance to integrate optimization and simulation to secure efficiency of mathematical optimization.

This thesis is a part of an internal project at Fraunhofer Chalmers Research Centre for Industrial Mathematics (FCC) [17]. One of the objectives of the overall project is to develop an optimization tool that integrates multi-criteria optimization and simulation. We have investigated and attempted to optimize the design of a linkage arm. The final aim is to use our work to develop a mechanics optimization tool.

The linkage arm that we used as a test case, is used to transfer power from the wheels of a vehicle to its chassis. It is bent and is exposed to loads in its length direction. The objective is to find a structure of the linkage arm such that its maximal stress is minimized, as high stress can cause fractures which can easily break the arm in two. For this purpose, we use methods of structural optimization. Some simplifying assumptions are made, e.g. on the amount of material that can be used. Also, the final design should be feasible to produce. Figure 1 shows what the linkage arm, which is used today, looks like, with color-coded stresses for a certain load case.

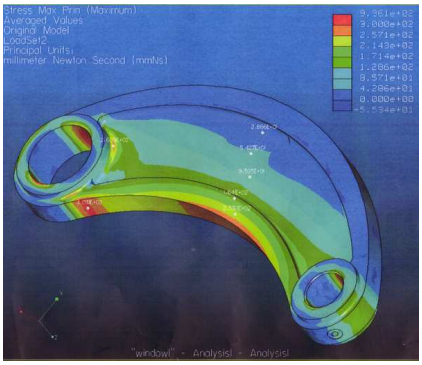


Figure 1: The original linkage arm

The remainder of this thesis is organized as follows.

In Section 2, the concepts of stresses and strains are introduced for simple objects, and then generalized to two and three dimensions. The physical laws and the mathematical relations are essential for the calculations that we perform.

The models derived in Section 2 have no analytical solution in general. In order to get an approximative solution, some numerical calculations have to be performed. In Section 3, we discuss the Finite Element Method (FEM) and compare two mesh types. We used the theory presented in this section to implement a FE-solver and a pseudo-code for this solver is given in the appendix. The results in this thesis are not based on this solver, but a comparison with other FE-solvers gave satisfactory results.

In Section 4, we discuss optimization. We start by general optimization theory, and then move to a field called structural optimization, where optimization and mechanics are combined. We finally discuss the two main subfields of structural optimization, namely shape optimization and topology optimization.

In Section 5 we describe our models for the two different optimization approaches in Section 4.

In Section 6, we discuss the programs that we used for our calculations. We start by Matlab and Femlab, and then move to the shape optimization programs NEWUOA and MultiOb. Finally, we discuss the topology

optimization programs. These are a 99-line Matlab code, which finds the structure of a cantilever beam such that its compliance is minimized, and TO++, where general topology optimization problems can be modeled and solved.

In Section 7, we display some of our results, first for shape optimization and then for topology optimization. We end this section by displaying a possibly improved shape optimization model, based on the results from the topology optimization.

Finally, in Section 8, we summarize and discuss our main conclusions and list possible future work.

This is the first part of a longer term project with the goal of maximizing the fatigue limit of the object.

2 Mechanics of Materials

In this section the basic theory of mechanics of materials is introduced, emphasizing the concept of stress and strain and their relation in an isotropic material. Apart from the simple spring example stated below, the authors had no knowledge of the material presented in this section prior to this project.

2.1 A simple model

Figure 2 shows a spring of length x. When a small force, dF, is applied to the spring, it increases in length and its new length is $x + dx$.

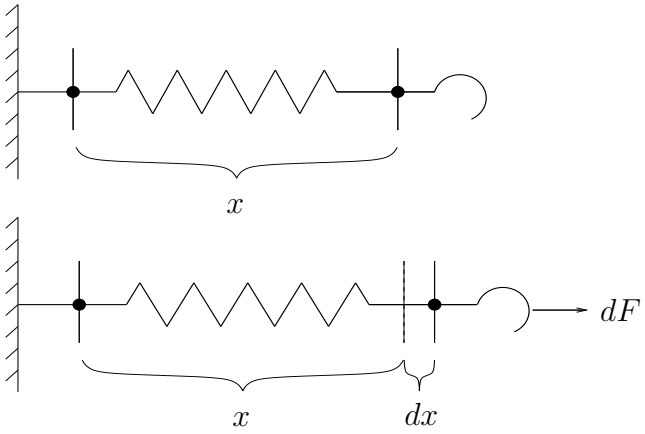


Figure 2: A schematic picture of a spring

A well known approximation, known as Hooke's law, states that there is a constant, k, depending on the material in the spring, such that the relation

$$dF = k \cdot dx \tag{1}$$

holds for sufficiently small forces, dF.

2.2 Introduction to stresses and strains

In order to introduce the concept of stresses and strains, it is common to look at a prismatic bar subjected to axial forces (see for example [4]). A schematic figure is shown in Figure 3. Here, P is the axial force, L is the length of the bar, A is its crosssectional area, and δ denotes the elongation of the bar subject to the force. In practice, we have $\delta \ll L$.

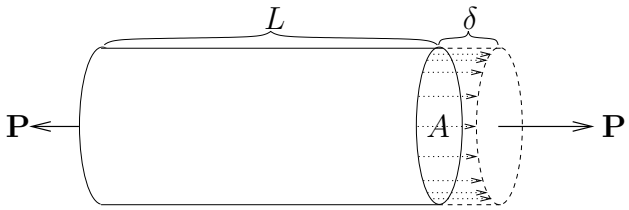


Figure 3: A bar in tension

The *stress* and (axial) *strain* are denoted σ and ε, respectively, and are given by

$$\sigma = \frac{P}{A} \qquad \text{and} \qquad \varepsilon = \frac{\delta}{L}. \tag{2}$$

We note, that the unit for stress is $[\mathrm{N/m^2}] = [\mathrm{Pa}]$, but the strain is without a unit. For small values of P we have a linear relation between stress and strain, namely,

$$\sigma = E\varepsilon, \tag{3}$$

where E is a constant, depending on the material of the bar. Note the resemblance between equation (1) and equation (3). Equation (3) is also known as $Hooke's\ law$, and E is called $Young's\ modulus$. For steel, E is between 190 and 210 GPa.

Figure 3 only shows the bar getting longer, but in reality the cross sectional area also gets smaller. This is shown schematically in Figure 4.

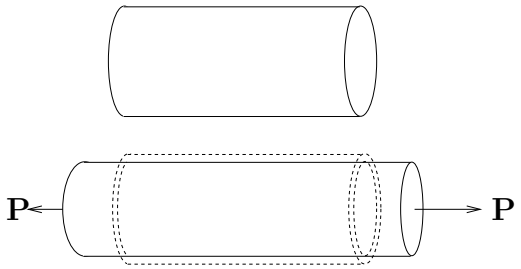


Figure 4: A bar without an applied load (above) and how it deforms under axial forces (below)

$Poisson's\ ratio$ is denoted by ν, and is defined as the dimensionless ratio

$$\nu = -\frac{\text{lateral strain}}{\text{axial strain}} = -\frac{\varepsilon'}{\varepsilon}. \tag{4}$$

For metals, ν varies between 0.25 and 0.35, for rubber it is close to 0.5, for concrete it is between 0.1 and 0.2, and for cork is is close to zero. Figure 5 shows how a bar deforms, with the left end fixed and tensile axial force applied on the right end. The figure is made with a program called Femlab (see Section 6.1). The colors of the bar indicate stress, where red is high stress and blue is low stress. One can see that the stress is the same along the bar, except at the right end. The bar gets longer and thinner.

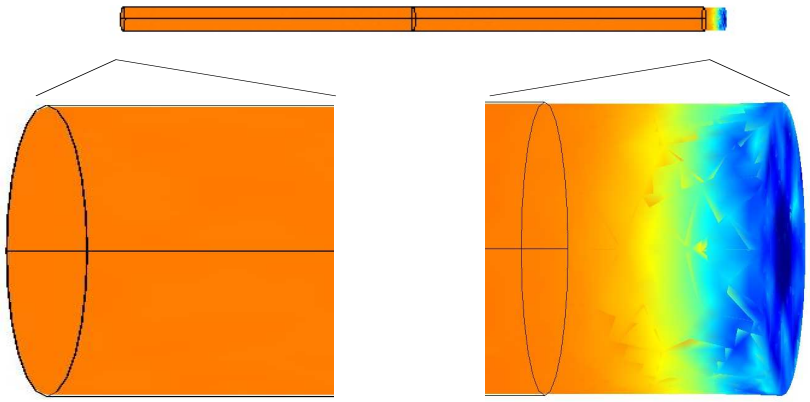


Figure 5: Experimenting with Femlab

2.3 Generalization of stresses and strains

Here, the definition of stress and strain, defined for simple objects in equation (2), and their relation stated in equation (3), will generalized to arbitrary 3-dimensional objects. Some basic assumptions are made, for example that the object in question is sufficiently regular. Many books exist on this subject. We have mainly followed [9], but [13] has also been useful.

For a 3-dimensional solid object, which is subjected to some force, we distinguish between forces which are applied to volume and those which are applied to areas. We denote a force, which is applied to a volume, by the letter $\mathbf{b} = [b_x\ b_y\ b_z]'$ and a force, which is applied to an area, by the letter $\mathbf{t} = [t_x\ t_y\ t_z]'$. We call $\mathbf{b}$ a *body force* and $\mathbf{t}$ is called a *traction force*.

In Figure 6, a force $d\mathbf{P}$ is applied to an area of size dA, where dA is a part of a surface around a point $p \in \mathbb{R}^3$. This surface can either be a part of the boundary of a solid object or a 2D cross-section. We have that $\mathbf{n} = [n_x\ n_y\ n_z]'$ denotes the unit normal of dA at p with outward direction.

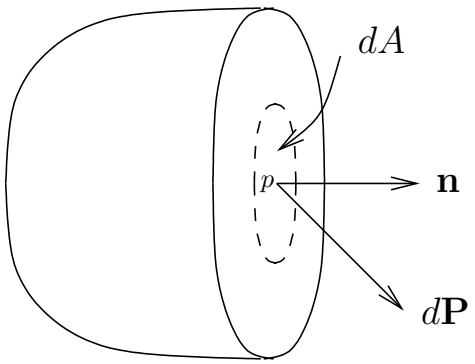


Figure 6: A traction force

The traction force, $\mathbf{t}$, at p is defined as the limit value

$$\mathbf{t} = \lim_{dA \to 0} \frac{d\mathbf{P}}{dA}.$$

We note that this definition depends on the surface which p lies in. In order to be able to work with arbitrary surfaces we only need to evaluate the traction force for three special types of areas dA. These areas and the traction components are shown in Figure 7, where dA lies in the yz-plane, zx-plane, and the xy-plane, respectively.

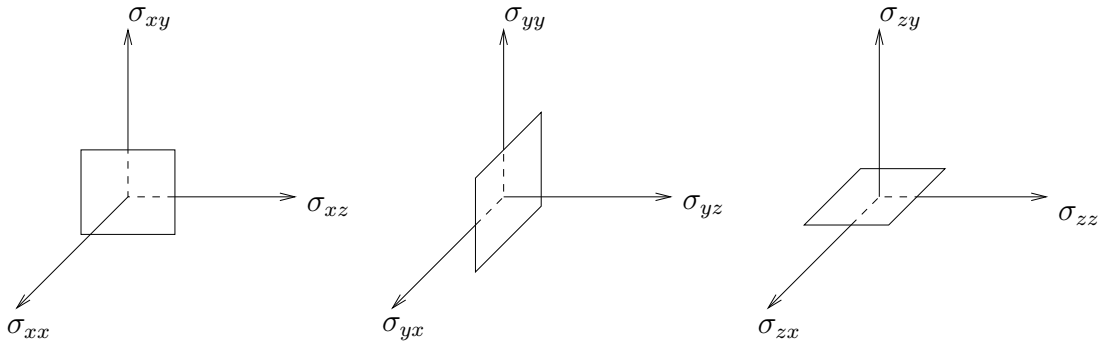


Figure 7: Components of the stress tensor

The traction forces for these planes are denoted by

$$\boldsymbol{\sigma_x} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{xy} \\ \sigma_{xz} \end{bmatrix}, \qquad \boldsymbol{\sigma_y} = \begin{bmatrix} \sigma_{yx} \\ \sigma_{yy} \\ \sigma_{yz} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\sigma_z} = \begin{bmatrix} \sigma_{zx} \\ \sigma_{zy} \\ \sigma_{zz} \end{bmatrix},$$

respectively, and the matrix, $\mathbf{S}$, defined by

$$\mathbf{S} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$

is called the *stress tensor*. It can be shown that $\mathbf{S}$ is symmetric.

2.3.1 Stress

The components σ_{xx}, σ_{yy}, and σ_{zz} of the stress tensor are called *normal stresses* and $\sigma_{xy} = \sigma_{yx}, \sigma_{yz} = \sigma_{zy}$, and $\sigma_{zx} = \sigma_{xz}$ are called *shear stresses*. Knowing the stress tensor gives us the possibility to calculate an arbitrary traction force, since we have the relation

$$\mathbf{t} = \mathbf{S}\mathbf{n}. \tag{5}$$

We now define the *stress vector*, $\boldsymbol{\sigma}$, by

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix}.$$

The following partial differential equation

$$\tilde{\nabla}^T \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0}, \tag{6}$$

where the differential operator $\tilde{\nabla}$ is defined as

$$\tilde{\nabla} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix}, \tag{7}$$

is the *equilibrium condition* for the body in question. This formula can be derived from equation (5) and Gauss' divergence theorem. For details, see e. g. [9].

2.3.2 Strains

Let $(x, y, z) \in \mathcal{D}$, where $\mathcal{D} \subset \mathbb{R}^3$ is a 3-dimensional domain or object. When a load is applied to $\mathcal{D}$, the point (x, y, z) moves to $(x + u_x, y + u_y, z + u_z)$. The vector

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

is called the *displacement vector*.

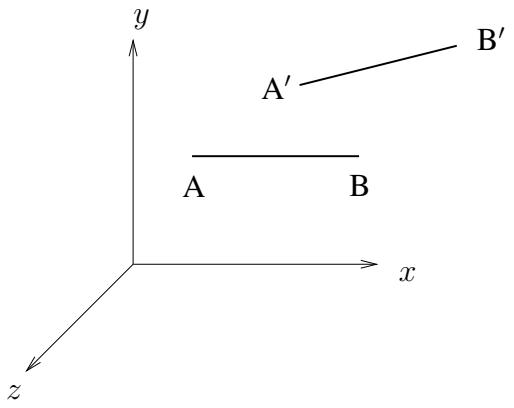


Figure 8: Deformation of the line |AB|

Let $|AB|$ be a line which is parallel to the x-axis, as shown in Figure 8, and let $|A'B'|$ be the same line after displacement resulting from some load.

We define the strain ε_{xx} by

$$\varepsilon_{xx} = \frac{|A'B'| - |AB|}{|AB|}.$$

It can be shown that

$$\frac{|A'B'| - |AB|}{|AB|} = \frac{\partial u_x}{\partial x}.$$

By analogous definition of ε_{yy} and ε_{zz}, we have

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x}, \qquad \varepsilon_{yy} = \frac{\partial u_y}{\partial y} \qquad \text{and} \qquad \varepsilon_{zz} = \frac{\partial u_z}{\partial z}.$$

The strains, $\varepsilon_{xx}, \varepsilon_{yy}$ and ε_{zz} are called $normal\ strains$.

Now let |AB| be parallel to the x-axis, and |AC| be parallel to the y-axis, as shown in Figure 9.

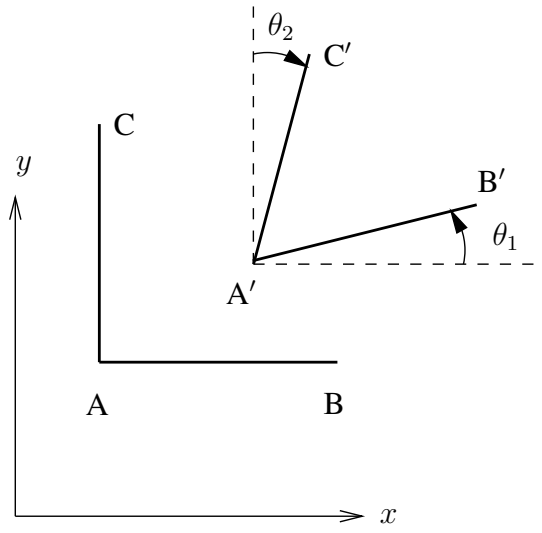


Figure 9: Deformation of the lines |AC| and |AB|

We define the strain γ_{xy} to be

$$\gamma_{xy} = \theta_1 + \theta_2,$$

where θ_1 and θ_2 are the angles shown in Figure 9. In practice, $\theta_i, i = 1, 2$, are small and we can approximate $\sin\theta_i$ by $\theta_i, i = 1, 2$. Using this approximation the following equation can be derived.

$$\gamma_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}.$$

By an analogous definition of γ_{xz} and γ_{yz}, we have that

$$\gamma_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}, \qquad \gamma_{xz} = \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}, \qquad \text{and} \qquad \gamma_{yz} = \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}.$$

The strains γ_{xy}, γ_{xz} and γ_{yz} are called *shear strains*.

The vector

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix}$$

is called the *strain vector*. It is easy to verify that the equation

$$\boldsymbol{\varepsilon} = \tilde{\nabla}\mathbf{u} \tag{8}$$

holds.

2.4 Constitutive relation

The following relation

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}, \tag{9}$$

where

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2 \end{bmatrix},$$

is called the *generalized Hooke's law* in three dimensions. Here, E is Young's modulus, and ν is *Poisson's ratio*, which were defined in equation (3) and equation (4), respectively. $\mathbf{D}$ is called the *constitutive matrix*. This relation is derived assuming isotropic material, i.e. if $\mathbf{D}$ is independent of coordinate system.

2.5 Reduction to 2 dimensions

When we have plain strain, certain components of the stress and strain vectors become zero. In these cases we redefine $\boldsymbol{\sigma}$, $\boldsymbol{\varepsilon}$ and $\tilde{\nabla}$ as follows:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix}, \quad \tilde{\nabla} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}.$$

3 The Finite Element Method (FEM)

In Section 2, the differential equation of equilibrium,

$$\tilde{\nabla}^T \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0}, \tag{10}$$

was stated (see equation (6)). Equation (10) is an example of a non-linear partial differential equation (PDE), which in general has no solution on a closed form. Various methods exist to find an approximate solution, including the Boundary Element Method (BED), the Finite Difference Method (FDM), and the Finite Element Method (FEM). FEM was used in our calculations. Prior to working on this thesis, the authors had little or no knowledge of FEM nor experience in solving PDE:s numerically. In this section, we will discuss some basic concepts of FEM in 2D and 3D. At the end of the section, equation (10) is rewritten in matrix form. More advanced discussions can be found in the literature, for example in [9] and [13].

3.1 Meshes and shape functions

The main idea of FEM is to divide the spatial domain, which the differential equation is defined on, into smaller subdomains, called $elements$, which satisfy certain criteria. We then seek an approximate solution to the differential equation above, having certain simple properties on each element. The elements are generally polygons in 2D and polyhedra in 3D. The elements are mutually disjoint, except at their boundaries, and their union is the original domain, at least approximately. In 2D, elements are connected to each other by their sides, and in 3D, they are connected by their faces. We call two elements $neighbors$ if they share a side in 2D or share a face in 3D.

The most common approaches for dividing a 2D domain is to divide it into rectangles or triangles. In 3D, one uses boxes and tetrahedra. Each triangle or rectangle has corner points, called $nodes$, which we denote by n_e. Figure 10 shows an example of a triangular division of the unit square in 2D, along with an enumeration of the elements and nodes. Each triangle has exactly three adjacent nodes.

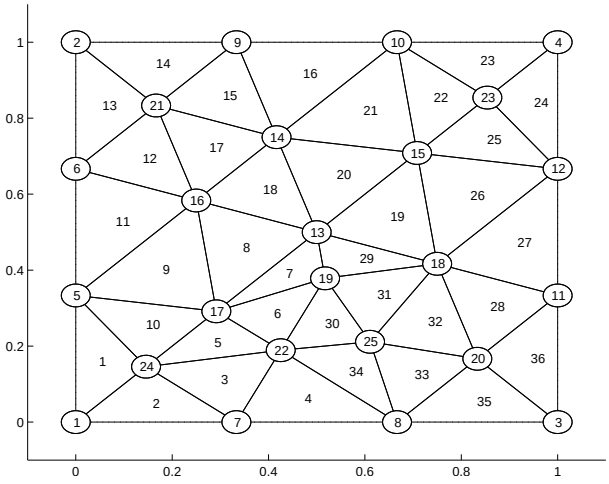


Figure 10: Triangular mesh in 2D with $n = 25$ nodes and $m = 36$ elements

We say that an element, e, is $adjacent$ to a node i if e has i as an endpoint. For example, the nodes $i = 13$, $i = 17$, and $i = 19$ are the adjacent nodes to element $e = 7$ in Figure 10.

A $shape\ function$ is a continuous function, defined on the domain, that has the value 1 at some node point, i, and zero on all elements not adjacent to i. In practice, it should also have some simple properties on the elements adjacent to i. When we talk about a shape function on a particular element, e, with respect to some end point, i, we mean the shape function which takes the value 1 at i restricted to the element, e.

The procedure of dividing the domain into smaller elements is called $meshing$. Every element, e, has n_e nodal points, and thus n_e corresponding shape functions, $N_1^e, N_2^e, \ldots, N_{n_e}^e$. Before we continue further, we introduce a notation used in 3D. One can easily reduce this to 2D.

3.1.1 Elements - local notation

1. Let $\tau \in \{x, y, z\}$, and $i \in \{1, 2, \ldots, n_e\}$. Then we denote by
$$u_{\tau i}$$
the displacement along the τ-axis of the point i.

2. We define the matrix of functions
$$\mathbf{N^e} = \begin{bmatrix} N_1^e & 0 & 0 & N_2^e & 0 & 0 & \cdots & N_{n_e}^e & 0 & 0 \\ 0 & N_1^e & 0 & 0 & N_2^e & 0 & \cdots & 0 & N_{n_e}^e & 0 \\ 0 & 0 & N_1^e & 0 & 0 & N_2^e & \cdots & 0 & 0 & N_{n_e}^e \end{bmatrix}.$$

3. We define the $element\ displacement\ vector$
$$\mathbf{a}^e = [u_{x1} \quad u_{y1} \quad u_{z1} \quad u_{x2} \quad u_{y2} \quad u_{z2} \quad \cdots \quad u_{xn_e} \quad u_{yn_e} \quad u_{zn_e}]'.$$

4. We approximate the function $\mathbf{u}$ defined on the element e by
$$\mathbf{u} = \mathbf{N}^e\mathbf{a^e}.$$

5. We define
$$\mathbf{B}^e = \tilde{\nabla}\mathbf{N}^e,$$
where the differential operator $\tilde{\nabla}$ is defined in equation (7). The dimension of $\mathbf{B}^e$ is $6 \times 3n_e$.

3.1.2 Global notation

1. Let $\tau \in \{x, y, z\}$, and $i \in \{1, 2, \ldots, n\}$, where n is the total number of nodal points in the mesh. Then we denote by
$$u_{\tau i}$$
the displacement along the τ-axis of the point i.

2. We define the matrix of functions
$$\mathbf{N} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \cdots & N_n & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \cdots & 0 & N_n & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \cdots & 0 & 0 & N_n \end{bmatrix}.$$

3. We define the $displacement\ vector$
$$\mathbf{a} = [u_{x1} \quad u_{y1} \quad u_{z1} \quad u_{x2} \quad u_{y2} \quad u_{z2} \quad \cdots \quad u_{xn} \quad u_{yn} \quad u_{zn}]'.$$

4. We approximate the function $\mathbf{u}$ defined on the whole domain $\mathcal{D} \subset \mathbb{R}^3$ by
$$\mathbf{u} = \mathbf{Na}.$$

5. We define
$$\mathbf{B} = \tilde{\nabla}\mathbf{N},$$
where the differential operator $\tilde{\nabla}$ is defined in equation (7). The dimension of $\mathbf{B}$ is $6 \times 3n$.

In Section 3.1.3 and Section 3.1.4 we discuss two types of meshing approaches, which we use in the analysis in this thesis. The former uses regular rectangular squares in two dimensions and boxes in three dimensions. The latter uses triangles in two dimensions and tetrahedra in three dimensions. Both methods have their benefits and faults.

3.1.3 Regular rectangular mesh and corresponding shape functions

In Figure 11, a regular rectangular element in 3D is shown.

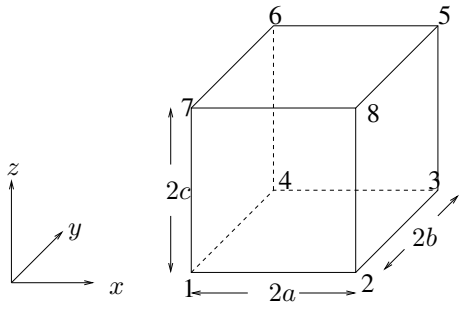


Figure 11: A rectangular element in three dimensions

On this element we have eight shape functions, defined for each node point in the element. The general form of the shape function is given by

$$N_i^e(x) = a_1 + a_2x + a_3y + a_4z + a_5xy + a_6yz + a_7zx + a_8xyz.$$

Other more complex forms of N_i^e are also used. Their exact expressions are given below; the left column is for arbitrary a, b, and c, and the right one is when $a = b = c = 1/2$.

$$\begin{aligned}
N_1^e &= -(x-2a)(y-2b)(z-2c)/(8abc) & N_1^e &= -(x-1)(y-1)(z-1)\\
N_2^e &= x(y-2b)(z-2c)/(8abc) & N_2^e &= x(y-1)(z-1)\\
N_3^e &= -xy(z-2c)/(8abc) & N_3^e &= -xy(z-1)\\
N_4^e &= (x-2a)y(z-2c)/(8abc) & N_4^e &= (x-1)y(z-1)\\
N_5^e &= xyz/(8abc) & N_5^e &= xyz\\
N_6^e &= -(x-2a)yz/(8abc) & N_6^e &= -(x-1)yz\\
N_7^e &= (x-2a)(y-2b)z/(8abc) & N_7^e &= (x-1)(y-1)z\\
N_8^e &= -x(y-2b)z/(8abc) & N_8 &= -x(y-1)z
\end{aligned}$$

Using a regular rectangular mesh has been quite popular, especially in topology optimization, which is one of the main application in this thesis. Apart for being easy to visualize and explain, it usually speeds up calculations significantly. On the other hand it can be quite ill adapted to domains where the boundary has a high curvature. In Figure 12 we see examples of 2D and 3D meshes. The 2D mesh has $40 \times 12 = 480$ elements while the 3D one has $40 \times 6 \times 12 = 2880$ elements.

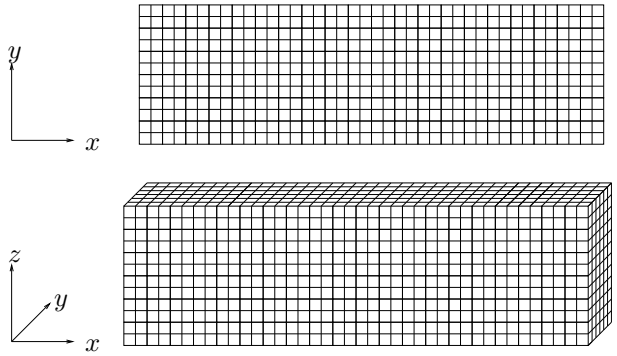


Figure 12: Rectangular meshes in 2D and 3D

3.1.4 Triangular meshes

In Figure 10, we saw an example of a triangular mesh in 2D. This mesh is constructed by the $Delauny$ triangulation. Matlab/Femlab (see Section 6.1) uses this algorithm to generate meshes.

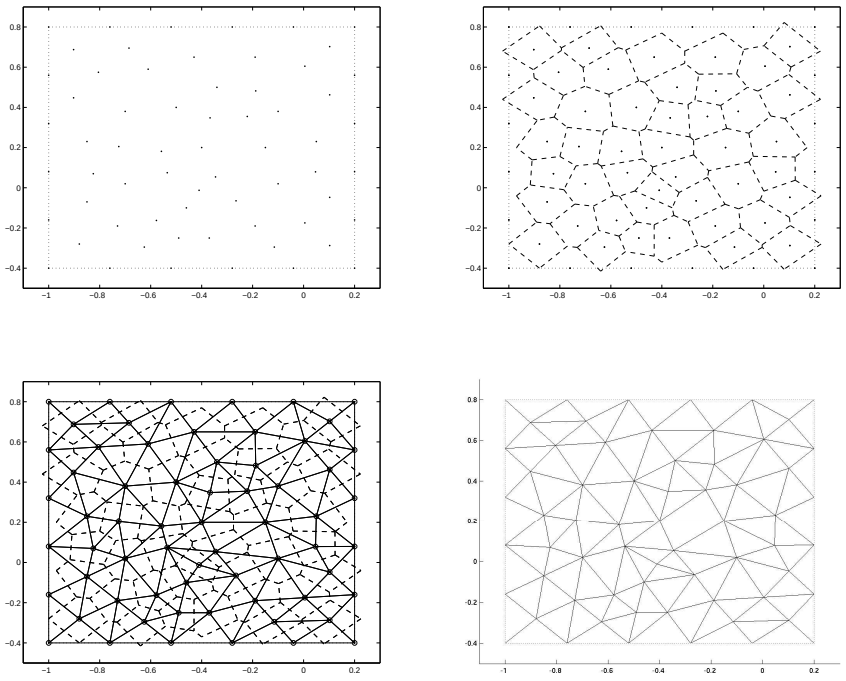


Figure 13: Illustration of Delauny's algorithm

Delauny's algorithm generates points in a somewhat random fashion on the domain to be triangulated, but with certain restrictions, which the user can control. These restriction include e.g. maximum and minimum distances between the points. An example of this is shown in the upper left image in Figure 13. For each of the generated points, j, the set of all points in the domain which are closer to j than any other generated point form a polygon, which is called $Voronoi\ polygons$. These polygons divide the domain into disjoint sets. An example of this is shown in the upper right image in Figure 13. In order to get a Delauny triangulation, one draws a line perpendicular to each edge of every Voronoi polygon connecting two of the originally generated points in the polygons which has the current edge included. See an example on the bottom left in Figure 13. The final result (the mesh) is then obtained by using the triangles which are generated by this method. See the bottom right image in Figure 13 for an example.

3.1.5 2D triangular shape functions

For a triangular mesh in 2D, every element, $e = 1, \ldots, m$, has three nodal points, p_1^e, p_2^e, and p_3^e, with coordinates $(x_1^e, y_1^e), (x_2^e, y_2^e)$, and (x_3^e, y_3^e), see Figure 14. We will sometimes omit e when there is no danger of misunderstanding.

The corresponding shape functions, N_1^e, N_2^e and N_3^e are given by

$$N_1^e = \frac{1}{2A_e}|x_2y_3 - x_3y_2 + (y_2 - y_3)x + (x_3 - x_2)y|,$$
$$N_2^e = \frac{1}{2A_e}|x_3y_1 - x_1y_3 + (y_3 - y_1)x + (x_1 - x_3)y|,$$
$$N_3^e = \frac{1}{2A_e}|x_1y_2 - x_2y_1 + (y_1 - y_2)x + (x_2 - x_1)y|,$$

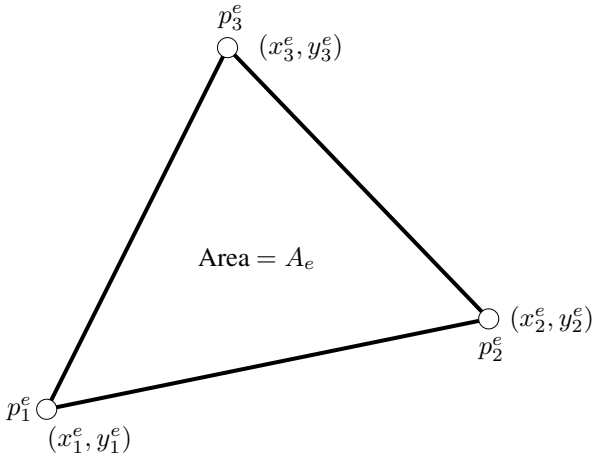


Figure 14: 2 dimensional element

where A_e is the area of element e. Note that for $j, k = 1, \ldots 3$, we have

$$N_j^e(x_k, y_k) = \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{otherwise.} \end{cases}.$$

We define

$$N^e = [N_1^e \; N_2^e \; N_3^e] \qquad \text{and} \qquad B^e = \tilde{\nabla} N^e.$$

Figure 15 shows how two elements, e_1, and e_2, can be enumeraed.

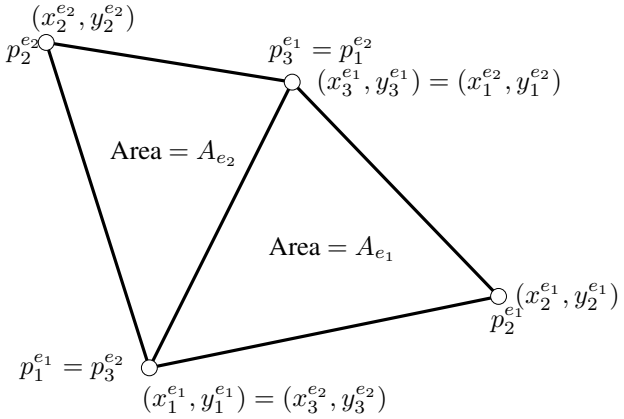


Figure 15: Two adjacent 2D elements

3.1.6 Shape functions for a 3D element

In 3D, every tetrahedron element, $e, e = 1, \ldots, m$, has four nodal points, p_1^e, p_2^e, p_3^e, and p_4^e, with coordinates $(x_1^e, y_1^e), (x_2^e, y_2^e), (x_3^e, y_3^e)$, and (x_4^e, y_4^e), see Figure 16.

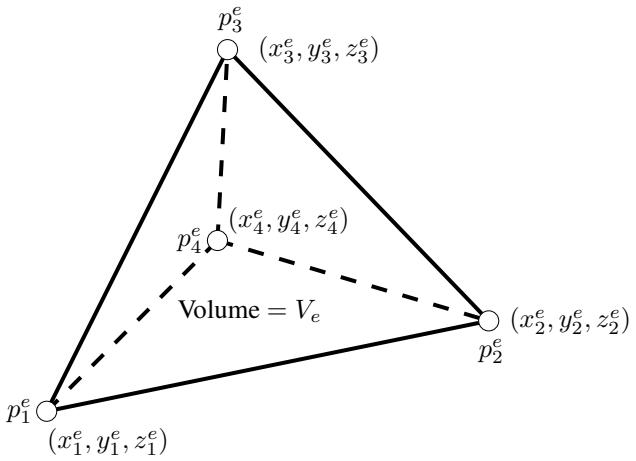


Figure 16: 3D element

The corresponding shape functions, N_1^e, N_2^e, N_3^e, and N_4^e, are given by

$$N_1^e = \tfrac{1}{4V_e}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x & x_2 & x_3 & x_4 \\ y & y_2 & y_3 & y_4 \\ z & z_2 & z_3 & z_4 \end{vmatrix}, \quad N_2^e = \tfrac{1}{4V_e}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x & x_3 & x_4 \\ y_1 & y & y_3 & y_4 \\ z_1 & z & z_3 & z_4 \end{vmatrix},$$

$$N_3^e = \tfrac{1}{4V_e}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x & x_4 \\ y_1 & y_2 & y & y_4 \\ z_1 & z_2 & z & z_4 \end{vmatrix}, \quad N_4^e = \tfrac{1}{4V_e}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ z_1 & z_2 & z_3 & z \end{vmatrix},$$

where

$$V_e = \frac{1}{4}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{vmatrix},$$

is the volume of element e. Note that for $j, k = 1, \ldots 4$, we have

$$N_j^e(x_k, y_k) = \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{otherwise.} \end{cases}.$$

We define

$$N^e = [N_1^e \; N_2^e \; N_3^e \; N_4^e] \qquad \text{and} \qquad B^e = \tilde{\nabla} N^e,$$

as in the 2D case. In order to calculate B it is convenient to write the shape functions in a different form,

namely

$$N_1^e = \frac{1}{4V_e}\left(\begin{vmatrix} x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} - x\begin{vmatrix} 1 & 1 & 1 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} + y\begin{vmatrix} 1 & 1 & 1 \\ x_2 & x_3 & x_4 \\ z_2 & z_3 & z_4 \end{vmatrix} - z\begin{vmatrix} 1 & 1 & 1 \\ x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \end{vmatrix}\right),$$

$$N_2^e = \frac{1}{4V_e}\left(-\begin{vmatrix} x_3 & x_4 & x_1 \\ y_3 & y_4 & y_1 \\ z_3 & z_4 & z_1 \end{vmatrix} + x\begin{vmatrix} 1 & 1 & 1 \\ y_3 & y_4 & y_1 \\ z_3 & z_4 & z_1 \end{vmatrix} - y\begin{vmatrix} 1 & 1 & 1 \\ x_3 & x_4 & x_1 \\ z_3 & z_4 & z_1 \end{vmatrix} + z\begin{vmatrix} 1 & 1 & 1 \\ x_3 & x_4 & x_1 \\ y_3 & y_4 & y_1 \end{vmatrix}\right),$$

$$N_3^e = \frac{1}{4V_e}\left(\begin{vmatrix} x_4 & x_1 & x_2 \\ y_4 & y_1 & y_2 \\ z_4 & z_1 & z_2 \end{vmatrix} - x\begin{vmatrix} 1 & 1 & 1 \\ y_4 & y_1 & y_2 \\ z_4 & z_1 & z_2 \end{vmatrix} + y\begin{vmatrix} 1 & 1 & 1 \\ x_4 & x_1 & x_2 \\ z_4 & z_1 & z_2 \end{vmatrix} - z\begin{vmatrix} 1 & 1 & 1 \\ x_4 & x_1 & x_2 \\ y_4 & y_1 & y_2 \end{vmatrix}\right),$$

$$N_4^e = \frac{1}{4V_e}\left(-\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} + x\begin{vmatrix} 1 & 1 & 1 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} - y\begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ z_1 & z_2 & z_3 \end{vmatrix} + z\begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}\right).$$

Let $\tau, \sigma \in \{x, y, z\}$ and $i, j, k \in [1, 2, 3, 4]$. We define

$$\tau\sigma(ijk) = \begin{vmatrix} 1 & 1 & 1 \\ \tau_i & \tau_j & \tau_k \\ \sigma_i & \sigma_j & \sigma_k \end{vmatrix}.$$

By using the above notation, a calculation of $(B^e)^T$ gives

$$(B^e)^T = \frac{1}{4V_e}\begin{bmatrix} -yz(234) & 0 & 0 & xz(234) & -xy(234) & 0 \\ 0 & xz(234) & 0 & -yz(234) & 0 & -xy(234) \\ 0 & 0 & -xy(234) & 0 & -yz(234) & xz(234) \\ yz(341) & 0 & 0 & -xz(341) & xy(341) & 0 \\ 0 & -xz(341) & 0 & yz(341) & 0 & xy(341) \\ 0 & 0 & xy(341) & 0 & yz(341) & -xz(341) \\ -yz(412) & 0 & 0 & xz(412) & -xy(412) & 0 \\ 0 & xz(412) & 0 & -yz(412) & 0 & -xy(412) \\ 0 & 0 & -xy(412) & 0 & -yz(412) & xz(412) \\ yz(123) & 0 & 0 & -xz(123) & xy(123) & 0 \\ 0 & -xz(123) & 0 & yz(123) & 0 & xy(123) \\ 0 & 0 & xy(123) & 0 & yz(123) & -xz(123) \end{bmatrix}.$$

Figure 17 shows how one can enumerate two elements, e_1, and e_2.

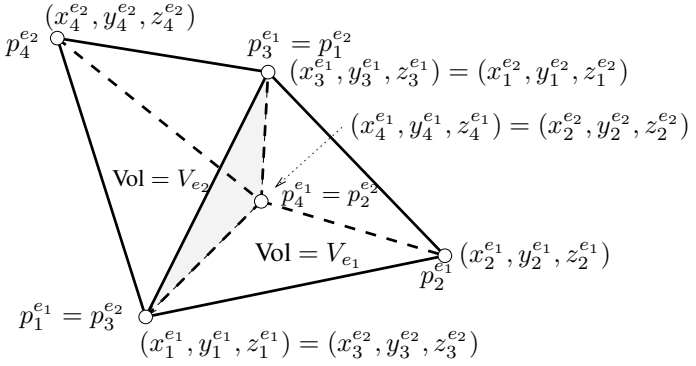


Figure 17: Two 3D elements

3.2 Comparison between triangular and rectangular meshes

Even though the triangular mesh is much more complicated than the rectangular one, it is often more efficient when the boundaries of the domain have a high curvature. Figure 18 shows how Delauny's algorithm makes the mesh much denser close to boundaries with a high curvature.

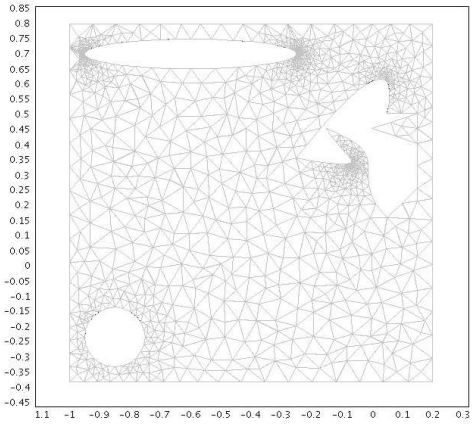


Figure 18: Triangular meshing on an irregular domain

3.3 FEM assembly of the equilibrium PDE

Let us look again at the first equation of this section, namely the equation for equilibrium,

$$\tilde{\nabla}\boldsymbol{\sigma} + \mathbf{b} = \mathbf{0},$$

which is defined on a domain $\mathcal{D} \subset \mathbb{R}^3$. We denote by the letter V the volume of $\mathcal{D}$, and by the letter S the surface area of the boundary, $\partial\mathcal{D}$.

Let $\mathbf{v} \in \mathbb{R}^3$ be an arbitrary vector. If we multiply the equation above by $\mathbf{v}^T$ and integrate over V we get the following weak form of the equation

$$\int_V (\tilde{\nabla}\mathbf{v})^T \boldsymbol{\sigma}\, dV = \int_S \mathbf{v}^T \mathbf{t}\, dS + \int_V \mathbf{v}^T \mathbf{b}\, dV$$

after some manipulation. After some more manipulation we get

$$\int_V \mathbf{B}^T \boldsymbol{\sigma}\, dV = \int_S \mathbf{N}^T \mathbf{t}\, dS + \int_V \mathbf{N}^T \mathbf{b}\, dV.$$

Since $\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}$ and $\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}$, we get

$$\left(\int_V \mathbf{B}^T\mathbf{D}\mathbf{B}\, dV\right)\mathbf{a} = \int_S \mathbf{N}^T \mathbf{t}\, dS + \int_V \mathbf{N}^T \mathbf{b}\, dV.$$

The matrix

$$\mathbf{K} = \int_V \mathbf{B}^T\mathbf{D}\mathbf{B}\, dV$$

is called the *stiffness matrix*, and the vector

$$\mathbf{f} = \int_S \mathbf{N}^T \mathbf{t}\, dS + \int_V \mathbf{N}^T \mathbf{b}\, dV$$

is called the $load\ vector$. The equation above can thus be written in what is called the standard FE formulation as

$$\mathbf{K}\mathbf{a} = \mathbf{f}. \tag{11}$$

Equation (11) describes a large system of equations.

For more details see for example [9] or [13]. A simple FE-solver was implemented during the work of this thesis and its pseudo-code is available in the appendix.

4 Optimization

In this section we will introduce some basic concepts regarding optimization which are essential to understand some of the theory in the following sections. Various literature on optimization is available and we suggest the interested reader to further explore the field by reading [1] and [8] .

4.1 Some basic concepts from general optimization

A dictionary would declare optimization as the process of making a system or design as effective or functional as possible. In mathematical optimization we study an objective function f and try to find its minimal value among a set of feasible solutions, especially by mathematical techniques. The corresponding optimal solution, x_{opt}, is a vector of design variables.

Any parameters which effect the characteristics of the system being optimized can be chosen as design variables. Thus design variables can be either discrete taking values from a specific set of magnitudes or continuous taking any magnitude in a given range. The set of design variables needs to be feasible in the sense that if an optimal solution is obtained it can also be implemented.

The objective function, f, which constitutes the actual system function that can be improved, is often subject to mathematical constraints which are restrictions of the feasible solutions. There are two types of constraints that we have to deal with: inequality constraints ($\leq$ or $\geq$) and equality constraints ($=$). If there are no inequality constraints then the problem can be transformed into an equivalent unconstrained one by including the equality constraints in the objective function formulation.

In classical optimization the problem can be treated by differential calculus and calculus of variations which requires that we formulate the problem as a partial differential equation subject to simplifications.

A general mathematical problem formulation reads:

$$\min f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \geq 0, i \in \mathcal{I}, h_j(\mathbf{x}) = 0, j \in \mathcal{J}, \tag{12}$$

where $\mathbf{x}$ is chosen from a set of design variables and $\mathcal{I}$ and $\mathcal{J}$ are finite sets.

When optimizing for a given problem we make the assumption that a solution to the given problem exists and that our formulation of the problem provides flexibility for the type and range of modifications to the initial design. In order to define an optimal value we need some further definitions.

4.1.1 Differentiability

Let $D \subseteq \mathbb{R}^n$ and $f : D \to \mathbb{R}$. f is said to be differentiable at $\mathbf{x} = (x_1, ..., x_n) \in D$ if the limit:

$$\lim_{h \to 0} \frac{f(x_1, ..., x_{i-1}, x_i + h, x_{i+1}, ..., x_n) - f(x_1, ..., x_n)}{h} \tag{13}$$

exists for every variable x_i at the point $\mathbf{x}$. A function is said to be differentiable on a set if it is differentiable at every point within the set.

4.1.2 Convexity

A set $S \subseteq \mathbb{R}^n$ is convex if for all $\mathbf{x_1}$ and $\mathbf{x_2}$ in S and all λ in the interval $0 < \lambda < 1$ the point $\lambda\mathbf{x_1} + (1 - \lambda)\mathbf{x_2}$ is also in S, or in mathematical terms:

$$\left.\begin{array}{c} \mathbf{x_1}, \mathbf{x_2} \in S \\ \lambda \in (0, 1) \end{array}\right\} \Rightarrow \lambda\mathbf{x_1} + (1 - \lambda)\mathbf{x_2} \in S. \tag{14}$$

If $S \subseteq \mathbb{R}^n$ is a convex set, then a function $f : S \to \mathbb{R}$ is said to be convex if for all $\mathbf{x_1}$ and $\mathbf{x_2}$ in S and all λ in the interval $0 < \lambda < 1$ we have:

$$\left.\begin{matrix}\mathbf{x_1}, \mathbf{x_2} \in S \\ \lambda \in (0,1)\end{matrix}\right\} \Rightarrow f(\lambda \mathbf{x_1} + (1-\lambda)\mathbf{x_2}) \leq \lambda f(\mathbf{x_1}) + (1-\lambda) f(\mathbf{x_2}). \tag{15}$$

4.1.3 Optimality

Let $S \subseteq \mathbb{R}^n$ and $f : S \to \mathbb{R}$.

Global minimum: A variable $\mathbf{x_1} \in S$ is said to be a global minimum of f over S if f attains the lowest value over S at $\mathbf{x_1}$. That is, $\mathbf{x_1} \in S$ is a global minimum of f over S if and only if

$$f(\mathbf{x_1}) \leq f(\mathbf{x}), \forall \mathbf{x} \in S. \tag{16}$$

Local minimum: A variable value $\mathbf{x_1}$ represents a local minimum of a function f over S if there exists a small enough neighborhood intersected with S around $\mathbf{x_1}$ such that $\mathbf{x_1}$ it is a global minimum in the intersection. Note that a global minimum in particular is a local minimum.

A local minimum of a convex function is also a global minimum if the set we are optimizing for is convex. A strictly convex function will have at most one global minimum over a convex set. For a convex function, a linear interpolation is never lower then the function itself.

In order to create an efficient, locally or globally convergent iterative algorithm for an optimization problem one needs to directly base it on necessary and/or sufficient local optimality conditions. Optimality conditions depend on the problem type

4.1.4 Penalty functions

In order to relax constraints in the problem formulation we use penalty functions. Consider the optimization problem stated in Equation (12). We assume that the set $S \subseteq \mathbb{R}^n$ is non-empty, closed and that the function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. The basic idea behind all penalty methods is to replace the constrained optimization problem with an equivalent unconstrained problem:

$$\text{minimize } f(\mathbf{x}) + X_s(\mathbf{x}) \tag{17}$$

where we have introduced the penalty function:

$$X_s(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in S, \\ +\infty & \text{otherwise} \end{cases} \tag{18}$$

The idea is that only when achieving feasibility can we concentrate on minimizing f. Since this penalty function is non-differentiable, discontinuous, and even not finite (though it is convex provided S is convex) it is computationally bad.

In the sections to come, where we describe how our programs in shape optimization works, we will refer to a penalty function. We define a constant, α, which is of orders of magnitude larger than the maximum stress in the linkage arm. We let:

$$\rho = \frac{\text{Volume of linkage arm}}{\text{Allowed maximum volume}}$$

If a suggested linkage arm from an iteration has greater volume than the initial design we let $X_s(x) = \alpha \cdot \rho$ in Equation (17). Since α is orders of magnitude larger then the maximum stress we decided to skip the stress calculations in order to decrease computing time. In addition we return α every time a program

used to construct a design, Femlab (see Section 6.1), is not able to construct or calculate for some given parameters thus returning a high objective value rather than having the algorithms crash. For a more detailed discussion on penalty functions we refer to [8].

Upper and lower bound constraints on design variables in NEWUOA are implemented by a logarithmic barrier method. We transform the objective function f in the inequality constraint problem in Equation (12) into a combined objective barrier function which is given by:

$$P(\mathbf{x};\mu) = f(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log g_i(\mathbf{x}) \tag{19}$$

In our calculations we start with $\mu = 1$ and then multiply it by a factor 0.8 in every following iteration and implement the barrier only when approaching the boundary.

4.1.5 Problem types

Below we list various problem types:

Linear programming (LP): The objective function f and all constraints g_i, $i \in \mathcal{I}$ and h_j, $j \in \mathcal{J}$ are linear.

Nonlinear programming (NLP): The objective function f or some constraints g_i, $i \in \mathcal{I}$ and h_j, $j \in \mathcal{J}$ are nonlinear.

Integer programming (IP): f is defined on a set X, where $X \subseteq \mathbb{Z}^n$.

Convex programming (CP): f is convex; g_i, $i \in \mathcal{I}$ are concave; h_j, $j \in \mathcal{J}$ are affine; X is closed and convex.

Non-convex programming (NCP): The complement of the above.

Continuous optimization: f, g_i, $i \in \mathcal{I}$ and h_j, $j \in \mathcal{J}$ are continuous.

Unconstrained optimization: $\mathcal{I} \cup \mathcal{J} = \emptyset$.

Constrained optimization: $\mathcal{I} \cup \mathcal{J} \neq \emptyset$.

Differentiable optimization: f, g_i, $i \in \mathcal{I}$ and h_j, $j \in \mathcal{J}$ are at least once continuously differentiable on an open set containing X.

Non-differentiable optimization: At least one of f, g_i, $i \in \mathcal{I}$ and h_j, $j \in \mathcal{J}$ is non-differentiable.

P-complete: A problem that can be solved efficiently using parallel computers is in P. A problem is P-complete if a parallel computer with a polynomial number of processors can reduce all other problems in P to this problem in polylogarithmic time, that is there exists a constant $c > 0$ such that the execution time is not longer than $O((\log n)^c)$.

NP-complete: A problem that can be verified in nondeterministic polynomial time, that is, if there exists a constant $c > 0$ such that the execution time is $O(n^c)$, then the problem is in NP. A problem is NP-complete if it is in NP and all other NP-problems can be translated into this problem. Every P-problem is also an NP-problem. No polynomial time algorithm exists for any NP-complete problem (unless P=NP), and they are generally considered infeasible to solve.

4.2 Structural optimization

Using optimization algorithms in order to construct good mechanical designs is an area of great interest. Its objective is to find a structure which is optimal in some sense. Structural optimization can be divided into three main categories:

$$\left\{ \begin{array}{c} \text{Sizing} \\ \text{Shape} \\ \text{Topology} \end{array} \right\} \text{ optimization}$$

Generally, we have two kinds of variables in a structural optimization problem:

$$\left\{ \begin{array}{c} \text{Design} \\ \text{State} \end{array} \right\} \text{ variables}$$

Below some examples are shown

Example (sheet,objects)

Sizing optimization:	Determine the thickness (> 0) of sheet.
Shape optimization:	Determine the boundary of object.
Topology optimization:	Determine whether the design should contain holes.
Design variables:	Thickness of sheet, position of boundary etc.
State variables:	Displacements, stresses.

For more about the general theory of structural optimization, see for example [7]. Here we only give a brief idea of the subject, but more detailed discussion will be given later in this section, when topology optimization an d shape optimization will have separate discussion.

General formulation

The general form of a structural optimization problem for a linearly elastic material can be of the form (See e.g. [11])

$$\left\{ \begin{array}{l} \quad \min_{s,d} F(s,d), \\ \text{subject to} \\ \quad \Pi(s,d) \leq \Pi(\overline{s},d), \quad \forall \overline{s} \in U, \qquad (*) \\ \text{and} \\ \quad (s,d) \in S \times D. \end{array} \right. \tag{20}$$

$(*)$: requirement on the equilibrium according to the principle of minimum potential energy. Here

1. F is the *objective function*,
2. Π is the *total potential energy*,
3. d denotes *designs variables*,
4. s denotes *state variables*,
5. U is the set of *kinematically admissible displacements*,
6. D is the set of *admissible designs*, and
7. S is the set of *admissible states*.

In Section 4.3 we will discuss problem (20) and the approach of finding a solution to it using shape optimization. In Section 4.3 we will discuss another approach for solving (20), namely topology optimization. The definitions of D and S are different with these two methods.

4.3 Shape optimization (SO)

Within the field of mechanical structures, shape optimization usually evolves around finding the shape of a structure, which best resists stress. In general, there exists an infinite number of possible solutions to the problem and thus we need to introduce some constraints on our design. Typically, these constraints have

certain cost aspects, such as using as little material as possible, or are related to some boundary constraints, obstructions, in the application of implementation.

Several approaches can be used to represent the shape of an object in a computer application. Here, we use a geometric approach, implemented in Femlab (see Section 6.1), where the shape of the linkage arm has a predefined boundary representation, which is determined by a set of design parameters. After choosing the design variables and completing the modeling stage, we need to apply boundary conditions to the design to be able to calculate the maximum stress; this is also done in Femlab. For more details on how this is done, see Section 5.3.

When choosing from the infinite number of possible design variables for our linkage arm some things we require:

- The current design of the linkage arm needs to be among the set of feasible solutions using our selection of design variables.
- We want to limit the number of design variables in order to reduce computing time.
- The size of the linkage arm might be bounded by the spread of other details in the vehicle design.
- We want to ensure differentiability of the boundary.
- The linkage arm must be feasible to produce.
- We need to formulate our problem in a way that enables us to modify the design according to the results from the optimization algorithm.

When the set of design variables has been determined, the shape optimization process uses iterative methods, which in general evolve around modifying one or more of the design variables. By modifying the design variables, we also change the shape of the structure which results in a new estimate of the objective function. The new objective value, in our case the maximum stress, can then be used to determine whether progress is being made. The optimization process is repeated until some termination criteria is fulfilled, e.g. a maximum number of function calculations.

In a typical linear shape optimization problem we calculate the gradient of the objective function, ∇f, and use gradient methods to converge to an optimal solution. However, since the maximum stress in the linkage arm is a highly non-linear and non-differentiable function, we need to rely on algorithms that do not pay attention to the gradient. Here, we have used two algorithms, NEWUOA and MultiOb, to minimize the maximum stress objective function, the algorithms are discussed in Section 6.2 and Section 6.3 respectively. Since NEWUOA is intended for use in unconstrained optimization and MultiOb only has boundary constraints on the design variables, all other constraints, such as volume constraints and boundary constraints in NEWUOA need to be implemented inside the objective function by the means of a penalty function which were discussed in Section 4.1.4. As input, the algorithms take the value of the design variables and the objective value after each iteration. As output, they return suggested modifications to the design variables. The suggested modifications are then passed on to the boundary representation modeling program, here written in Femlab, which updates the geometry and returns a new estimate of the maximum stress. The modeling process, boundary conditions for the linkage arm, and the optimization algorithms will all be described further in the following sections. The interested reader can find more detail on shape optimization in [14].

Beginning with a 2D boundary representation of the linkage arm, we gradually increase the detail of the model as we move on to 3D.

4.4 Topology optimization (TO)

In topology optimization, a domain, $\mathcal{D}$, is defined (often a rectangle in 2D or a box in 3D) which we want to fill with material, which generally has less volume than the original domain. The domain is meshed, and

boundary conditions are defined. The objective is to determine which elements in the mesh should contain material and which should not contain material, in order for some predefined objective function to be either minimized or maximized. Solving a TO problem is an integer programming (IP) problem which in general is NP complete. This means that a true optimal solution can not be found in any reasonable amount of time (see Section 4.1 for definitions). It is evident that some approximations and relaxations must be made, and we will discuss some of them in this section. Our main sources for this section is the excellent book by Bendsøe and Sigmund [2], which is a great starting point to learn about this field, with a very detailed reference list.

4.4.1 Some concepts in topology optimization

In all our topology optimization applications in this thesis, we use a regular rectangular mesh. When we talk about *mesh size* we mean the length of an edge of an element. One would expect a more accurate optimal solution on a finer mesh but this is not the case in general. What usually happens is that when we move to a finer mesh, we obtain more holes in the optimal structure. As we let the mesh size go to zero, the solution approaches a design with a non-isotropic material. This mesh dependency is actually the result of a general rule of the TO problem, which is the lack of existence of an optimal solution.

As stated before, TO problems are IP problems in general and thus are often difficult to solve. Instead of finding an integer solution, one relaxes the problem and allows intermediate values. However, as it is infeasible to have intermediate values in the final structure, we must introduce some kind of an interpolation (penalty) function, which makes intermediate values more expensive. Equation (21) shows two types of penalty functions.

$$\begin{aligned} f(\rho) &= \rho^p, \\ f(\rho) &= \frac{\rho^p}{1+q(1-p)}. \end{aligned} \tag{21}$$

Here, $\rho \in [0,1]$, denotes the density of an element. The penalty parameters, $p \geq 1$, and $q \geq 0$, have the property that the larger they are, the more expensive it is to use intermediate values. Note that when $\rho = 0$ we have $f(\rho) = 0$, and when $\rho = 1$ then $f(\rho) = 1$. However, if $\rho \notin \{0,1\}$, we have $f(\rho) < \rho$, except if $p = 1$ and $q = 0$.

The former interpolation function is used in [15], an article which we discuss in more detail in Section 6.4. The latter is a special case of a so-called SIMP interpolation function, and is used in TO++ (see Section 6.5).

The stiffness matrix, defined in Section 3.3, is defined on the whole domain, $\mathcal{D}$ but is now written as

$$\mathbf{K} = \int_{\mathcal{D}} f\mathbf{B}^T\mathbf{D}\mathbf{B}d\mathcal{D}.$$

It is common to introduce the concept of *filter radius* in TO algorithms in order to prevent the so-called *checkerboard effect*. By checkerboard effect, we mean that there are regions in the calculated optimal solution where adjacent element have completely different material status. The filter radius is defined in the following way:

$$\tilde{\rho}(\mathbf{x}) = S(\rho, \mathbf{x}) = \frac{1}{C}\int_{\mathbb{R}^d} \rho(\mathbf{y}) \max\left(0, 1 - \frac{|\mathbf{x}-\mathbf{y}|}{R}\right) d\mathbf{y},$$

where C is a normalization constant. If $d = 2$, we obtain

$$C = \int_{S(0,R)} rdA = \int_0^R \int_0^{2\pi} \left(1 - \frac{r}{R}\right) rd\theta dr = 2\pi\left(\frac{1}{2}R^2 - \frac{R^2}{3}\right) = \frac{\pi R^2}{3},$$

and if $d = 3$, we obtain

$$C = \int_{S(0,R)} r dV = \int_0^R \int_0^{2\pi} \int_0^{\pi} \left(1 - \frac{r}{R}\right) r^2 d\phi d\theta dr = 4\pi \left(\frac{1}{3}R^3 - \frac{R^3}{4}\right) = \frac{\pi R^3}{3}.$$

In optimization algorithms, the original density is replaced by the density we get by using the filtered value, which depends on the filter radius. Generally, the filter radius should be a few multiples of the mesh size.

Is is natural to constrain the allowed volume of the linkage arm ($V \leqslant V_0$) in a given topology optimization problem. The topological result is sensitive to changes in the volume restriction. Furthermore, we often want to predefine regions that must have a certain material status in the final solution.

5 Modeling

5.1 Introduction

In the previous sections, we described some of the theory needed for solving the linkage arm problem. In this section, we describe the way in which we model the problem for shape optimization and topology optimization.

In the shape optimization part, we wanted to ensure that we would obtain a solution with a similar geometry to the original linkage arm, shown in Figure 19. In the topology optimization part, we allowed an arbitrary shape of the linkage arm, though we required it to be contained in a box with certain edge lengths.

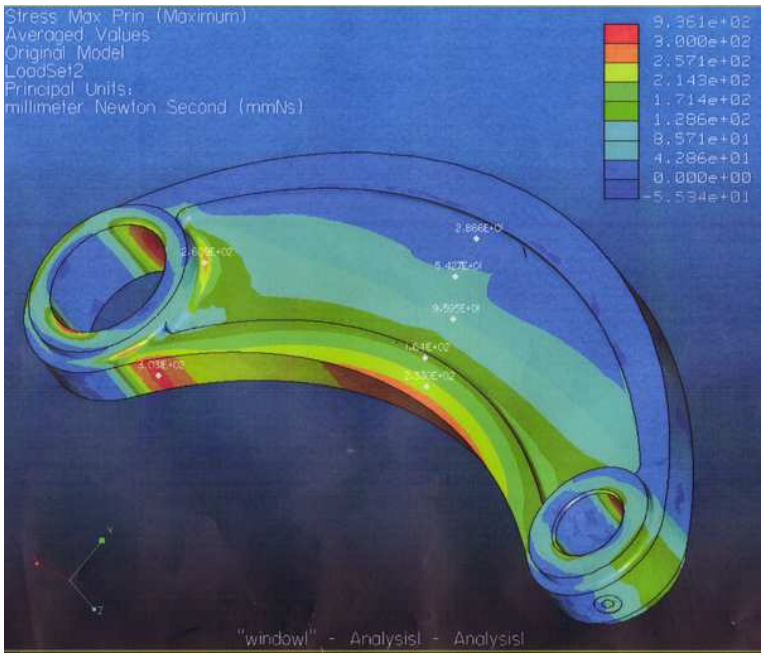


Figure 19: The original linkage arm

In both types of models, we assume that we have two holes of certain sizes with a fixed distance between them. Around each hole we require some prescribed material in the shape of a circle. This is shown in Figure 20.

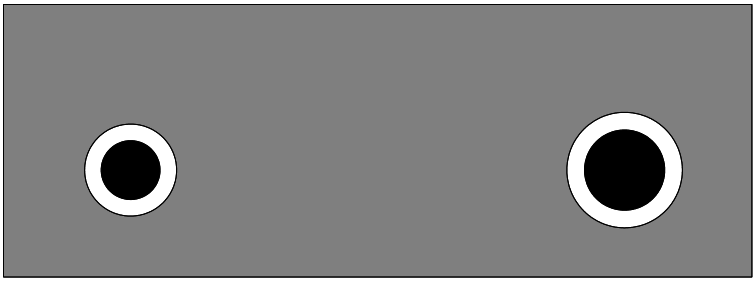


Figure 20: Two holes (black color) with prescribed material (white color) surrounding them

In Table 1 the dimensions of the holes are given, along with some physical constants necessary for our calculations.

Description of parameter	*Value*
Radius of prescribed left circle	0.043 m
Radius of left hole	0.0275 m
Radius of prescribed right circle	0.054 m
Radius of left hole	0.0375 m
Distance between holes (center points)	0.462 m
Thickness of prescribed material	0.0588 m
Young's modulus	210000 Pa
Poisson's ration	0.3
Density of material (steel)	$7.8 \cdot 10^{-9}$

Table 1: Some parameters used for the structural optimization models

5.2 Shape optimization modeling

5.2.1 Modeling in 2D

The original linkage arm has a smooth outer boundary that is represented with circles with varying center points and radii. When choosing the design variables for the optimization process we use this initial boundary representation and select our design variables related to this set of center points and radii.

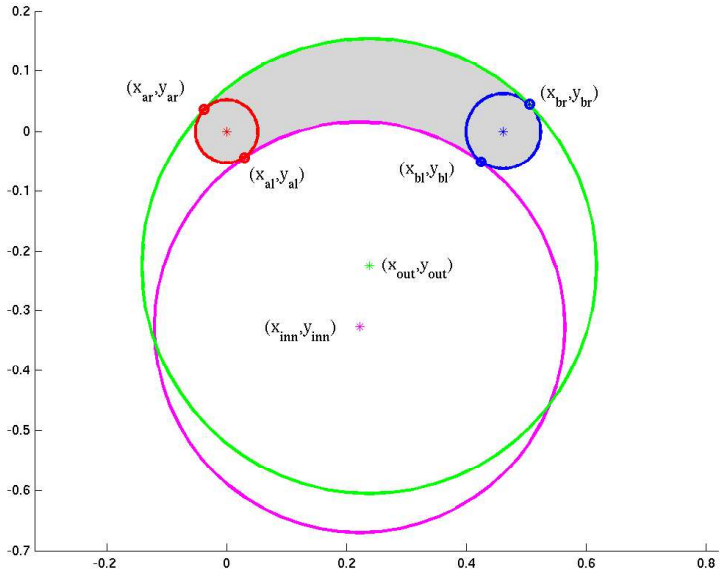


Figure 21: An illustration of possible design parameters for the linkage arm in 2D.

Since the boundary of the original linkage arm is smooth we need to secure the differentiability of the boundary when connecting the circular edges. Allowing all variables to vary freely can result in a non-connectible boundary and even when the resulting boundary is continuous, differentiability is not automatically guaranteed at all points. Thus in order to meet this differentiability requirement only the center points and radii of the circles that represent the outer part (r_{out},$(x_{\text{out}}, y_{\text{out}})$) and inner part ($r_{\text{inn}}$,$(x_{\text{inn}}, y_{\text{inn}})$) of the boundary are used as design variables, see Figure 21. Since the boundary is represented by circles differentiability is guaranteed at all points on the boundary except where circles meet. To secure differentiability at the intersection points it is possible to modify the variables that represent the left (r_{left},$(x_{\text{left}}, y_{\text{left}})$) and right ($r_{\text{right}}$,$(x_{\text{right}}, y_{\text{right}})$) end circles, see Figure 23.

Now which conditions does an end circle need to fulfill to secure differentiability at the connection points?

- It needs to have an outward normal at the connection point which is parallel to the outward normal of the outer circle and at the same time parallel to the outward normal of the inner circle.
- Its center point $(x_{\text{right/left}}, y_{\text{right/left}})$ must be spaced at an equal distance ($r^i_{\text{right/left}} = r^o_{\text{right/left}}$) from the two connection points in a direction parallel to the normals and this distance would then serve as the radius $r_{\text{right/left}}$ of the end circle.

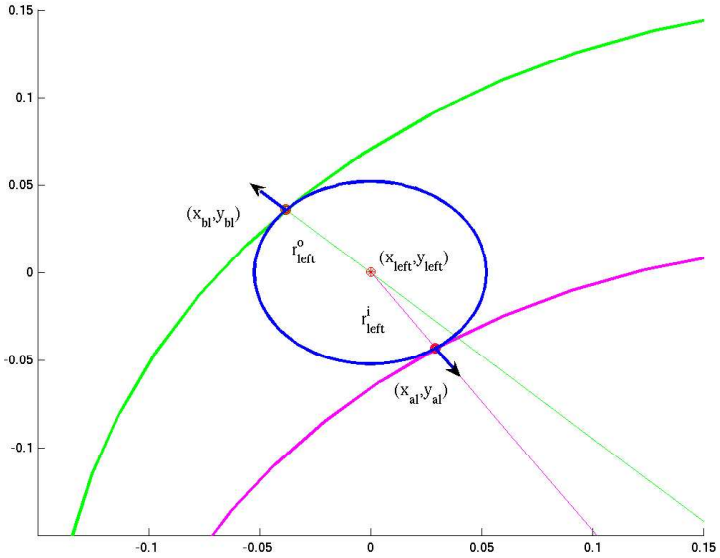


Figure 22: An illustration of differentiability requirements for the linkage arm

These conditions are illustrated in Figure 22 and an end circle that fulfills these conditions is guaranteed to have C^1 continuity at the connection points. The next step in the modeling process is to find this radii $r_{\text{right/left}}$ and center points $(x_{\text{right/left}}, y_{\text{right/left}})$ of the right and left end circles. To solve this problem one can use the center points of the holes in the original linkage arm as an initial solution. Lines are then drawn from the center points of the outer and inner circles, through these initial points, intersecting the design circles at the initial coordinates of the connection points on the edges (see Figure 23).

When the coordinates of the initial intersection points have been found the distance from the center point $(x_{\text{right/left}}, y_{\text{right/left}})$ to the respective connection point is easily calculated. In general these distances are not equal, that is $r^i_{\text{right/left}} \neq r^o_{\text{right/left}}$. Lets look at the following possibilities:

- The center point $(x_{\text{right/left}}, y_{\text{right/left}})$ is moved along the line connecting connection point $(x_{\text{b}}, y_{\text{b}})$ and $(x_{\text{out}}, y_{\text{out}})$ or
- the center point can be moved along the line connecting connection point $(x_{\text{a}}, y_{\text{a}})$ and $(x_{\text{inn}}, y_{\text{inn}})$.

Some definitions are in order at this stage. Now $r_{\text{right/left}}$ is defined as the radius of the resulting end circle and $(x_{\text{right/left}}, y_{\text{right/left}})$ as the center point, further definition include (see Figure 21):

- $(x_{\text{al}}, y_{\text{al}})$ as the connection point between left circle and lower edge
- $(x_{\text{bl}}, y_{\text{bl}})$ as the connection point between left circle and upper edge
- $(x_{\text{ar}}, y_{\text{ar}})$ as the connection point between right circle and lower edge
- $(x_{\text{br}}, y_{\text{br}})$ as the connection point between right circle and upper edge

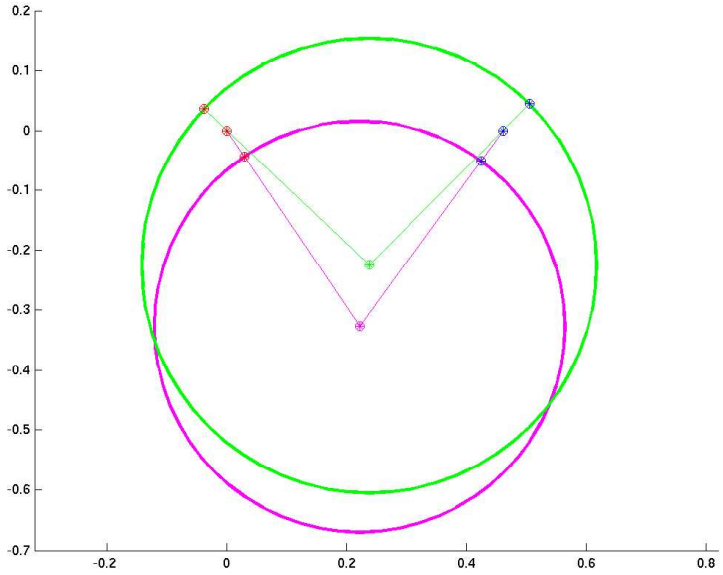


Figure 23: Illustration of the procedure for ensuring differentiability

If $(x_{\text{right/left}}, y_{\text{right/left}})$ moves along the line connecting $(x_{\text{out}}, y_{\text{out}})$ and $(x_{\text{b}}, y_{\text{b}})$ there are two choices. Either $(x_{\text{right/left}}, y_{\text{right/left}})$ moves towards $(x_{\text{b}}, y_{\text{b}})$ thus decreasing $r^o_{\text{right/left}}$ or it moves towards $(x_{\text{out}}, y_{\text{out}})$ thus increasing $r^o_{\text{right/left}}$. Only one direction will give a solution in each case. If the initial distance $r^o_{\text{right/left}}$ from $(x_{\text{right/left}}, y_{\text{right/left}})$ to $(x_{\text{b}}, y_{\text{b}})$ is longer then the distance $r^i_{\text{right/left}}$ from $(x_{\text{right/left}}, y_{\text{right/left}})$ to $(x_{\text{a}}, y_{\text{a}})$ then increasing the length of $r^o_{\text{right/left}}$ will not give a solution since it causes a decrease in the length of $r^i_{\text{right/left}}$ and thus a circle with center point $(x_{\text{right/left}}, y_{\text{right/left}})$ and radius $r_{\text{right/left}}$ that connects the upper and lower circles will not exist.

Thus only two possible approaches will give a solution when determining $(x_{\text{right/left}}, y_{\text{right/left}})$ and $r_{\text{right/left}}$.

- Moving $(x_{\text{right/left}}, y_{\text{right/left}})$ along the line towards the connection point $(x_{\text{a/b}}, y_{\text{a/b}})$ which is further away (decrease max($r^o_{\text{right/left}}$,$r^i_{\text{right/left}}$)).

- Moving $(x_{\text{right/left}}, y_{\text{right/left}})$ along the line away from the connection point $(x_{\text{a/b}}, y_{\text{a/b}})$ which is closer to $(x_{\text{right/left}}, y_{\text{right/left}})$ (increase min($r^o_{\text{right/left}}$,$r^i_{\text{right/left}}$)).

In the 2D model of the linkage arm the latter approach is used, thus always increasing the shorter of the distances resulting in a larger end circle.

The following relations hold:

$$\begin{aligned} (r_{\text{out}} - r_{\text{right/left}})^2 &= (x_{\text{right/left}} - x_{\text{out}})^2 + (y_{\text{right/left}} - y_{\text{out}})^2, \\ x_{\text{right/left}} &= x_{\text{a}} + (x_{\text{a}} - x_{\text{inn}})\frac{r_{\text{right/left}}}{r_{\text{inn}}}, \\ y_{\text{right/left}} &= y_{\text{a}} + (y_{\text{a}} - y_{\text{inn}})\frac{r_{\text{right/left}}}{r_{\text{inn}}} \end{aligned} \tag{22}$$

and

$$
\begin{aligned}
(r_{\text{inn}} + r_{\text{right/left}})^2 &= (x_{\text{right/left}} - x_{\text{inn}})^2 + (y_{\text{right/left}} - y_{\text{inn}})^2, \\
x_{\text{right/left}} &= x_{\text{b}} - (x_{\text{b}} - x_{\text{out}})\frac{r_{\text{right/left}}}{r_{\text{out}}}, \\
y_{\text{right/left}} &= y_{\text{b}} - (y_{\text{b}} - y_{\text{out}})\frac{r_{\text{right/left}}}{r_{\text{out}}}.
\end{aligned} \tag{23}
$$

Thus one of two formulas needs to be solved depending on which distance is shorter: If the distance $(x_{\text{right/left}}, y_{\text{right/left}})$ and $(x_{\text{b}}, y_{\text{b}})$ is shorter $(x_{\text{b}}, y_{\text{b}})$ is kept constant and $r_{\text{right/left}}$ is solved by:

$$
r_{\text{right/left}} = \frac{r_{\text{inn}}^2 - (y_{\text{b}} - y_{\text{inn}})^2 - (x_{\text{b}} - x_{\text{inn}})^2}{\frac{2(x_{\text{inn}} - x_{\text{b}})(x_{\text{b}} - x_{\text{out}})}{r_{\text{out}}} + \frac{2(y_{\text{inn}} - y_{\text{b}})(y_{\text{b}} - y_{\text{out}})}{r_{\text{out}}} - 2r_{\text{inn}}} \tag{24}
$$

which follows from Equation (23). However if the distance from $(x_{\text{right/left}}, y_{\text{right/left}})$ to $(x_{\text{a}}, y_{\text{a}})$ is shorter, $(x_{\text{a}}, y_{\text{a}})$ is kept constant and instead $r_{\text{right/left}}$ is solved by:

$$
r_{\text{right/left}} = \frac{r_{\text{out}}^2 - (y_{\text{a}} - y_{\text{out}})^2 - (x_{\text{a}} - x_{\text{out}})^2}{2r_{\text{out}} + \frac{2(x_{\text{a}} - x_{\text{inn}})(x_{\text{a}} - x_{\text{out}})}{r_{\text{inn}}} + \frac{2(y_{\text{a}} - y_{\text{inn}})(y_{\text{a}} - y_{\text{out}})}{r_{\text{inn}}}} \tag{25}
$$

which follows from Equation (22).

Now that the outer boundary of the 2D linkage arm has been solved for it only remains to fill it with material and insert two holes in the structure to finish the modeling approach. The resulting 2D model can be seen in Figure 24.

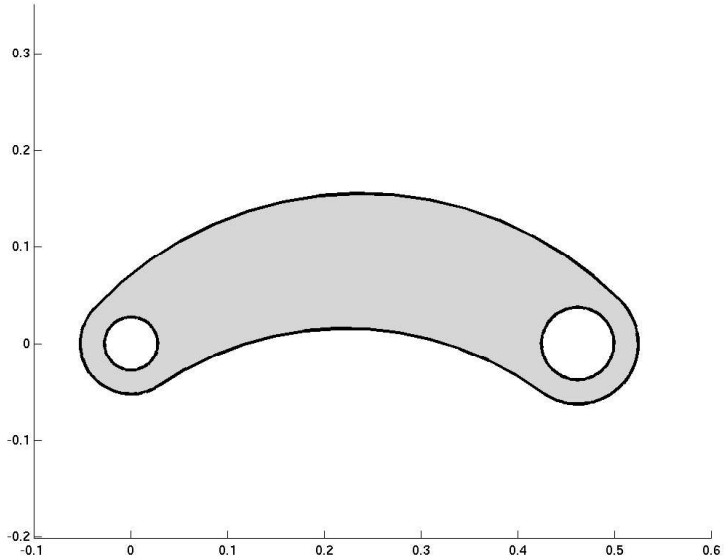


Figure 24: The linkage arm defined by Equations (22)–(25) in 2D

5.2.2 Modeling in 3D

Now that the 2D model version of the linkage arm has been defined it is time to step into the more complicated world of 3D linkage arms. As was stated earlier the selection of design variables should be able to represent the current design of the linkage arm and therefore the parameter values from the linkage arm

model data sheet will be used as an initial solution; for more details see [19]. In the simplest case the 3D linkage arm has the same outer boundary as the 2D model in the xy-plane and is constructed in the same manner. However, a more detailed 3D model varies in two aspects from a simple 3D version, where we have given a constant height to the 2D model.

- There is extra thick material around the holes.
- An inner section has been removed.

The thickening around the holes is fairly simple to construct as it constitutes adding two thick circles outside the respective holes in the linkage arm. The inner section is much more complex to construct and is worth further discussion.

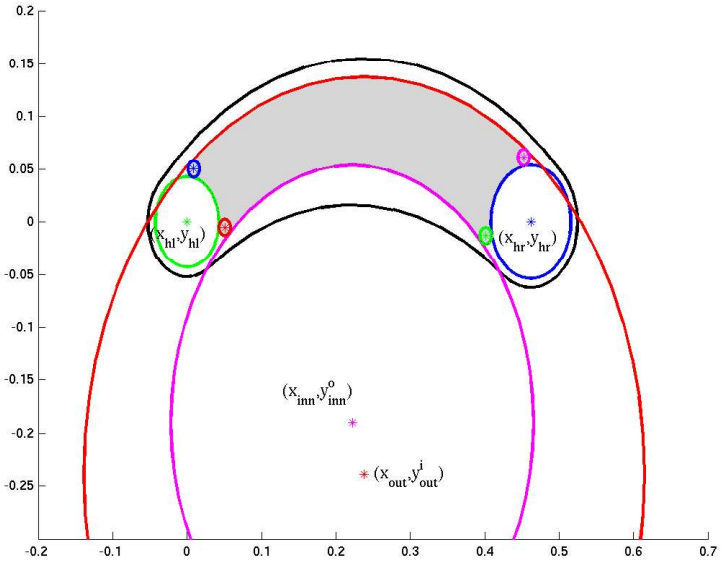


Figure 25: Additional possible design parameters in 3D. xy-plan view

The inner section that is removed can be represented by 8 circles with different center points and radii. We face the problem of differentiable boundaries again and therefore we use the smaller connecting circles to secure differentiability. This leaves 3 new possible design variables for each of the 4 circles that can be varied. However, the center points of the right $(x_{\text{hr}}, y_{\text{hr}})$ and left $(x_{\text{hl}}, y_{\text{hl}})$ end circles representing the inner boundary are the same as the center points of the holes. We also decided to link the x-coordinates of the upper, x^i_{out}, and lower, x^i_{inn}, inner boundary with the respective x-coordinates for the upper, x_{out}, and lower, x_{inn}, outer boundary. This results in a total of 6 new design variables for the 3D version of the linkage arm: y^i_{out} and r^i_{out} for the circle representing the outer part of the inner boundary, y^o_{inn} and r^o_{inn} representing the lower part of the inner boundary ,r^{out}_{hr} the radius of the right circle and r^{out}_{hl} as the radius of the left circle, see Figure 25.

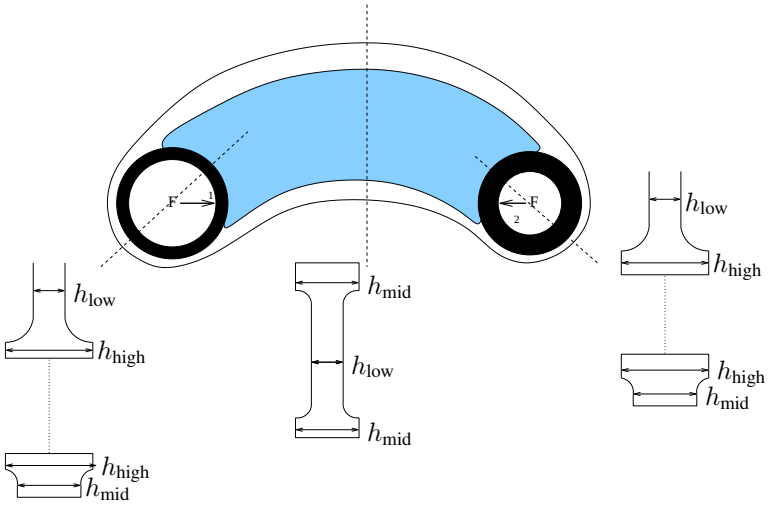


Figure 26: Cross sections showing the heights in different parts of the linkage arm

In Figure 26 the different heights of the linkage arm are demonstrated. Note that they are not used as design variables. Now that we have seen the various possible design variables for the 3D linkage arm it is time to define the various modeling approaches that have been tested.

Modeling approach 1 in 3D

As was stated earlier the first approach uses a simple 3D model where the 2D version of the linkage arm has been extended into 3D by giving it a constant height. In Figure 27 we can see this simplified version of the linkage arm.

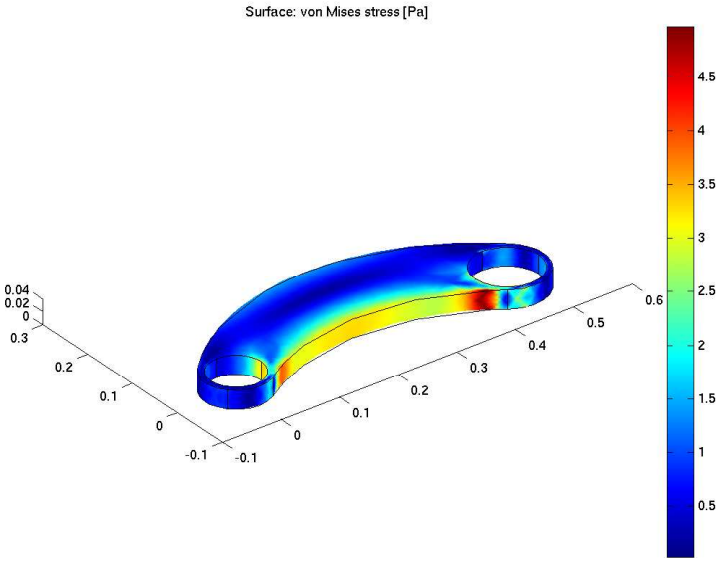


Figure 27: A linkage arm with constant thickness in 3D

Running some tests on this simple 3D version, mainly for programming purposes, gave results that were similar to the results for the 2D version, that is the resulting best linkage arm would have greater volume then the initial one. Since there would not be a lot of information to gain from this model we soon decided that it was time to increase the complexity of our model.

Modeling approach 2 and 3 in 3D

There are two model details in the linkage arm that we can use for our next model: the inner section and the thickness around the holes. By hindsight it might have been more logical to implement one detail at a time and study the results. However, here both the thicker circles around the holes and the inner section removal defined earlier are implemented. There is still some freedom in selecting among the possible design variables for the model, at least for the inner section. As was mentioned earlier we require that the inner boundary is differentiable and thus we can not vary all the circles representing it for the same reasons as were discussed in Section 5.2.1. When we have decided on design parameters for representing the inner boundary it is still not trivial to connect the circles into a differentiable boundary. Two possible approaches are:

- **Modeling approach 2 in 3D**: The first approach is similar to the one used for the outer boundary of the 2D linkage arm. Starting with an initial center point for each of the 4 connecting circles and their respective radius are then varied in order to fit the circles into a differentiable boundary representation using the same methods as were stated in more detail earlier in Section 5.2.1. Using this approach results in a different radius for each of the 4 connecting circles but the center points of the circles are kept at a close to constant position.
- **Modeling approach 3 in 3D**: In the second approach we keep a constant radius of $r = 8$mm on each of the 4 connecting circles. We find the center point of each circle by varying the radius of the respective inner boundary design circle by this fixed constant radius. The center point will be at the intersection of the 2 varied design circles and differentiability is secured at the connection points. Thus in this approach the center points of the connecting circles move freely but the radius is fixed. If we are not able to connect the design circles with a circle of fixed radius we return a penalty on the objective function; see Section 4.1.4.

Both approaches result in a linkage arm that resembles the original linkage arm for the given initial design variables and for comparison one can look at Figure 28. However, as will be shown in Section 7, the maximum stress is higher for the initial solution in the second approach which results in a better improvement of the objective value for that approach. Both approaches were implemented and tested and results are presented in Section 7.

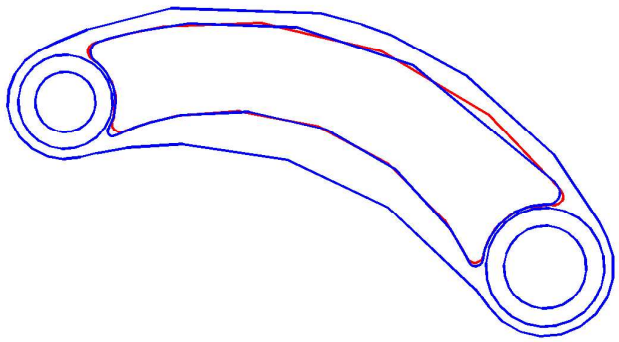


Figure 28: The linkage arm resulting from modeling approach 2 (blue) and 3 (red) in 3D with the initial values on the design variables

Modeling approach 4 in 3D

When boundary conditions were applied to the linkage arm resulting from modeling approach 2 the results would show a very high stress concentration on the topmost edge of the inner holes (see Figure 29). Motivated by an article on reducing stress concentration by Pedersen and Laursen (see [10]) we decided to increase the complexity of the 3D model further by smoothing the edges of the inner boundary by transforming them into the shape of an ellipse, hoping to relieve these high stress concentrations. This smoothing of the inner boundary was implemented in Femlab by revolving an elliptic curve along the circular path of the inner boundary. The parameters used were the ones suggested in the above article and no tests were made as to determining the optimal elliptic slope.

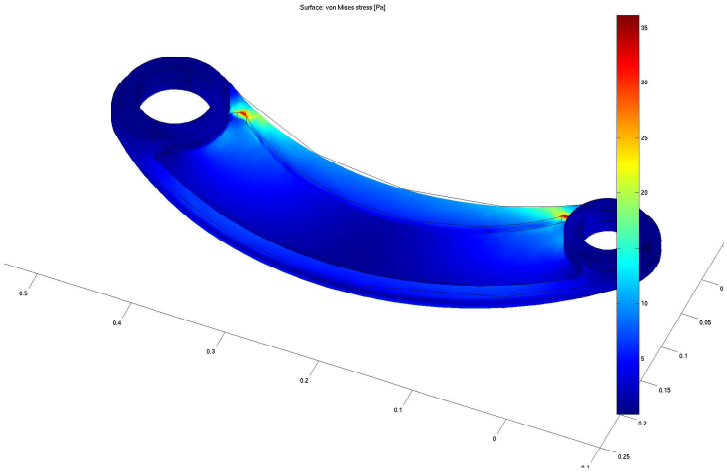


Figure 29: The resulting stress concentration in the linkage arm created by modeling approach 2. Note high stress on edges

The new linkage arm can be seen in Figure 30. This smoothing of the inner boundary increases the resemblance with the linkage arm in the test papers from Atlas Copco (see [16]) even if the data sheet does not clearly imply any smoothing of edges.

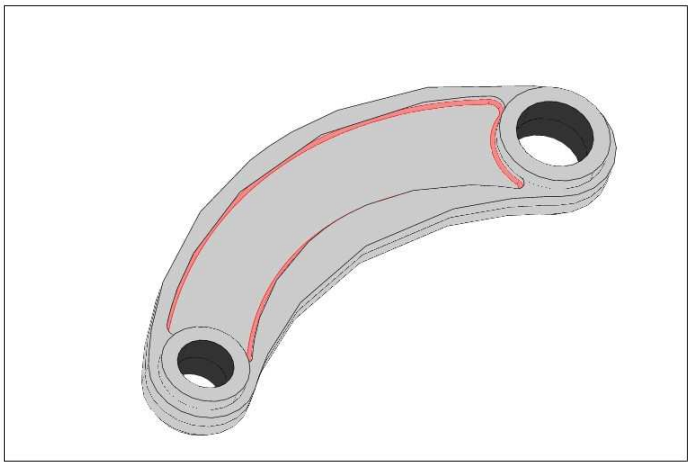


Figure 30: Linkage arm in 3D with an elliptic inner boundary (red)

It is hard to conclude whether this model relieved the stress concentration on the edges since this detailed model lead to problems with meshing in Femlab. Normal mesh parameter settings resulted in mesh with high stress concentrations in various places inside the elliptic boundary. In order to prevent this one needs to increase the resolution of the mesh elements to improve their quality which results in an increased number of mesh elements. The additional mesh elements needed in order to be able to evaluate the stresses in this modeling approach increase the amount of memory allocation required for the FEM-solvers in Femlab and here this results in an out of memory error when LU-factorizing the solution matrix. We tried using various FEM-solvers but we had no luck converging to a solution. We also tried meshing with different parameters but we were not able to get a satisfying result with the different meshing approaches. Thus we are not able to solve for this type of linkage arm in Femlab since we were unsuccessful in making the mesh accurate enough to get a decent solution. We therefore decided to wait with further tests on this model until equipped with better computational power and we are forced to remove some details from our model.

Modeling approach 5 and 6 in 3D

The problem with a high stress concentration on the inner boundary edges remains. Now that it is clear that smoother edges can not help to solve the problem another approach is needed. Instead of making the inner edges curved one can make the inner section which is removed smaller by increasing the radii of the right and left circles defining the boundary. This approach manages to relieve the edges of the design from high stress concentration and at the same time it resolves a model implementation problem in Femlab which has to do with limited ability in constructing two curved objects that are too close in space. This simplification can be implemented in both modeling approaches 2 and 3 which results in two new modeling approaches, 5 and 6, respectively. Other details are the same as in modeling approaches 2 and 3 discussed earlier. Results for each approach will be presented in Section 7. The difference is that instead of using r_{hr}^{out} and r_{hl}^{out} as control parameters for the edges of the inner boundary we add a constant 3mm to each radii.

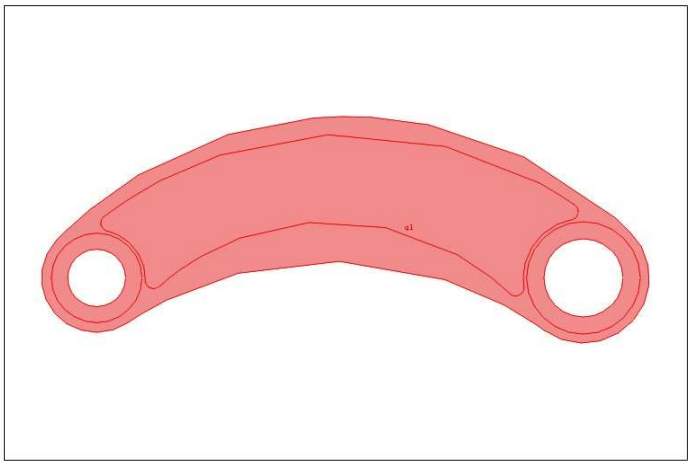


Figure 31: Linkage arm in 3D with smaller inner section removed

5.3 Boundary Conditions

5.3.1 Boundary Conditions in 2D

When the linkage arm model has been constructed the next step is to apply boundary forces to the 2D structure in order to calculate the desired maximum stress objective function. When defining the boundary

conditions for the linkage arm we have emphasized two things. First we want the boundary conditions to give symmetric stress concentrations for symmetric objects. Second we want to be able to imitate the real loading of the linkage arm as well as possible and as a motivation we look at Figure 1 for comparison. We start with a simple approach and gradually increase the complexity in order to fulfill the goals we have set forth.

Boundary Conditions 1 in 2D

This first approach is based on a typical steel plate example which will be discussed in Section 6.1 where parts of the structure are fixed in place while other parts are subject to applied loading. In Figure 32 we illustrate these first boundary conditions where a longitudinal inward force is applied along the inner right segment of the left hole while we hold the inner left edge segment of the right hole fixed in place.

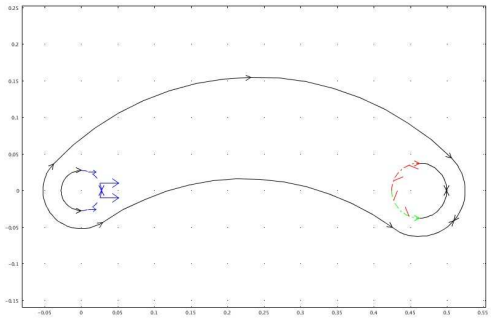


Figure 32: Pushing the left hole while holding the right hole fixed in place.

As we are still working in 2D we can not compare the resulting stress concentration to the 3D case except in a limited way. However we can study the symmetry of this approach for symmetric objects. As can be seen by looking at Figure 33 this approach does not give a symmetric stress solution for a symmetric object. An improvement in the boundary conditions is needed and we keep on working in 2D as that gives a more intuitive understanding then more complex models in 3D would.

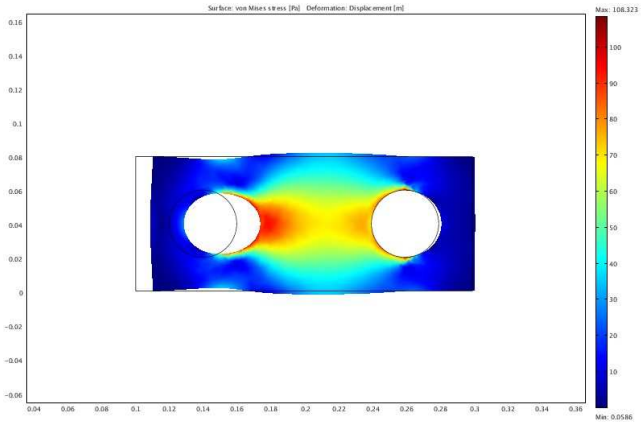


Figure 33: Resulting stress distribution when applying Boundary Condition 1 to a symmetric object.

Boundary Conditions 2 in 2D

In order to improve the symmetry in the solution we study the stress concentration for the previous approach. It can be seen in Figure 33 that the stress concentration is higher around the hole where force is applied than in the hole that is kept fixed, excluding the two points there were the boundary conditions are relaxed, where we have very high stress peaks. In order to remove these two stress peaks and increase the symmetry we decided that we would now apply equal longitudinal inward forces on the inside of both holes instead of keeping one hole fixed in place. This approach is illustrated in Figure 34.

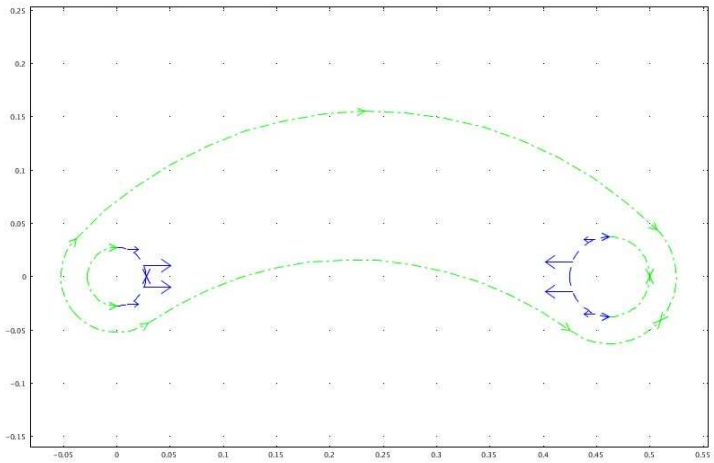


Figure 34: Pushing left hole and right hole with equal forces

The effects that these boundary conditions had was that the stress concentration was much better for a symmetric object as can be seen on the image to the left in Figure 34. However a new problem is introduced. Since the linkage arm is not symmetric in the xy-plane, that is, the left and right halves are not identical, we get numerically unstable solutions when applying boundary condition 2 to a non-symmetric object; see the right image in Figure 35. This resulting instability might be explained by the fact that the linkage arm no longer has a fixed position in space and thus results in some sort of blurring of the stresses in the linkage arm as it starts to move, making the stress evaluation unreliable. We thus need a new approach which uses the progress we made in ensuring symmetry while at the same time prevents the linkage arm from moving in space.

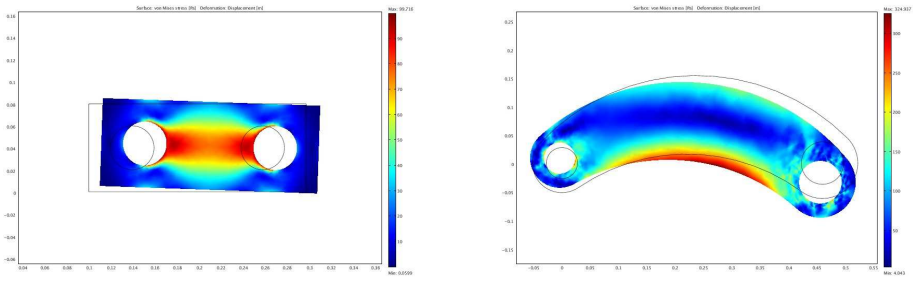


Figure 35: Results when pushing left hole and right hole with equal forces

Boundary Conditions 3 in 2D

Here the basic idea of applying forces to both holes is still used. Now in order to fix the linkage arm in place we need to introduce displacements into the boundary conditions. Instead of pushing inside the right hole with the same force as is in the left hole which resulted in an unstable solution we now look at the displacements, u_x on the edge of the right hole. To prevent it from moving we push against the displacement of the edge with a force $F = -ku_x$ where k is very high. This approach can be visualized in the same manner as before in Figure 34.

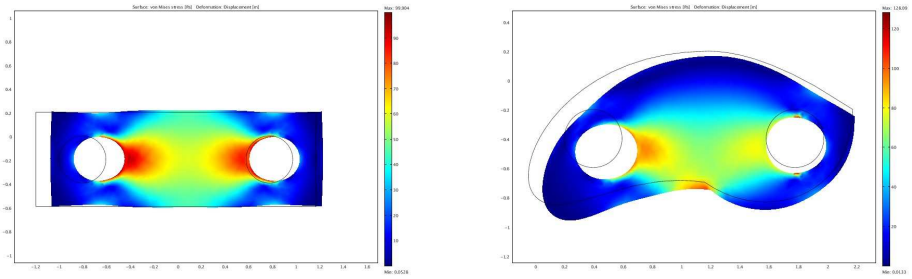


Figure 36: Results when pushing right hole with a proportional force

As can be seen in Figure 36 using boundary conditions 3 in 2D gives good symmetry in the stress concentration for symmetric objects while at the same time ensuring numerically stable solutions for non-symmetric objects. The linkage arm does not move as much as before and we do not get the blurred stress solution we got earlier. Thus with improved boundary conditions we are now able to move on to 3D where we will further explore the quality of the boundary conditions and add the requirements set forth earlier of imitating the real loading as well as possible.

5.3.2 Boundary Conditions in 3D

Boundary Conditions 1 in 3D

When applying the boundary conditions in 3D we used our experience from the 2D simulations and started with the approach that showed the best performance, that is, boundary conditions 3. However, after running some tests with this approach we realized a complication:

- We do not know in advance the magnitude of the force which is applied to the right hole as it is proportional to the displacement, u_x, which changes as the shape of the linkage arm varies.

We are thus not guaranteed that we are applying the same amount of force in every iteration which in turn might affect the resulting maximum stress in every iteration.It also limits the ability of simulating the true boundary conditions. Some sort of penalty function might have helped here but we did not take that approach. In order to implement satisfying boundary conditions we now focused on the true loading of the linkage arm and try to find an approach which would increase the resemblance while at the same time using the experience from previous boundary conditions.

Boundary Conditions 2 in 3D

The linkage arm we are researching is intended for use in a vehicle where its purpose is to move power from the chassis and to the tires. This means that the linkage arm will have two cylindrical objects placed inside each of its holes. These cylinders will then move in a longitudinal direction introducing forces

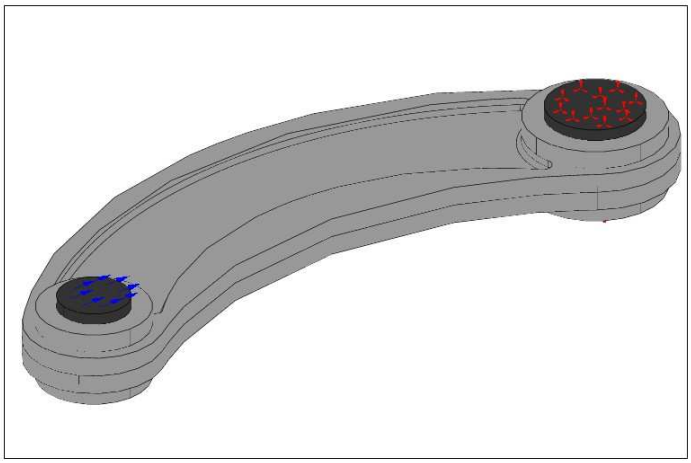


Figure 37: Loading with cylinder approach for forces

inside the linkage arm which carries them on to the other cylinder. We keep these settings in mind when we implement the final version of our boundary conditions. Instead of pushing directly on the linkage arm itself we place cylinders inside each of the holes in the linkage arm and apply a force on the cylinder inside the left hole while at the same time keeping the other cylinder in the right hole fixed in place, see Figure 37. We then use weakly formulated boundary conditions to force the boundary inside the holes of the linkage arm to move in the exact same way as the boundary of the respective cylinder. This gives good symmetry results and in our opinion is the best approach in imitating the true loading of the linkage arm.

5.4 Topology optimization modeling

Figure 38 displays the boundary conditions used in our TO runs. This approach was previously defined as boundary condition 1, described in Section 5.3.1. We were unable to use more sophisticated boundary conditions in our topology optimization program, as the software seemed not able to deal with more than one object at a time. The red color in Figure 38 illustrates points which have prescribed displacements, in our case these nodes are not allowed to move. The blue color illustrates points which have prescribed forces. The upper image shows how we model the inward force approach and the lower image shows how we model the outward force approach. We use the same approach in the 2D and 3D calculations.

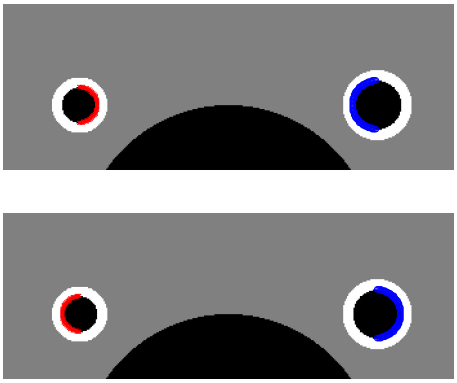


Figure 38: Boundary approach. Upper image: Push. Lower image: Pull.

In Figure 38 an obstacle has also been put on the figure, where material is not allowed in the final design of the linkage arm. Since we had no information on how this obstacle should looks in reality, we decided to let it take the form of a circle. The upper horizontal tangent of the circle lies through the center points of the linkage arm. Moreover, the vertical tangents also go through the two centers of the linkage arm circles.

We wanted to investigate the effects on the structure of the linkage arm, by changing the parameters of the penalty function and the filter radius, defined in Section 4.4. Since runs in the 3D version of TO++ (see Section 6.5) took considerably longer to perform than in the 2D version, we made most runs in 2D. Another reason was that despite some effort, we were unable to import solutions from the 3D version into Femlab for visualization. In 3D, we made some experiments by changing the allowed volume, as allowing the same volume as the original linkage arm gave a solution, which was very far from the original linkage arm. The results of these experiments are shown in Section 7.2.

The size of the two holes and the prescribed material can be found in Section 5.1. Other parameters which we used in our runs are included in Table 2.

Description of parameter	*Value*
Allowed width of box	0.7 m
Allowed height of box	0.26 m
Allowed thickness (for 3D runs only)	0.0294 m
Position of the center of the left hole	(0.119 m,0.1 m)
Position of the center of the right hole	(0.581 m,0.1 m)
Position of the center of the obstacle	(0.350 m,-0.131 m)
Mesh size	0.002 m
Minimum allowed density	$1 \cdot 10^{-9}$

Table 2: Some additional parameters for the topological optimization models

In the 3D approach we add thickness to the prescribed circles, and expand the boundary condition in a natural way to 3D.

About our objective function in TO

In our runs we used the software program TO++, which we describe in more detail in Section 6.5. The program has several implemented solvers for various objective functions, but unfortunately our objective function was not implemented. We decided to use another related objective function instead, namely to minimize the compliance (see Section 6.4). The original idea was to use the solution provided as a starting solution later for our objective. We were however not able to make use of this idea.

6 Programs

6.1 Matlab and Femlab/Comsol Multiphysics

Matlab is a well known and versatile programming language used in various fields within the academic world and in the industry. Matlab is generally criticized for being slow compared to programs such as C++, but as its syntax is simple it is often a good starting point when getting acquainted with mathematical problems.

The most vital tool we used in the work of this thesis was Comsol Multiphysics, also known as Femlab, which is a Matlab oriented programming language. Matlab and Femlab are user friendly and easy to use, making them popular choices for solving various problems. The software has a graphical interface, and is well documented.

The authors had good knowledge of Matlab before the work of this thesis started, but had no experience with Femlab. It is useful when solving PDE:s numerically. Below we give an example of how Femlab can be used. We will look at a simple example of a steel plate with some boundary conditions, how it is constructed and how one performs the numerical calculations. This simple example highlights the greatest drawback of our objective function, the minimization of the maximum stress. Figure 39 shows eight images of a steel plate in Femlab, in various forms. We will now discuss these images:

1. The first image shows a 2D plate which has been constructed in Femlab.
2. The second image (the right of the first one) shows how we have defined some boundary conditions. We keep the two lines in the bottom left hand corner of the steel plate fixed, and apply a load on the curved boundary on the upper right half.
3. The next figure displays the mesh obtained when Delauny's triangularization algorithm (see Section 3) is applied using the default parameters. This mesh is then used for numerical calculations.
4. The next figure illustrates how the stress in the steel plate looks like with the boundary conditions described earlier. The maximum stress is calculated to be 84 Pa.
5. Here, we see another solution, when we have increased the number of elements by a factor four. The maximum stress is now calculated to be 113 Pa.
6. Now we have increased the number of elements again, and this time the maximum stress is calculated to be 129 Pa.
7. Here we use a very fine mesh and now the maximum stress is as high as 212 Pa.
8. In the final image, a 3D perspective of the solution is displayed. We can see that there is a great spike where the maximum stress is calculated.

The example above illustrates how sensitive our objective function is to the implemented boundary conditions. If we had unlimited computer power and could use arbitrarily many elements for our calculation in Femlab, the maximum stress calculated would tend to infinity. In reality this does not happen. The model we implemented Femlab is too simple for the objective function of minimizing the maximum stress, and a more advanced mechanical model should be introduced in order to model the plasticity of that would occur. For other objective functions, like minimizing the compliance (see Section 6.4), the results are not as dependent on physical approximations as for the maximum stress objective.

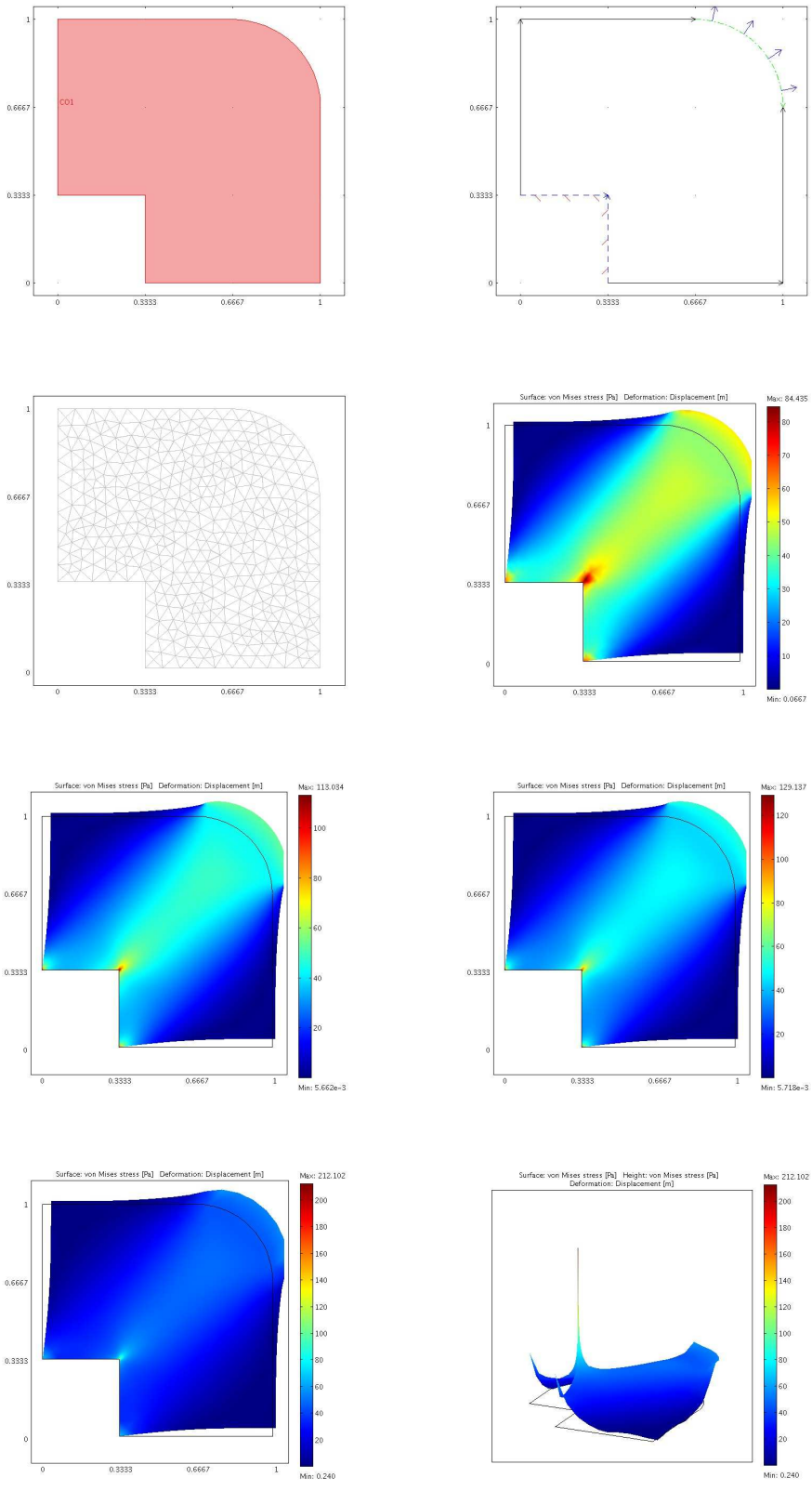


Figure 39: Steel plate example

6.2 NEWUOA

For the shape optimization problem the Fortran optimization solver NEWUOA, see [12], was recommended by our supervisors. NEWUOA uses a trust region method to find a local minimum of a function $F : \mathbb{R}^n \rightarrow \mathbb{R}$ and the algorithm has been tested on various test problems by its inventor. He has concluded that it gives good performance for as many as 160 variables. It is provided for use in unconstrained optimization and forms a quadratic model Q of F by interpolation, thereby avoiding gradient calculations altogether. Defining all parameters of a quadratic model would require:

$$NPT = \frac{(N+1)(N+2)}{2} \tag{26}$$

interpolation conditions for N variables. The NEWUOA algorithm uses only $2N + 1$ interpolation points for the model Q and for the remaining freedom it uses a minimum Frobenius norm updating procedure where it minimizes the change to the second derivative matrix of the model Q. This approach is more efficient for large N since the amount of computations required in each iteration has been heavily reduced.

As NEWUOA is an unconstrained optimization algorithm all constraints for the specific problem need to be implemented into the objective function subroutine. NEWUOA allows the vector of design variables, $\mathbf{x}$, to take on any value as long as $\mathbf{x} \in \mathbb{R}^n$. Since NEWUOA works in an iterative fashion it requires that the objective function subroutine returns an objective value in each iteration. Thus, if we want to constrain our problem to specific values of the design parameters we need to implement these boundaries inside the algorithm. Our design of the linkage arm requires that certain design parameters lie in a specific interval since allowing the parameters to vary freely would lead to many infeasible designs which in turn would leave us with a worse description of the objective function. Therefore we decided to implement upper and lower bounds on the design parameters by means of a penalty function; see Section 4.1.4 for more details.

Having constrained all design variables by upper and lower boundaries still does not guarantee feasible solutions for the linkage arm. Thus, in each iteration we need to check the validity of the design parameters supplied from each model update in the NEWUOA algorithm and return a penalty if the unfeasible solution domain is entered. Additional model constraints such as volume constraints etc. need also be implemented by means of a penalty function in the subroutine that calculates the objective.

The original Fortran code for NEWUOA assumes that a subroutine is provided. This subroutine called CALFUN(N,X,F), takes as input the number N, of design parameters and their values, X. It returns the corresponding objective value, F, which is then used to increase the accuracy of the model Q.

As our modeling of the linkage arm requires more sophisticated methods for the function evaluation than what can be easily implemented in a Fortran subroutine, such as boundary conditions and weak formulation, we needed to write an interface between the Fortran program and the actual objective function evaluation which is implemented in Femlab. The mex capabilities of Matlab allow the communication between Fortran and Femlab.

6.2.1 Parameters

The NEWUOA implementation has some important parameters that can be set and modified according to the problem at hand. These parameters are:

- `N`: The number of design variables in the optimization. $N \geq 2$ is required.
- `NPT`: The number of interpolation conditions to be used in the interpolation Q of the the objective function F. $N + 2 \leq NPT \leq \frac{(N+1)(N+2)}{2}$ is required.
- `RHOBEG`: The initial value of the trust region radius. The suggested value is about one tenth of the greatest expected change in a design variable from its starting point.

- `RHOEND`: The final value of the trust region radius which should indicate the accuracy that is required in the final values of the variables.
- `MAXFUN`: An upper bound on the allowed number of function calls during the iteration process.

In addition to the parameters specific to NEWUOA there are parameters that need to be set to implement the constraints which apply in the specific problem. For our problem these parameters include a vector of initial values of the design variables, one vector representing the upper bound for the design parameters and another vector indicating the lower bound for the design parameters, a parameter for the maximal allowed volume of the design as well as some additional parameters for the penalty implementation.

6.3 MultiOb

The MultiOb evolution algorithm is written in C++ and is intended for use in multiobjective optimization; for further detail see [3]. It assumes constraints in the form of upper and lower bounds on the design variables, $\mathbf{x} \in \mathbb{R}^n$. The user needs to supply the boundary constraints for the design variables, $\mathbf{x}$, as well as the objective function, f. Like NEWUOA the MultiOb algorithm does not use any information from the objective function other than the objective value and regards it as coming from a black box. For optimization purposes the MultiOb algorithm uses evolution strategies; see Section 6.3.1.

The original C++ code for MultiOb requires one to provide all necessary objective function calculations in a subroutine:

```
eval(double* objs, double* parameters, problem* currentproblem)
```

The subroutine takes as input pointers to the current objective (`objs`), design parameter values (`parameter`) and the problem instance being solved (`currentproblem`). The MultiOb algorithm requires that this subroutine performs calculations on the design parameters and returns the objective values in the correct pointer address. Since our objective function is implemented in Femlab we need to find a way for the C++ algorithm to communicate with Femlab. Again this has been solved by using the mex capabilities of Matlab. Here we are however no longer able to call the algorithm from Matlab and instead we need to call Matlab from the C++ code. Again we need to make some modifications to the optimization algorithm and add a few new parameters.

We mentioned earlier that MultiOb uses evolution strategies for the optimization. We give a brief introduction to the concept based mainly on [5] and [6].

6.3.1 Evolutionary strategies

. Evolution strategies are optimization methods which are based on the ideas of evolution and adaptation. The process is started by creating a set of μ random feasible solutions to the optimization problem. This set of random solutions represent a population which is also a first generation of parents.

The design variables of parent solutions are then used to create λ offspring solutions. During the reproduction of an offspring random errors are allowed. These errors are called mutations and are performed by adding a random value with a Gaussian distribution to each of the design variables from the parent solution. To increase the variation in offspring solutions further, some of the offspring solutions exchange design parameter values in a process called recombination. All resulting offspring solutions are then evaluated and the best solutions are chosen as parents for the next iteration/generation. When selecting the best solutions for the next generation there are two methods available:

- In comma strategies, (','), parents live one generation only. Possibly more than λ offspring have to be generated to ensure a constant population size of μ feasible alternatives.
- In plus strategies, ('+'), parents can live more than one generation if fitter than their offspring.

The process is iterated until we reach a stop criterion such as maximum generations.

The process of selecting the best individuals for each generation is more complicated in multiobjective evolution strategies since there are more than one objective to look at. However the author of the MultiOb algorithm has tested the algorithm on various test problems with good results. Diversification, that is introducing worse solutions as parents in the next generation, can be used in order to increase the variety in a population.

6.3.2 Parameters

The MultiOb algorithm has some important parameters that can be set and modified according to the problem at hand. These parameters are:

- `vinit`: Vector of initial values for design parameters
- `vinf`: Vector of lower boundaries for design variables
- `vsup`: Vector of upper boundaries for design variables
- `no_parameters`: Number of design variables
- `no_objectives`: Number of objectives in optimization problem
- `no_generations`: How many generations to create
- `no_objectives`: Numbers of objectives in the optimization problem
- `pop_size`: Size of population (parameter mu in evolution strategies)
- `no_offspring`: Number of offspring to create, (parameter lambda in evolution strategies)
- `strat_type`: Type of strategy to use, (comma or plus strategy as in evolution strategies)
- `opt_type`: Type of optimization, (min: '-',max '+')
- `p_mut`: Mutation value for each of the design variables
- `p_mut_parameters`: Possible to mutate each design variable in a different way
- `p_rec`: The probability for recombination of offspring
- `p_rec2`: The probability for exchange of each design variable in case we recombine
- `select_objective`: parameter for multiobjective selection, a value > 0 indicates the selected criterion, (-1 = domgrade2; -2 = weighting objectives)

6.4 A 99 line TO code used to solve a classic example: The cantilever beam

In [15] a 99 line topology optimization code written in Matlab is given, which is used to find a structure of a cantilever beam so that its compliance is minimized. The minimum compliance problem is equivalent to maximizing the global stiffness of a structure and thus it is highly related to the minimization of the maximum stress. Minimizing the compliance of a structure results in an objective function which is both differentiable and convex, so it is a much easier problem to solve than the objective function we want to solve for the linkage arm. To be more precise, the minimum compliance problem can be stated as follows

$$\begin{cases} \min \mathbf{f^T u}, \\ \text{subject to} \\ \quad \mathbf{Ku} = \mathbf{f}, \\ \quad V \leq V_0, \\ \text{and} \\ x_0 \leq x_e \leq 1 \quad \text{for all elements } e, \end{cases} \tag{27}$$

where x_0 is a number very close to, but greater than zero. We denote by x_e the density of an element, e, where $x_e = 0$ means no density, and $x_e = 1$ means full density. The reason for not allowing x_e to take zero values has to be accounted for, since otherwise a solution cannot be guaranteed and optimization algorithms experience numerical problems. The volume constraint, V_0, is a number less than or equal to the volume of the design domain, and $V = \sum_e x_e V(x_e)$, where $V(x_e)$ is the volume of element e. $\mathbf{K} = \mathbf{K}(\mathbf{x})$ is the stiffness matrix and $\mathbf{f}$ is the load vector, defined in Section 3.

The formulation stated in equation (27) is a special case of the general structural optimization problem stated in equation (20), where the total potential energy is given with

$$\Pi(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{f}^T\mathbf{u}$$

in this case. By taking the derivative of 6.4 we get the equation criteria stated in equation (27).

Finding a structure, such that its maximum stress is minimized is in fact a neither convex nor differentiable optimization problem.

It is easy to modify the 99 line program for it to solve many other topological problems. In the article [15] many interesting 2 dimensional problems are stated and solved. The program is available online [15] and is an elegant starting point for those who are not familiar with the concept. To run the program, one simply types

```
top(nelx,nely,volfrac,penal,rmin)
```

in Matlab, where

1. `nelx` is the width of the design square,
2. `nely` is the height of the design square,
3. `volfrac` is the allowed fraction of filled material,
4. `penal` is the penalty power p in the first line in equation (21),
5. `rmin` is the filter radius divided by the mesh size.

A schematic figure of the cantilever beam is shown in Figure 40.

The default input parameters are as follows:

```
top(60,20,0.5,3,1.5).
```

The result of the program is shown in Figure 42

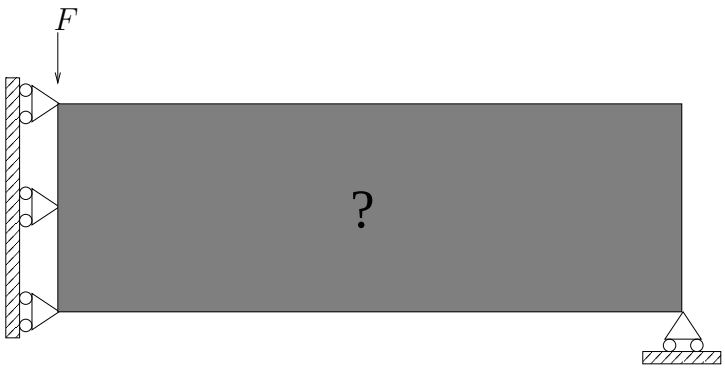


Figure 40: A cantilever beam: Initial solution, $x_e = 0.5$ for all elements

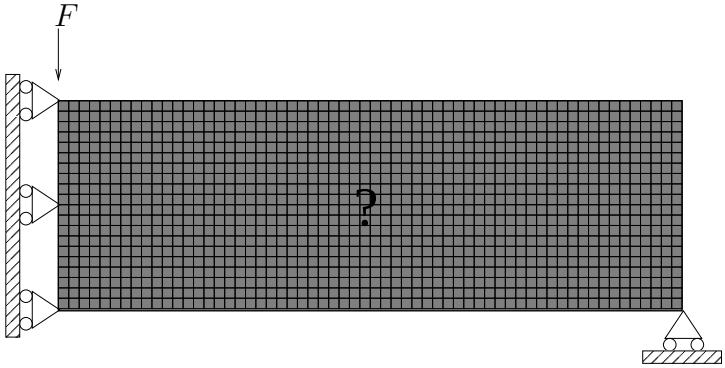


Figure 41: A cantilever beam: Meshing 60×20

Changing parameters for the cantilever beam

Figures 43–50 show solutions for the cantilever beam where the parameters have been varied in several ways. Most parameters are kept fixed in every figure, while we show the effect of changing one at a time. We next discuss in more detail the parameter settings used in the following figures.

1. In Figures 43 and 44 we have changed the mesh size. In Figure 43 we have a coarser mesh and in Figure 44 we have a finer mesh. As we can see we get an entirely new topology for the finer mesh.

2. In Figures 45 and 46 we have varied the filter radius. In Figure 45 we have filter radius equal to 1, and then we clearly see the checkerboard effect. In Figure 46 we have increased the filter radius and there we can see that the solution is not nearly as sharp as in the original solution.

3. In Figures 47 and in Figure 48 we have changed the allowed amount of volume. In Figure 47 we have less allowed volume (25%) and in Figure 48 we have more allowed volume (60%). As can be seen the topology is dependent on the allowed volume.

4. In Figures 49 and in Figure 50 we have changed the penalty parameters. In Figure 49 we have smaller penalty parameters which results in a higher amount of intermediate values and in Figure 50 we have a higher penalty parameter which results in a lower amount of intermediate values.

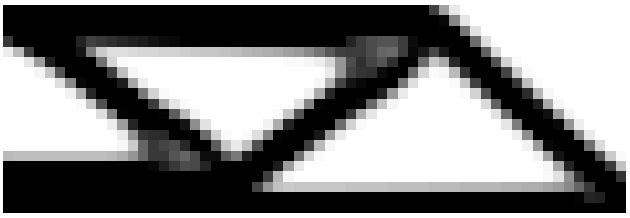


Figure 42: Solution for the minimum compliance of a cantilever beam obtained using a mesh of size 60×20 ($\texttt{nelx} = 60$, $\texttt{nely} = 20$), 50% allowed volume ($\texttt{volfrac} = 0.5$), penalty parameter 3 ($\texttt{penal} = 3$) and filter radius (divided by the mesh size) is 1.5 ($\texttt{rmin} = 1.5$). In Matlab: `top(60,20,0.5,3.0,1.5)`

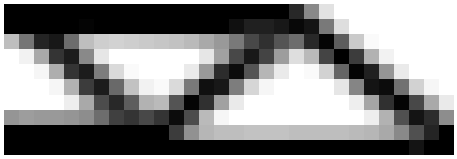


Figure 43: Decreased mesh size ($\texttt{nelx} = 30$, $\texttt{nely} = 10$)

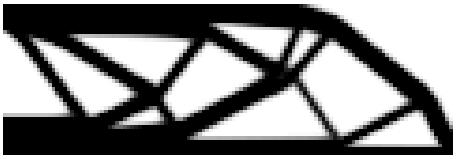


Figure 44: Increased mesh size ($\texttt{nelx} = 120$, $\texttt{nely} = 40$)

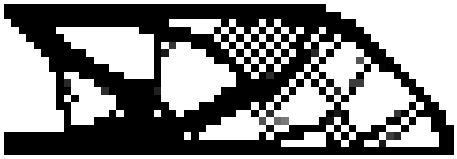


Figure 45: No filtering ($\texttt{rmin} = 1$)

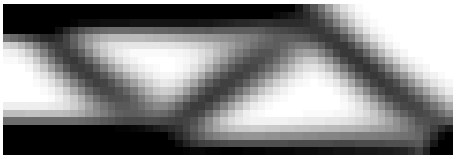


Figure 46: Higher filter radius ($\texttt{rmin}= 5$)

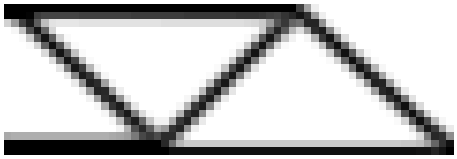


Figure 47: Decreased allowed volume fraction ($\texttt{volfrac} = 0.25$)

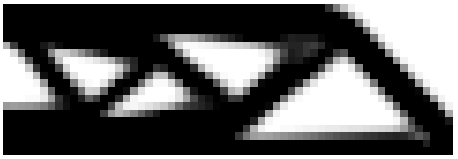


Figure 48: Increased allowed volume fraction ($\texttt{volfrac} = 0.6$)

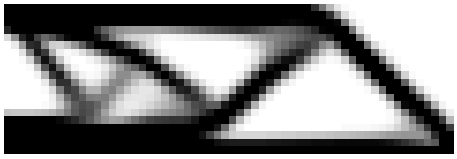


Figure 49: Lower penalty parameter ($\texttt{penal} = 2$)

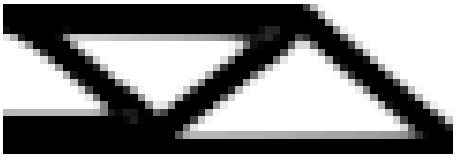


Figure 50: Higher penalty parameter ($\texttt{penal} = 5$)

In general there always exist some intermediate design values in our solution and one needs to decide how one goes from the discrete solution displayed in Figure 42 to a continuous solution which can be manufactured. One approach is to approximate the elements to their nearest integer (0 or 1, that is) and then fit smooth curves to the boundaries. We will not do this for the cantilever beam, but this is done for the linkage arm in Section 5.

It is not possible to say that the solution displayed in Figure 42 is the optimal solution to the cantilever beam problem. We have seen that changing the parameters has a great effect on the final solution. It is generally problem dependent how one should choose the parameters in order to get a good approximation of a design that is easy to manufacture.

6.5 TO++

In order to try to solve the problem of minimizing the maximum stress of a linkage arm using topology optimization, it was suggested to use a software, called TO++. In the manual this program is described in the following way:

TO++ is an object oriented computer code written in C++ aiming at solving continuum mechanical topology optimization problems [. . .]. *The program consists basically of a finite element part and an optimization part. These two parts can be combined with the purpose to solve topology optimization problems but can also be used individually to solve other types of problems.*

TO++ is far more advanced than the 99 line Matlab program described in Section 6.4. It has much more advanced optimization packages, including MMA [18]. It is far more difficult to use, and it always needs a quite amount of programming for every problem. No graphical interface is available. In this section a brief description is given how TO++ works from the user's point of view.

TO++ is constructed to solve 2D topological optimization problems. It is written in C++ by PhD Thomas Borrvall. A 3D version is also available, but it has less features and is more demanding for the user. The 3D version which was used in this project was able to solve the minimum compliance problem.

Despite discovering that the 2D version didn't have a solver to minimize the maximum stress, it was decided to spend a considerable amount of time preparing files for both the 2D and the 3D version of the program. Approximating boundary conditions, defined as boundary condition 1 in Section 5.3.1, were used despite knowing their weaknesses, as we were unable to model anything more appropriate. The idea was that the solution could later be used as an initial solution for a more sophisticated software, or to get some ideas of how to construct a new type of shape optimization problem. Minimizing the compliance is highly related to minimize the maximum stress. For more details on the compliance problem, see Section 6.4.

6.5.1 Design domains and making objects

In TO++ one works with a design domain, denoted by Ω, which is divided in three disjoint sets

$$\Omega = \Omega_{\text{varying}} \cup \Omega_{\text{material}} \cup \Omega_{\text{rigid}},$$

where

$$\Omega_{\text{varying}} \cap \Omega_{\text{material}} = \Omega_{\text{material}} \cap \Omega_{\text{rigid}} = \Omega_{\text{rigid}} \cap \Omega_{\text{varying}} = \emptyset.$$

When defining objects it is convenient to use a different notation, namely

$$\tilde{\Omega} = \tilde{\Omega}_{\text{varying}} \cup \tilde{\Omega}_{\text{material}} \cup \tilde{\Omega}_{\text{void}} \cup \tilde{\Omega}_{\text{rigid}},$$

where

$$\begin{aligned}
\Omega_{\text{rigid}} &= \tilde{\Omega}_{\text{rigid}} \\
\Omega_{\text{material}} &= \tilde{\Omega}_{\text{material}} \backslash \{\tilde{\Omega}_{\text{rigid}} \cup \tilde{\Omega}_{\text{void}}\} \\
\Omega_{\text{varying}} &= \tilde{\Omega}_{\text{varying}} \backslash \{\tilde{\Omega}_{\text{rigid}} \cup \tilde{\Omega}_{\text{void}} \cup \tilde{\Omega}_{\text{material}}\}.
\end{aligned}$$

6.5.2 Constructing a piece - an example

Figure 51 is an example how one constructs an object in the 2D version of TO++.

Below is an example which describes how a simple piece can be structured in the 2D version of TO++. The table below corresponds to the images in Figure 51. A pseudo code of how TO++ works is also given in the appendix.

Add varying regions 1. Add left triangle 2. Add right triangle 3. Add rectangle 4. Add rectangle	Add prescribed material regions 1. Add left circle 2. Add right circle 3. Add top circle
Add a void region 1. Add top circle	Add rigid regions 1. Add left circle 2. Add right circle
50×50 mesh is chosen	Elements are created - approximation

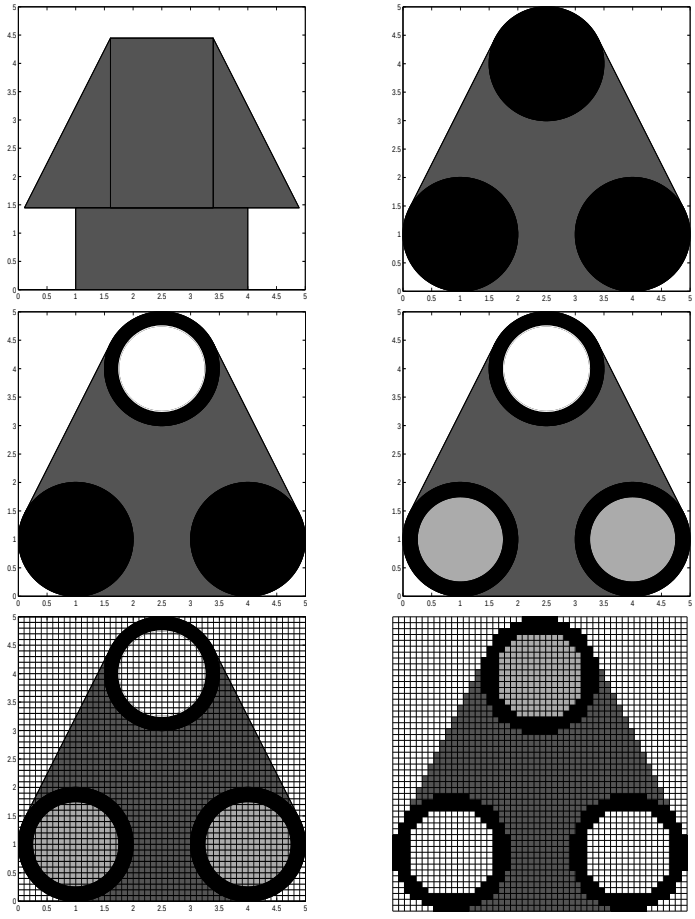


Figure 51: Making a piece

6.5.3 FE-solvers and the Methods of Moving Asymptotes (MMA)

As stated before, several FE-solvers are found in TO++ and also several optimization algorithms. The discussion about these solvers and algorithms are omitted in this thesis. However, we want to mention one of the optimization algorithms, namely the Methods of Moving Asymptotes (MMA) [18]. This algorithm is the one which is used in both the 2D and the 3D version of TO++. The FE-solver used is the conjugate gradient method for symmetric and positive definite systems.

6.5.4 Going from TO++ to Femlab

TO++ only outputs the final structure for an implemented model. It is of interest to see how the stress is distributed in the final design. We were able to implement a procedure for doing this in 2D, but had difficulties with the 3D designs. In Figure 52 we show how a solution from TO++ is manipulated in order to obtain a structure which can be used in Femlab. The first image in Figure 52 shows a solution from one of our 2D runs from TO++. Details will be omitted here but are presented in Section 7.2. We start by making the figure purely black-and-white, by approximating the density values with zero or one. This is shown in the second image in Figure 52. Then we run a boundary algorithm in order to obtain the boundary of the structure, and the result is shown in the third image. As we were unhappy with the smoothness of the boundary, we applied some smoothing to it, obtaining the structure shown in the fourth image. This structure contains 14 different objects which can be put into Femlab for calculation. The result of this particular example is shown in Section 7.2.

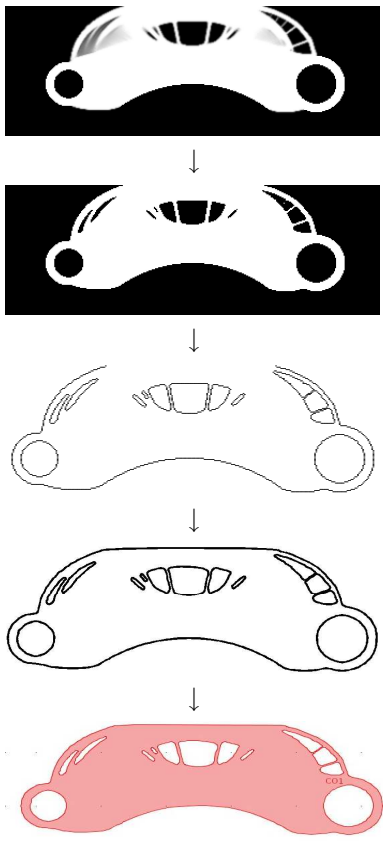


Figure 52: Going from TO++ to Femlab

7 Results

Here we will present our main results. We start by some results obtained using shape optimization, then present results using topology optimization, and end by showing results for an improved shape optimization model.

7.1 Shape optimization results

7.1.1 Results 2D

For the 2D model of the linkage arm we have selected $N = 6$ design variables which we feed into NEWUOA and MultiOb. These design variables, displayed in Figure 21, are:

- x_{out}, the x-coordinate of the circle representing the outer boundary,
- y_{out}, the y-coordinate of the circle representing the outer boundary,
- r_{out}, the radius of the circle representing the outer boundary,
- x_{inn}, the x-coordinate of the circle representing the inner boundary,
- y_{inn}, the y-coordinate of the circle representing the inner boundary,
- r_{inn}, the radius of the circle representing the inner boundary.

We start by supplying the initial parameter values of the linkage arm from the model data sheet [19]. Since we do not want to produce too many infeasible linkage arms we define boundary constraints for the design variables which limit the variation in each design variable to about $\pm 10\%$:

Design variable	Initial value	Lower bound	Upper bound
x_{out}	0.238	0.218	0.258
y_{out}	-0.225	-0.245	-0.205
r_{out}	0.380	0.350	0.410
x_{inn}	0.222	0.202	0.242
y_{inn}	-0.327	-0.357	-0.297
r_{inn}	0.343	0.313	0.373

Table 3: Initial values for the linkage arm design variables

For NEWUOA we normalize all design variables so that they are 0 when at their lower bound and 1 at their upper bound. All variables have 0.5 as a starting value and we use a trust region radius 0.2 with a final accuracy 10^{-6}, which is smaller than required. We create the quadratic model Q by interpolating the objective function in $2N + 1 = 13$ points and we allow NEWUOA to run a maximum of 150 function calculations.

For MultiOb we do not normalize the variables but create an initial population of 50 random solutions. In each iteration we create 20 new offspring and we select the next generation as the 50 solutions with the lowest objective values from both the parent and offspring solutions. The mutation change in an offspring design variable is in the range from 0 to 0.1 in each iteration. There is a 10% chance that two offspring solutions are recombined in each generation and if two offsprings are recombined there is a 25% chance for each design variable to be exchanged. There is, in general, no obvious relationship between the parameter values and the solution quality except that an increasing population size and an increasing number of generations will improve the quality while at the same time increasing the computational effort. Here we have used parameter values that gave good results within a given time frame.

We apply the boundary conditions 3 in 2D from Section 5.3.1 and run both algorithms with penalties for infeasible solutions only, that is, for inconstructable linkage arms. The penalty implementation is discussed in Section 4.1.4. The resulting linkage arm along with three intermediate solutions is shown in Figure 53, for NEWUOA. The stress concentration for the best solution is plotted twice for NEWUOA, first at a scale that represents the maximum stress in dark red and minimum stress in dark blue and then we change the scale so that dark red is the maximum stress in the initial linkage arm. This is done to emphasize that even though the best solutions for the following iterations are often dark red we need to look at the stress in perspective. In Figure 54 we see the linkage arm which has the minimum maximal stress for an inward force in the MultiOb iteration.

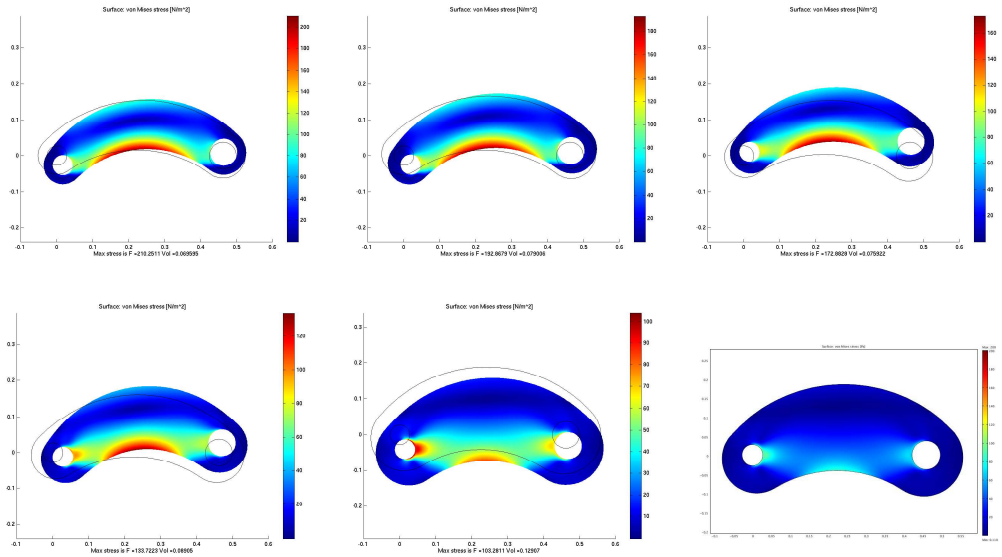


Figure 53: Results for NEWUOA when minimizing the maximum stress using the boundary conditions 3 in 2D with an inward force applied. No volume penalty is applied as can be seen by the size of the resulting linkage arm. The last image has the same color scale as the first one.

Both NEWUOA and MultiOb return design variable solutions that result in a lower value of the objective compared to the initial design. If we do not implement a penalty for the volume restriction, both algorithms will return solutions that have increased the total volume of the linkage arm.

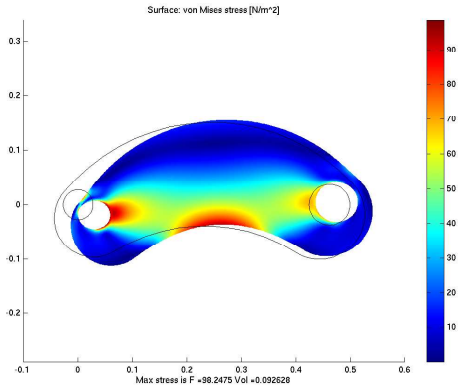


Figure 54: Results for MultiOb when minimizing the maximum stress using the boundary conditions 3 in 2D with an inward force applied. No volume penalty is implemented. The resulting linkage arm has increased in volume compared to the initial solution.

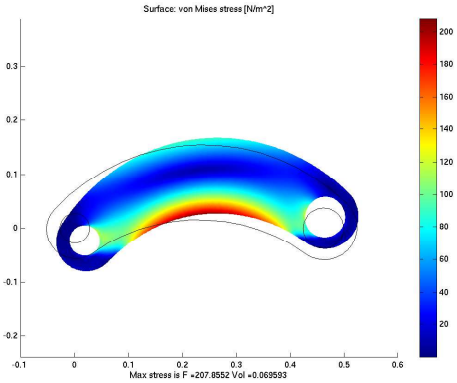


Figure 55: Results for NEWUOA when minimizing the maximum stress using the boundary conditions 3 in 2D with an inward force applied. A volume constraint is implemented by means of a penalty function.

Next we add a volume constraint, that the volume of the initial design should not be exceeded, to the problem and iterate again with both algorithms. The resulting linkage arm is shown in Figure 55 for NEWUOA and in Figure 56 for MultiOb. We do not show intermediate results.

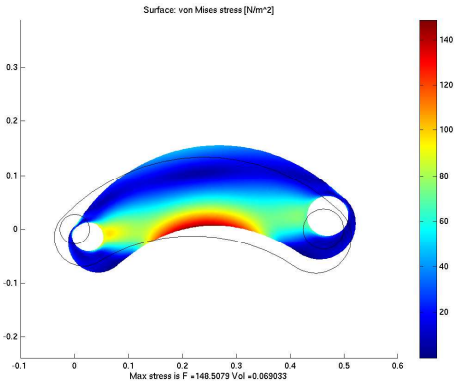


Figure 56: Results for MultiOb when minimizing the maximum stress using the boundary conditions 3 in 2D with an inward force applied. A volume constraint is implemented by means of a penalty function.

Even with a volume restriction still both algorithms are able to decrease the maximum stress. NEWUOA is not quite as efficient as MultiOb and returns a 1.1% decrease in the objective value which could be a local minimum. MultiOb does a lot better with a 29.4% decrease in the maximum stress value. These results represent only one implementation parameter setting in both algorithms and it would be interesting to vary the parameters to see if we are able to obtain a better solution. Further, for MultOb it would be of interest to optimize for pulling as well as pushing in order to utilize the multiobjective capabilities of the algorithm.

7.1.2 Results 3D

We start by looking at the solutions for modeling approach 1, discussed in Section 5.2.2. In the iteration stage of this modeling approach, 10 design variables are allowed to change in each iteration in NEWUOA. The model is subject to the boundary conditions 1 in 3D which are explained in detail in Section 5.3.2. In Figure 57 we see the resulting linkage arm as well as some intermediate iteration results in NEWUOA. We display the intermediate results here as well since these let us to doubt the boundary conditions.

After reformulating the boundary conditions we look at results from modeling approach 2 in 3D which are explained in Section 5.2.2. We use 10 design variables and apply boundary conditions 2 in 3D which are

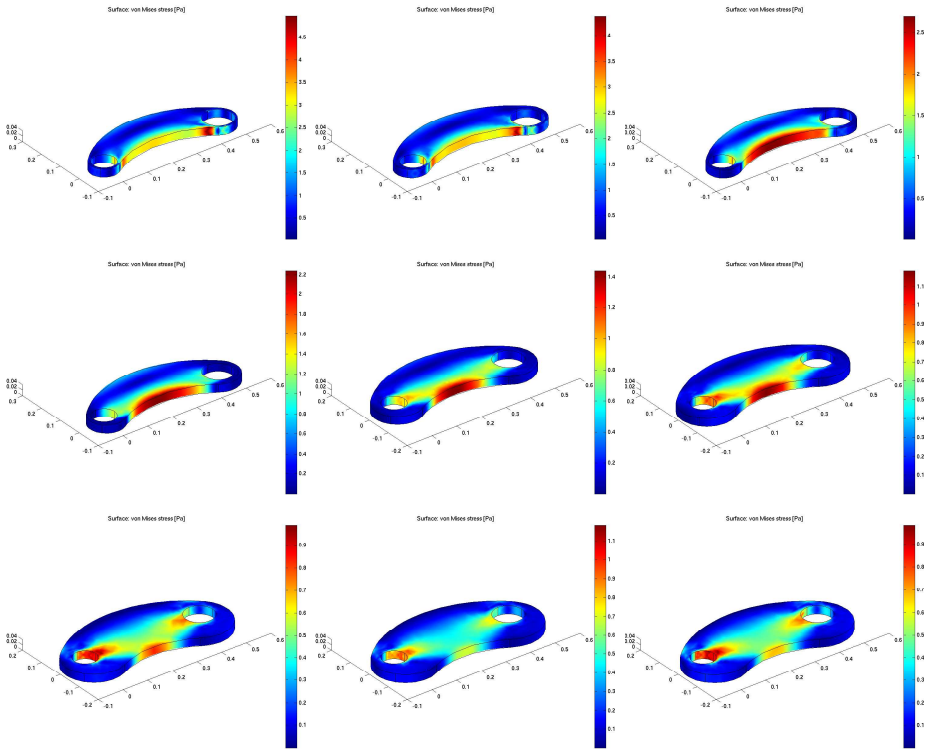


Figure 57: Results for NEWUOA using the proportional force approach represented in boundary conditions 1 in 3D (Section 5.3.2). The resulting best linkage arm has an increased volume and it is interesting to see how the stress concentration moves as the shape changes

discussed in further detail in Section 5.3.2. We start by minimizing the maximum stress when applying an inward force on the cylinder and then we run another test where we minimize the maximum stress while applying an outward force on the cylinder. For both optimization approaches we display the best linkage arm and its stress concentration for both pulling and pushing regardless of the objective value. In Figure 58 we can see the resulting linkage arm when optimized for pushing and in Figure 59 we see the stress concentration in the same linkage arm with a pull force applied. There is no volume restriction applied and the resulting linkage arm has increased in volume by making the inner section that is removed smaller. The final values of the design variables can be seen in Table 4 where the initial values are normalized with value 0.5.

Since we did not manage to decrease the maximum stress in the linkage arm when applying an outward force, we run another iteration with the same model in NEWUOA, but now we optimize for the pull approach. There is still no restriction on the volume of the final design and again we plot the resulting stress concentration for both outward force and inward force approaches. In Figure 61 we can see the resulting linkage arm when optimized for pushing and in Figure 60 we see the stress concentration in the same linkage arm with a pull force applied. The final values of the design variables can be seen in Table 5 where the initial values are normalized with value 0.5.

The final values of the design variables in Tables 4–5 are rather close to the initial values. In each iteration only one design variable has changed significantly and the best linkage arm had less volume than the initial design in one case. We decided to wait a little longer with implementing the volume constraints and ran two more tests now hoping to decrease the objective value by adding the radius of the thickening to the set of design variables resulting in a total of 12 variables; see Section 5.2.2 for possible design variables.

In Figures 62–65 the resulting linkage arm and stress concentrations for both inward and outward force applications are shown. Tables 6–7 show the final values of the 12 design parameters. We noticed that the

resulting design parameters in the best solutions for 12 design variables are similar to the solutions for 10 design variables. We decided that the modeling approach 5 in 3D, which was used in these iterations, was too limiting for the design and thus we implemented a new approach, modeling approach 6 in 3D, which is explained in further detail in Section 5.2.2 and used in the remaining test runs.

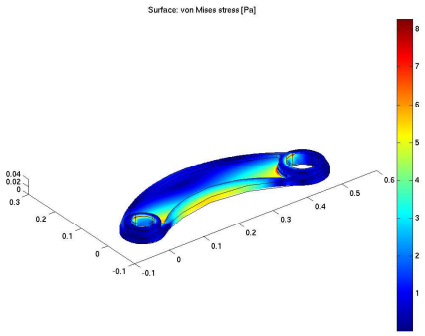


Figure 58: Results for NEWUOA when minimizing the maximum inward stress using the cylinder force approach represented in boundary conditions 2 in 3D (Section 5.3.2). The force is applied in an inward direction and even though there is no volume constraint the resulting best linkage arm has less volume than the initial design.

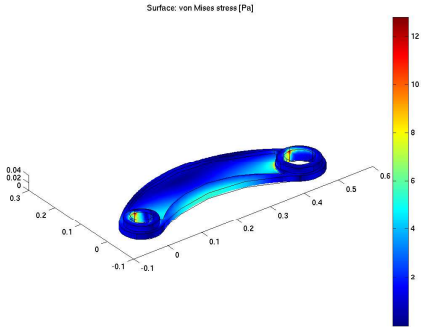


Figure 59: The same linkage arm as in Figure 58 but now we display the resulting stress concentration when applying the force in an outward direction. Since this linkage arm is optimized for the inward approach we have not reduced the maximum stress here.

Design variable	Normalized final value	Value increase(%)
x_{out}	0.5038	0.7521
y_{out}	0.4985	-0.2933
r_{out}	0.4994	-0.1191
x_{inn}	0.4983	-0.3324
y_{inn}	0.5001	0.0132
r_{inn}	0.4990	-0.2036
y^i_{out}	0.7008	40.1688
r^i_{out}	0.4997	-0.0604
y^o_{inn}	0.5002	0.0413
r^o_{inn}	0.4996	-0.0711

Table 4: Resulting design variables when running NEWUOA and optimizing for an inward force approach without volume constraints. The resulting linkage arm can be seen in Figure 58. Only one design variable, y^i_{out}, has changed significantly. This has the effect that the resulting linkage arm has a thinner upper edge than the initial linkage arm.

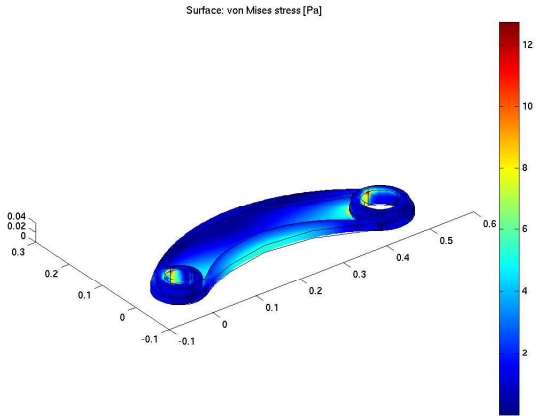


Figure 60: Results for NEWUOA when minimizing the maximum outward stress using the cylinder force approach represented in boundary conditions 2 in 3D (Section 5.3.2). The force is applied in an outward direction, and since there is no volume constraint the resulting best linkage arm has a larger volume than the initial design.

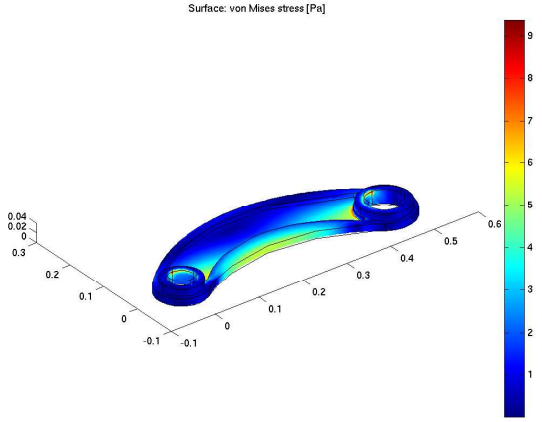


Figure 61: The same linkage arm as in Figure 60 but now we display the resulting stress concentration when applying the force in an inward direction. Even though this linkage arm is optimized for the outward approach we have reduced the maximum stress.

Design variable	Normalized final value	Value increase(%)
x_{out}	0.5003	0.0556
y_{out}	0.4996	-0.0829
r_{out}	0.4996	-0.0830
x_{inn}	0.4999	-0.0105
y_{inn}	0.5005	0.0931
r_{inn}	0.4997	-0.0625
y^i_{out}	0.4996	-0.0822
r^i_{out}	0.5002	0.0393
y^o_{inn}	0.6983	39.6657
r^o_{inn}	0.4994	-0.1123

Table 5: Resulting design variables when running NEWUOA and optimizing for an outward force approach without volume constraints. The resulting linkage arm can be seen in Figure 60. Only one design variable, y^o_{inn}, has changed significantly. This has the effect that the resulting linkage arm has a thicker lower edge than the initial linkage arm.

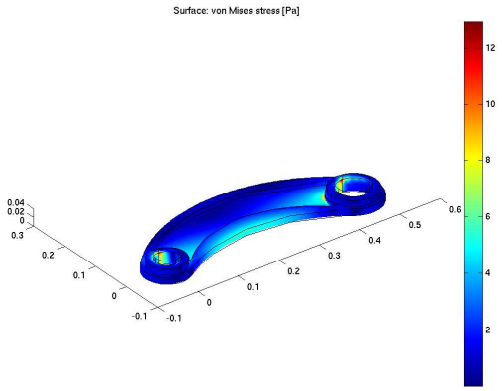


Figure 62: Results for NEWUOA when minimizing the maximum outward stress for 12 design variables using the cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2). The force is applied in an outward direction, and since there is no volume constraint the resulting best linkage arm has a larger volume than the initial design.

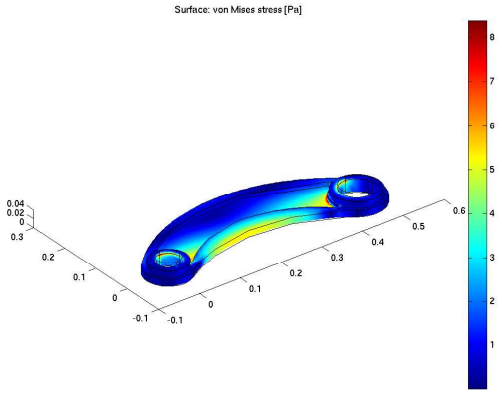


Figure 63: The same linkage arm as in Figure 62, but here we display the resulting stress concentration when applying the force in an inward direction. Even though this linkage arm is optimized for the outward approach we have reduced the maximum stress here.

Design variable	Normalized final value	Value increase(%)
x_{out}	0.4995	-0.1057
y_{out}	0.5005	0.0956
r_{out}	0.4985	-0.3036
x_{inn}	0.5012	0.2354
y_{inn}	0.5029	0.5886
r_{inn}	0.5030	0.5986
y^i_{out}	0.4981	-0.3867
r^i_{out}	0.4999	-0.0166
y^o_{inn}	0.6879	37.5867
r^o_{inn}	0.4977	-0.4549
r^{out}_{hr}	0.4983	-0.3355
r^{out}_{hl}	0.5169	3.3749

Table 6: Resulting design variables when running NEWUOA and optimizing for an outward force approach without volume constraints. The resulting linkage arm can be seen in Figure 62. Still, as was the case when we had 10 design variables, only one design variable, y^o_{inn}, has changed significantly. This has the effect that the resulting linkage arm has a thicker lower edge than the initial linkage arm.

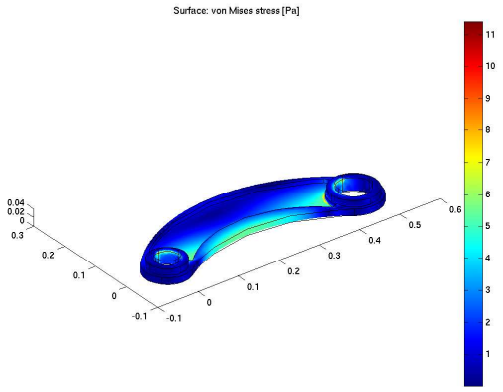


Figure 64: Results for NEWUOA when minimizing the maximum inward stress for 12 design variables using the cylinder force approach represented in boundary conditions 2 in 3D (Section 5.3.2). The force is applied in an inward direction and even though there is no volume constraint the resulting best linkage arm has less volume than the initial design.

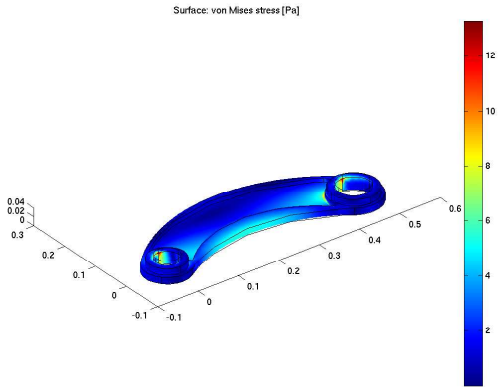


Figure 65: The same linkage arm as in Figure 64 but now we display the resulting stress concentration when applying the force in an outward direction. Since this linkage arm is optimized for the inward approach we have not reduced the maximum stress here.

Design variable	Normalized final value	Value increase(%)
x_{out}	0.5001	0.0275
y_{out}	0.5000	0.0047
r_{out}	0.4999	-0.0152
x_{inn}	0.5199	3.9882
y_{inn}	0.4989	-0.2201
r_{inn}	0.4990	-0.2046
y^i_{out}	0.7000	39.9914
r^i_{out}	0.4995	-0.1017
y^o_{inn}	0.5005	0.0905
r^o_{inn}	0.4996	-0.0882
r^{out}_{hr}	0.4993	-0.1375
r^{out}_{hl}	0.4993	-0.1447

Table 7: Resulting design variables when running NEWUOA and optimizing for an inward force approach without volume constraints. The resulting linkage arm can be seen in Figure 64. Still, as was the case when we had 10 design variables, only one design variable, y^i_{out}, has changed significantly. This has the effect that the resulting linkage arm has a thinner upper edge than the initial linkage arm.

Before running further tests we add volume constraints to the objective function. This is done by a penalty function; for more details see Section 4.1.4. We change our modeling approach and use modeling approach 3 in 3D; for further details we refer to Section 5.2.2. We run iterations and optimize for both an inward and an outward force in NEWUOA. Figure 66 shows the resulting best linkage arm optimized for an inward approach and in Figure 67 the resulting linkage arm for the outward approach is displayed. If we compare the results it is interesting to see that the best solutions in both cases do not vary that much for the maximum outward stress. However, using this new modeling approach results in much higher initial values for the maximum stresses, so due to the sensitivity of the objective function it might be unfair to compare the actual resulting values.

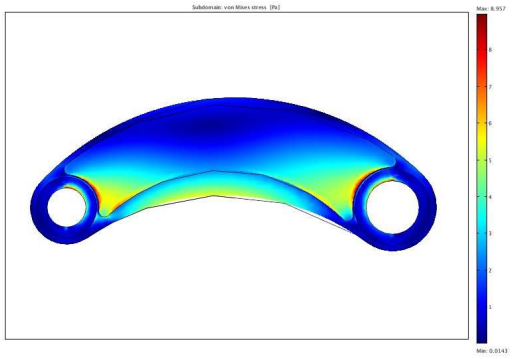


Figure 66: Results for NEWUOA when minimizing the maximum inward stress for the modeling approach 6 (Section 5.2.2) using the cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2). The resulting maximum stress for an inward force is $F_{inn} = 7.83$Pa and for the outward force $F_{out} = 13.15$Pa. The resulting volume in the best design is; $Vol = 1.034 \cdot 10^{-3} m^3$.

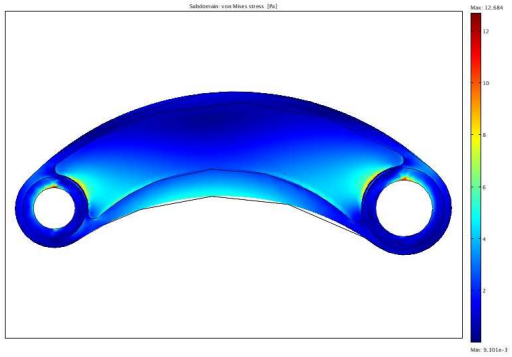


Figure 67: Results for NEWUOA when minimizing the maximum outward stress for the modeling approach 6 (Section 5.2.2) using the cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2). The resulting maximum stress for an outward force is $F_{out} = 12.81$Pa and for the inward force $F_{inn} = 9.68$Pa. The resulting volume in the best design is; $Vol = 1.059 \cdot 10^{-3} m^3$.

When we use the NEWUOA algorithm we are limited by the fact that it is only capable of optimizing for one objective value at a time. We are interested to see the effects of optimizing for both inward and outward forces at the same time since the actual implementation of the linkage arm requires that it is optimized for both cases. In Section 6.3 we introduced MultiOb which is an algorithm that has the capability to optimize for more than one objective value at a time. Since the initial test runs with NEWUOA did return linkage arms with decreased volume we decided to remove the volume constraint from the first iteration and use the volume as an objective function to be minimized along with the maximum stresses. The resulting linkage arm can be seen in Figure 68 and it is clear that the reduction in the objective for the stresses is superior to all linkage arms with less volume.

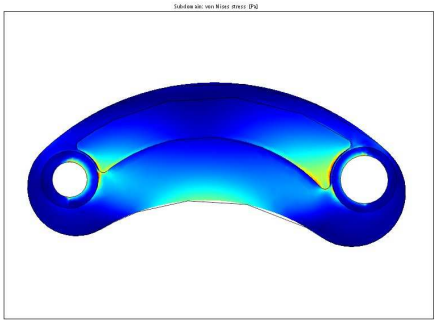


Figure 68: Results for MultiOb when minimizing the maximum outward stress, the maximum inward stress and the volume of the linkage arm all at the same time. The cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2) is used as boundary conditions for the modeling approach 6 (Section 5.2.2). The decrease in stress is dominant over the decrease in volume and thus the best linkage arm has more volume than the initial design. The resulting objective values are; $F_{inn} = 5.30$Pa, $F_{out} = 9.30$Pa and $Vol = 1.551 \cdot 10^{-3} m^3$.

To counter this volume increase we run another iteration where we implemented a volume constraint on the linkage arm design. The resulting best linkage arm is displayed in Figure 69 along with the objective values of the best solution.

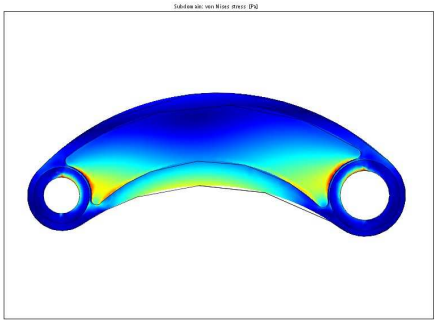


Figure 69: Results for MultiOb when minimizing the maximum outward stress and the maximum inward stress both at the same time. The cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2) is used as boundary conditions for the modeling approach 6 (Section 5.2.2). The resulting linkage arm can not have a larger volume than the initial design. The resulting objective values are; $F_{inn} = 8.80$Pa and $F_{out} = 12.94$Pa. The volume of the best design is; $Vol = 1.087 \cdot 10^{-3} m^3$.

In Table 8 we can see the resulting objective values for all iterations.

Algorithm (objectives)	F_{inn} [Pa]	F_{out} [Pa]	Volume [m^3]
Initial design (Modeling approach 5)	10.52	15.29	$1.086 \cdot 10^{-3}$
NEWUOA (F_{inn},$n = 10$)	7.44	11.69	$1.053 \cdot 10^{-3}$
NEWUOA (F_{out},$n = 10$)	9.03	11.80	$1.098 \cdot 10^{-3}$
NEWUOA (F_{out},$n = 12$)	7.96	11.45	$1.095 \cdot 10^{-3}$
NEWUOA (F_{inn},$n = 12$)	8.72	13.02	$1.054 \cdot 10^{-3}$
Initial design (Modeling approach 6)	37.93	36.21	$1.087 \cdot 10^{-3}$
NEWUOA (F_{inn})	7.83	13.15	$1.034 \cdot 10^{-3}$
NEWUOA (F_{out})	9.68	12.81	$1.059 \cdot 10^{-3}$
MultiOb (F_{inn},F_{out},$Volume$)	5.30	9.30	$1.551 \cdot 10^{-3}$
MultiOb (F_{inn},F_{out})	8.80	12.94	$1.087 \cdot 10^{-3}$

Table 8: Comparing results from NEWUOA and MultiOb in 3D

When comparing the results between NEWUOA and MultiOb it appears that NEWUOA returns better solutions than MultiOb. However, MultiOb uses dominance grade as an efficiency measure when ranking solutions. This means that when MultiOb has more than one objective value in an optimization problem, it ranks all solutions in a generation according to each one of the objectives, starting with rank 1 for the minimum. When this has been done the best solution is selected as the one solution that has the lowest rank sum from all the objective rankings. We can look at the set of best solutions from MultiOb and find solutions that are better than NEWUOA in minimizing both F_{inn} and F_{out}. Since MultiOb has to compromise between both objectives the best solution using MultiOb is worse than the best solution using NEWUOA for one objective. However if we were to add the best solution from NEWUOA to the set of best solutions in MultiOb we would still get the same linkage arm as we got earlier.

Using shape optimization we were able to come up with linkage arm designs which had a lower maximum stress than the current design of the linkage arm.

7.2 Topology optimization results

7.2.1 2D results

In Figures 70–73 various results are displayed. All the images in these figures have the following properties:

- The 9 images show the solution obtained by running the minimum compliance solver in TO++, with the penalty parameters (P, Q) (see equation (21)) set as

$(1,0)$	$(2,0)$	$(3,0)$
$(1,2)$	$(2,2)$	$(3,2)$
$(1,4)$	$(2,4)$	$(3,4)$

- The mesh size is $h = 0.002$m.
- Every image consist of $350 \times 125 = 43750$ elements.
- Allowed material is 40% of the design domain.

The images in Figures 70 and 71 have the following boundary properties, known as boundary condition 1 (see Section 5.3.1):

- The right half of the left hole is kept fixed.
- To the left half of the right hole is applied a constant force in the left direction.

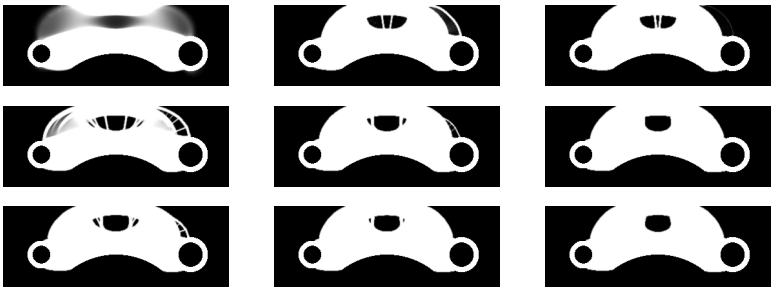


Figure 70: Results for different penalty parameters. Push approach. Filter radius is $1.5h$. From left to right: Q fixed, P increased. Top to bottom: P fixed, Q increased.

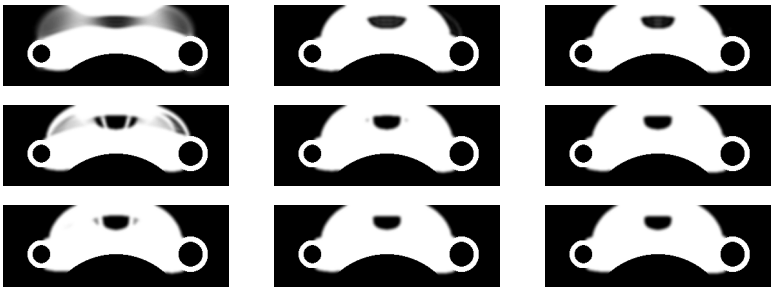


Figure 71: Results for different penalty parameters. Push approach. Filter radius is $5h$. From left to right: Q fixed, P increased. Top to bottom: P fixed, Q increased.

All the images in Figure 72 and in Figure 73 have the following boundary properties

- The left half of the left hole is kept fixed.
- To the right half of the right hole is applied a constant force in the right direction.

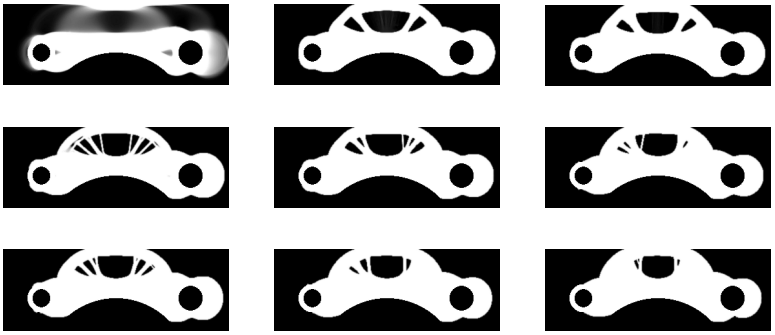


Figure 72: Results for different penalty parameters. Pull approach. Filter radius is $1.5h$. From left to right: Q fixed, P increased. Top to bottom: P fixed, Q increased.

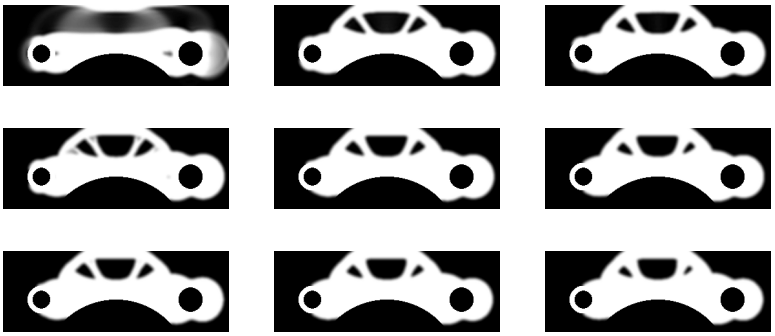


Figure 73: Results for different penalty parameters. Pull approach. Filter radius is $5h$. From left to right: Q fixed, P increased. Top to bottom: P fixed, Q increased.

From Figures 70–73 it seemed feasible to choose high penalty parameters and high filter radii, in order to obtain a structure which is easy to manufacture. We can see in Figures 72–73 that the boundary condition implemented leads to poor results, as the solution is very dependent on which of the two holes is kept fixed, and to which hole a load is applied.

7.2.2 An example of TO++ results in Femlab

In Section 6.5.4 it was described how the result from TO++ can be viewed in Femlab. Figure 74 shows the stress calculated in Femlab using the optimal design produced by TO++. The TO++ result is also shown in Figure 70 where the penalty parameters $(P, Q) = (1, 0)$.

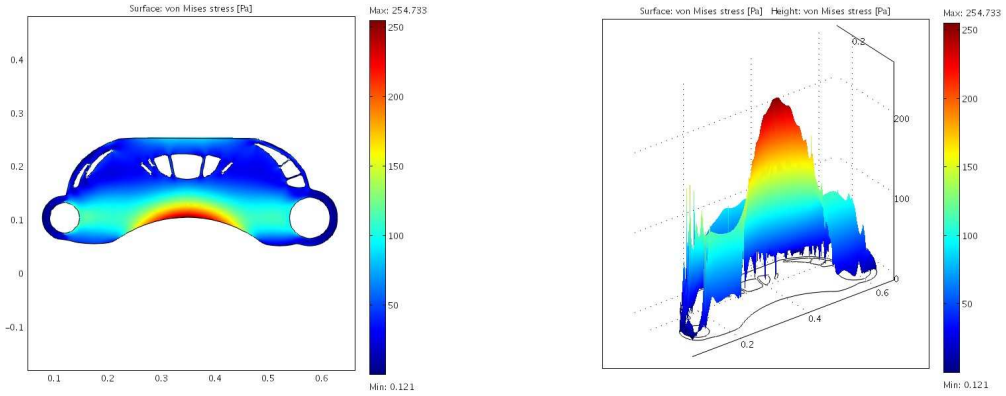


Figure 74: Von-Mises stress in a linkage arm computed in TO++

7.2.3 3D results

We will only show a few results for the 3D version of TO++. All the figures in this section are obtained using filter radius $2h$, where h is the mesh size. Every run had $350 \times 125 \times 15 = 656,250$ elements. Figures 75–77 display results for three volume fractions. The penalty parameters are set to $P = 2$ and $Q = 2$ in all the figures . We only display half of the linkage arm, due to symmetry properties. All the images on the left in the figures show the front view of the linkage arm, but the images on the right show how the linkage arms look like, when watched from the symmetry plane and outward.

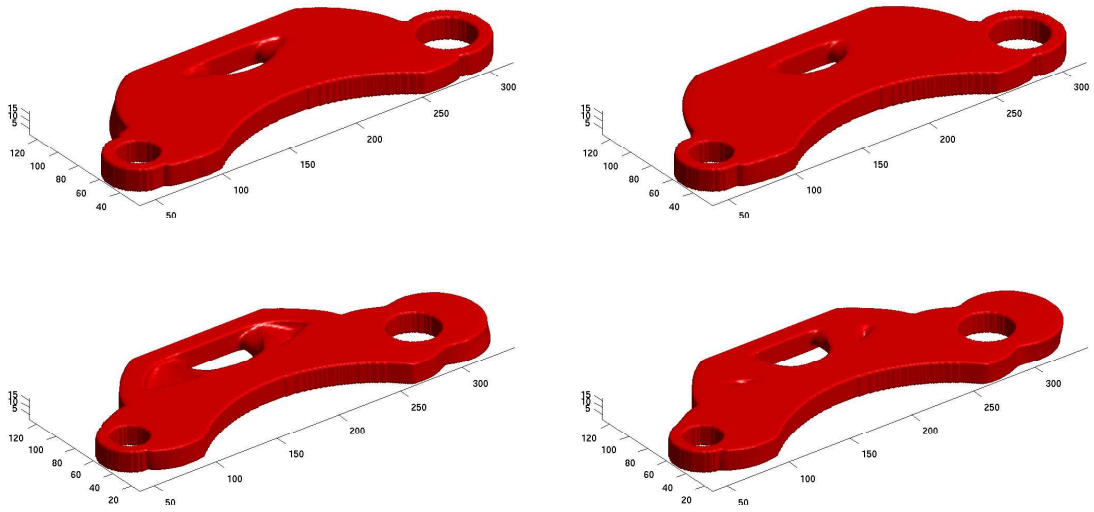


Figure 75: `volfrac` = 0.4. Upper: Pushing approach. Lower: Pulling approach. Left: Front view. Right: center view.

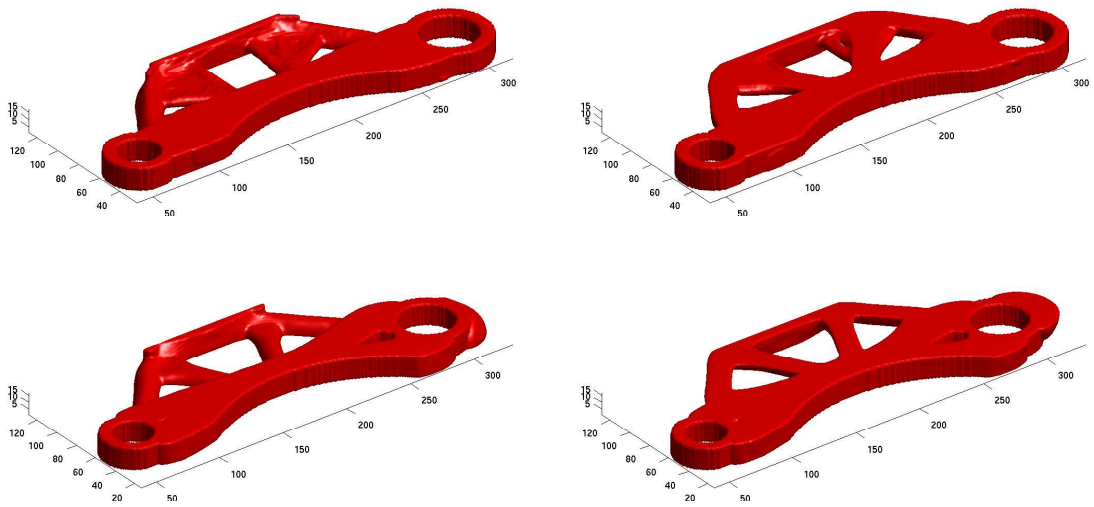


Figure 76: `volfrac` = 0.25. Upper: Pushing approach. Lower: Pulling approach. Left: Front view. Right: center view.

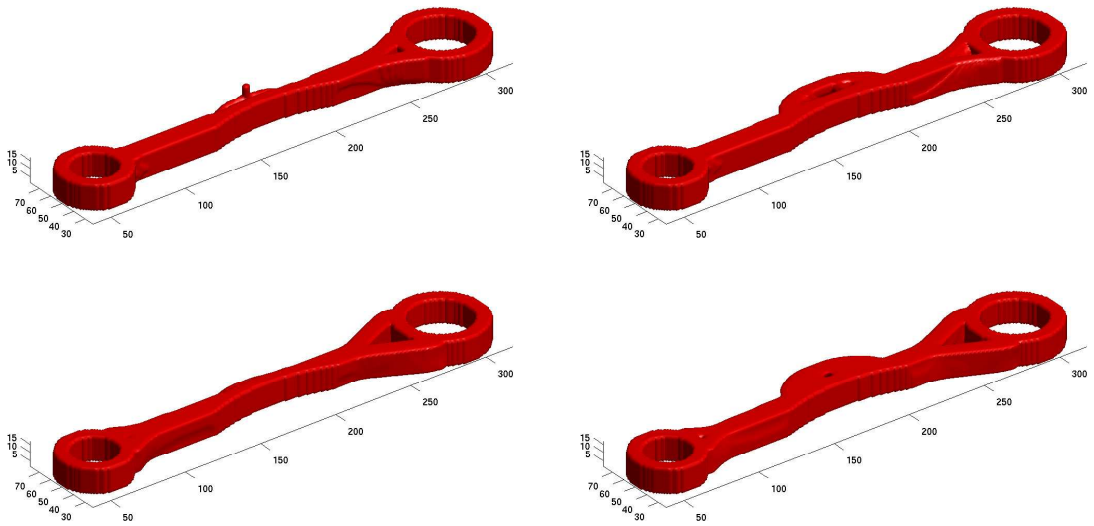


Figure 77: `volfrac` = 0.1. Upper: Pushing approach. Lower: Pulling approach. Left: Front view. Right: Center view.

We note that with the boundary conditions implemented, not so much is gained by using the 3D version of TO++ compared to the 2D version. However, it seems like a reasonable approach to add a hole in the linkage arm. Results from such an approach applied in shape optimization are shown in Section 7.3.

7.3 Results from shape optimization using topology optimization results

In order to combine the efforts of topology optimization and shape optimization we decided to construct a linkage arm with a hole. The resulting solution is displayed in Figure 78. It is worth mentioning that the best solution in this case is better than all previous solutions if we use dominance grade as the efficiency measure in MultiOb. It would be of great interest to also optimize over more parameters describing the void region in the middle of the linkage arm, which might even yield better designs.

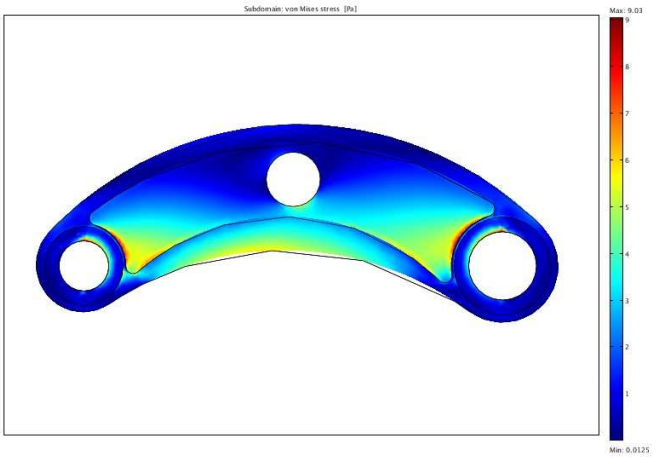


Figure 78: Results for MultiOb when minimizing the maximum outward stress and the maximum inward stress both at the same time. The cylinder force approach represented in the boundary conditions 2 in 3D (Section 5.3.2) is used as boundary conditions for the modeling approach 6 (Section 5.2.2) with an additional void region in the middle of the linkage arm. The resulting linkage arm can not have a larger volume than the initial design. The resulting objective values are; $F_{inn} = 8.23$Pa and $F_{out} = 13.38$Pa. The volume of the best design is; $Vol = 1.069 \cdot 10^{-3} m^3$.

In Table 9 we can see the resulting objective values for all iterations.

Algorithm (objectives)	F_{inn} [Pa]	F_{out} [Pa]	Volume [m^3]
Initial design (Modeling approach 5)	10.52	15.29	$1.086 \cdot 10^{-3}$
NEWUOA (F_{inn},$n = 10$)	7.44	11.69	$1.053 \cdot 10^{-3}$
NEWUOA (F_{out},$n = 10$)	9.03	11.80	$1.098 \cdot 10^{-3}$
NEWUOA (F_{out},$n = 12$)	7.96	11.45	$1.095 \cdot 10^{-3}$
NEWUOA (F_{inn},$n = 12$)	8.72	13.02	$1.054 \cdot 10^{-3}$
Initial design (Modeling approach 6)	37.93	36.21	$1.087 \cdot 10^{-3}$
NEWUOA (F_{inn})	7.83	13.15	$1.034 \cdot 10^{-3}$
NEWUOA (F_{out})	9.68	12.81	$1.059 \cdot 10^{-3}$
MultiOb (F_{inn},F_{out},$Volume$)	5.30	9.30	$1.551 \cdot 10^{-3}$
MultiOb (F_{inn},F_{out})	8.80	12.94	$1.087 \cdot 10^{-3}$
MultiOb (hole,F_{inn},F_{out})	8.23	13.38	$1.069 \cdot 10^{-3}$

Table 9: Comparing results from NEWUOA and MultiOb in 3D with help from topology optimization

8 Conclusions and Discussion

We have used two different approaches in order to find a good structure of a 3D linkage arm. We had some success and some difficulties when modeling the problem. The quality of the software we used was good, in general. However, a lack of some necessary tools and essential information resulted in not so accurate results.

In the topology optimization part, we were unable to model the forces on the holes in the linkage arm in a satisfying manner. Instead of trying to minimize the maximum stress, as was the original idea, we minimized the compliance, since our chosen objective function is not implemented in TO++. It is our belief, however, that the software TO++ could be improved to fit closer to our problem, but that is not a trivial project. TO++ is not a user-friendly software and we were faced with many difficulties when using it. The 2D version is much more developed that the 3D one, and much more work needs to be done before being able to use the current 3D version in a satisfying manner. The author of TO++ was however very helpful, responding to the numerous questions we asked during our work on the project.

Despite some difficulties using TO++, we got some ideas on how we could use the primitive results in order to improve the shape of the linkage arm. This was done in the shape optimization part, where we allowed a hole to be present in the middle of the linkage arm. We got a lower maximum stress by using a linkage arm with a hole, than without a hole for the same amount of material.

The shape optimization part was not without problems either, although we believe that the results we got are much closer to the original objective. We were able to use more accurate boundary conditions, and a correct objective function with the NEWUOA and MultiOb programs. The sensitivity of the objective function is a problem which could perhaps be solved by using more advanced meshing. Doing so would only solve part of the problem, as the problem lacks existence of a solution in general.

We feel that we have got a good experience by working on this project. The main disadvantages were that the we had very limited knowledge of the perhaps necessary requirements the project insisted on, and that the project was not very well defined.

8.1 Future work - improvements

In this thesis, we have made the first few steps in trying to find an optimal shape of a linkage arm, so that its maximum stress is minimized. We have got some primitive results, but we believe that a lot can be done in order to improve the results. Below we give some ideas on how to proceed.

1. One of our biggest problems was to be able to accurately model the loads on the linkage arm. We were more successful in the shape optimization part than in the topology optimization part, but a more rigorous way to model loads should be established.
2. The construction of an appropriate mesh is a theory on its own. It seemed to be especially difficult to find a good mesh around detailed sections of the linkage arm. This should be investigated further.
3. Changing various parameters in the optimization algorithms is an endless source for experiments.
4. It would be interesting to evaluate some error estimates for the Finite Element Method to assess thee accuracy of our solution.
5. It is necessary to examine whether there exists a solution to the problem and, if so, whether it is unique.
6. An extended mechanical mathematical model is needed to cope with the sensitivity.

References

[1] N. ANDRÉASSON, A. EVGRAFOV, AND M. PATRIKSSON, *An Introduction to Continuous Optimization: Foundations and Fundamental Algorithms*, Studentlitteratur, 2005.

[2] M. BENDSØE AND O. SIGMUND, *Topology Optimization: Theory, Methods and Applications*, Springer Verlag, Berlin Heidelberg, 2003.

[3] FRAUNHOFER ITWM, *MultiOb Reference Manual 1.0 (c)*, Dec 2005.

[4] J. M. GERE, *Mechanics of materials*, Brooks/Cole Publishing Company, 5 ed., 2001.

[5] T. HANNE, *On the convergence of multiobjective evolutionary algorithms*, European Journal of Operational Research, 117 (1999), pp. 553–564.

[6] T. HANNE, *Global multiobjective optimization using evolutionary algorithms*, Journal of Heuristics, 6 (2000), pp. 349–360.

[7] A. KLARBRING AND P. W. CHRISTENSEN, *A brief introduction to structural optimization*. Institute of Technology, Department of Mechanical Engineering. SE-581 83 Linköping, Sweden.

[8] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer Verlag, 1999.

[9] N. S. OTTOSEN AND H. PETERSSON, *Introduction to the finite element method*, Redwood Books, Trowbridge, Wiltshire, 1992.

[10] P. PEDERSEN AND C. LAURSEN, *Design for minimum stress concentration by finite elements and linear programming*, Journal of Structural Mechanics, 10 (1983), pp. 375–391.

[11] J. PETERSSON, *Optimization of structures in unilateral contact*, PhD thesis, Department of Mechanical Engineering, Linköping Institute of Technology, Linköping, Sweden, 1995.

[12] M. J. D. POWELL, *The NEWUOA software for unconstrained optimization without derivatives*, Report DAMTP 2004/NA08, Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, Cambridge University, Cambridge, UK, 2004.

[13] A. SAMUELSSON AND N.-E. WIBERG, *Finite Element Method: Basics*, Studentlitteratur, Lund, Sweden, 1998.

[14] A. A. SEIREG AND J. RODRIGUEZ, *Optimizing the Shape of Mechanical Elements and Structures*, Marcel Dekker Inc, New York, 1997.

[15] O. SIGMUND, *A 99 line topology optimization code written in Matlab*, Structural and Multidisciplinary Optimization, 21 (2001), pp. 120–127. `http://www.topopt.dtu.dk`.

[16] G. SJODIN, 2002. A Lab report showing various aspect of the current linkage arm. Atlas Copco Rock Drills AB.

[17] A.-B. STRÖMBERG, *Multi-criteria optimization and decision support in product and process design based on simulation*. Project plan. Fraunhofer-Chalmers Center for Industrial Mathematics, Chalmers Science Park, SE-412 88 Göteborg, Sweden, September 2005.

[18] K. SVANBERG, *The method of moving asymptotes - a new method for structural optimization*, International Journal for Numerical Methods in Engineering, 24 (1987), pp. 359–373.

[19] K. WEDDFELT, *Utmattningsprovning av länkarmar i konstantamplitud*, tech. rep., Atlas Copco Rock Drills AB, 701 91 Örebro, Sweden, 2002. (in Swedish).