Consistent formulation in isogeometric topology optimization for structural applications

Paul Salden
MSc Thesis
Consistent formulation in isogeometric topology optimization for structural applications
by
Paul Salden
to obtain the degree of Master of Science at the Delft University of Technology

Student number: 1216333

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft
Delft University of Technology
Faculty of Aerospace Engineering
Department of Aerospace Structures and Materials

GRADUATION COMMITTEE

Dated: 26th September 2016

Committee chairman: 

Dr. S.R. Turteltaub

Committee members:

Dr. M.M. Abdalla

Dr.ir. M. Langelaar
This report covers my thesis research, the final step in obtaining an MSc degree in Aerospace Engineering. I have worked with isogeometric analysis and topology optimization. They are two mathematically involved subjects with many prospects for industry application. It has been an honest pleasure studying them. The topics have also lead to many fruitful discussions with my faculty roommates, for which I am grateful.

I would like to thank my family and friends for their continuous, seemingly unconditional support over all this time. They have helped me get through in many ways. My promise of better days is one that continues to stand.

A special word of thanks is owed to my supervisor Dr. S.R. Turteltaub. Despite a true test of patience, his guidance and enthusiasm for the work have never seemed to diminish. It should go without saying that in absence of his support, I would not have been able to complete this project.

Thanks also go to Dr. M.M. Abdalla and Dr.ir. M. Langelaar, who have agreed to be on my assessment committee.

The Python code I used for generating the results in this report may be obtained from my GitHub account.¹

Paul Salden
Delft, September 2016

¹https://github.com/PaulSalden/
# Contents

1 **Introduction**        ......................................................... 1  
  1.1 Background and motivation ............................................... 1  
    1.1.1 Heuristic nature of SIMP implementations ...................... 1  
    1.1.2 Isogeometric framework ........................................... 2  
  1.2 Literature review ........................................................ 2  
    1.2.1 Topology optimization with the homogenization method .......... 2  
    1.2.2 Solid Isotropic Material with Penalization .................... 2  
    1.2.3 Mesh sensitivity and checkerboard formation .................... 3  
    1.2.4 Checkerboard prevention .......................................... 3  
    1.2.5 Isogeometric analysis ............................................. 4  
    1.2.6 Discussion of literature ......................................... 5  
  1.3 Research goals and report layout ...................................... 5  

2 **Mathematical descriptions in isogeometric topology optimization** 7  
  2.1 Formulation of the minimum compliance problem .................... 7  
    2.1.1 Balance equations .................................................. 7  
    2.1.2 Minimum compliance ................................................. 9  
  2.2 Isogeometric discretization ............................................ 10  
    2.2.1 B-splines .............................................................. 10  
    2.2.2 Important characteristics of B-splines ......................... 10  
    2.2.3 Knot insertion ....................................................... 11  
    2.2.4 NURBS ................................................................. 13  
    2.2.5 B-spline and NURBS surfaces ..................................... 13  
    2.2.6 Isogeometric formulation of optimization problem ............. 15  
  2.3 Numerical integration through Bézier extraction .................. 17  
    2.3.1 Numerical integration through Gaussian quadrature ............ 17  
    2.3.2 Bézier curves and Bernstein polynomials ....................... 18  
    2.3.3 Bézier extraction ................................................. 18  
    2.3.4 Element structure and integration ............................... 22  
  2.4 Descent method ............................................................ 24  
    2.4.1 Steepest descent ................................................... 25  
    2.4.2 Use of a modified gradient ...................................... 25  

3 **Variations on SIMP implementation** .................................. 27  
  3.1 Classical concepts within SIMP ....................................... 27  
    3.1.1 Solid Isotropic Material with Penalization .................... 27  
    3.1.2 Piecewise constant densities ..................................... 28  
    3.1.3 Discrete gradient derivation ..................................... 29  
    3.1.4 The checkerboard phenomenon .................................... 30  
    3.1.5 Density filters ..................................................... 30  
  3.2 Classical implementation of SIMP ..................................... 32  
    3.2.1 The continuation method and programming technique .......... 32  
    3.2.2 Implementation steps .............................................. 33  
  3.3 Consistent density representation ................................... 34  
    3.3.1 Continuous density ................................................ 34  
    3.3.2 Consistent gradient derivation ................................. 35  
    3.3.3 Implementation of consistent densities ......................... 39
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Separate design and analysis mesh refinements</td>
<td>39</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Defining the design and analysis mesh</td>
<td>39</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Modified displacement calculation</td>
<td>40</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Modified gradient calculation</td>
<td>41</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Integral evaluation over design elements</td>
<td>42</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Implementation of separate mesh refinements</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>Comparative analysis of SIMP variations</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Effect of continuous densities on checkerboard formation</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Checkerboard prevention through split meshes</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Additional optimization examples</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions and recommendations</td>
<td>69</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusions about isogeometric SIMP variations</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>Recommendations for future research</td>
<td>70</td>
</tr>
<tr>
<td>A</td>
<td>Validation of IGA code</td>
<td>71</td>
</tr>
<tr>
<td>A.1</td>
<td>Airy stress function</td>
<td>71</td>
</tr>
<tr>
<td>A.2</td>
<td>Validation of implementation</td>
<td>72</td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
<td>79</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>81</td>
</tr>
</tbody>
</table>
Introduction

Topology optimization is an automated design approach for structural applications. It has been adopted by the industry, including the aerospace sector. Resulting designs can be made suitable for traditional manufacturing techniques. One chooses an objective to optimize for, such as minimum compliance or maximum heat conduction. Given one or more constraints, the optimal distribution of material within a domain is computed.

A popular topology optimization method is Solid Isotropic Material with Penalization (SIMP). It is traditionally implemented using piecewise constant density values, specified per element. A typical issue that occurs, the formation of checkerboards, is circumvented through the application of filters. The resulting designs display structural members with relatively high thicknesses. It is often argued how this property is beneficial from a manufacturing viewpoint. However, given the rise of modern manufacturing techniques such as additive manufacturing, it is interesting to study optimization procedures that preserve the detail levels of unfiltered designs. For that reason, this report presents a continuous, consistent density representation and the use of separated design and analysis meshes.

Background and motivation are elaborated on in section 1.1. A review and discussion of relevant literature are presented in section 1.2. Section 1.3 lists specific goals for the project, as well as describing the layout of this report.

1.1. Background and motivation

This section gives an overview of relevant background and outlines the motivation for carrying out the project.

1.1.1. Heuristic nature of SIMP implementations

Structural optimization has been a topic of interest for a long time. Several branches that have been studied over the years are sizing optimization, shape optimization and topology optimization. The latter has gained a lot of attention since the publication of a milestone paper on optimization with homogenization techniques by Bendsøe and Kikuchi [13]. This research led to the introduction of the SIMP method a year later, by Bendsøe [12]. Several alternative topology optimization methods have been studied since, but SIMP appears to be the chosen method for many applications. Two major aspects of typical SIMP implementations are of questionable mathematical soundness.

The first issue relates to density fields, which are approximated as being piecewise constant. Each element in a mesh has one density value. SIMP typically produces so-called checkerboards, topologies with solid elements connected through corner hinges. Given the piecewise nature of such topologies, it is intuitive to suspect a connection with the element-wise density representation.

A second source of doubt concerns filtering, the method applied to prevent checkerboard formation. It adds an extra step to the optimization process, and does not integrate with the mathematical setup of the problem. The motivation for the approach are its pragmatic results.

Mathematically consistent alternatives may be found for both issues. It is possible to apply SIMP without making the assumption of element-wise constant densities. Filtering can be replaced by the
use of separate meshes for “design” and “analysis”, a method that has been studied in the past. This report provides implementations for both approaches, and presents an analysis of their effectiveness.

1.1.2. Isogeometric framework
Methods such as SIMP depend on Finite Element Analysis (FEA) for structural analysis. The FEA method has existed for a long time, with Lagrange polynomials employed as basis functions. In what is called an isoparametric setting, these polynomials are used to approximate both solution field and geometry. In Computer Aided Design (CAD), geometry is constructed using Non-Uniform Rational B-Splines (NURBS) rather than polynomials. Hughes et al. [31] have developed an analysis method centered around NURBS, called Isogeometric Analysis (IGA).

One may implement IGA for topology optimization. This removes the need for re-meshing after design. It also allows for more accurate approximation of arc-like geometries. A successful implementation would be of interest to the topology optimization research community, which has been dedicated to FEA for the past decades. For that reason, IGA is applied exclusively throughout the report.

1.2. Literature review
This section reviews the literature relevant to this report. It is not meant to be exhaustive. Instead, focus is put on material that is directly used in later chapters. Optimization using the homogenization method is treated first, followed by SIMP. Special attention is paid to problems with mesh sensitivity and checkerboards, as well as prevention schemes. IGA is presented next. The section finishes with a discussion of the covered material, highlighting the relation to the rest of this report.

1.2.1. Topology optimization with the homogenization method
In shape optimization, boundaries of a domain are moved such that an objective is minimized. The procedure typically does not allow for the creation of holes, or the change of topology. In an attempt to improve on this, Bendsoe and Kikuchi [13] have considered a distribution of rectangular holes. They choose both the size and the orientation of the holes as design variables. Macroscopic material properties are then derived through the homogenization method for composite materials. Results in this milestone paper reveal a polynomial-like relation between material stiffness and hole density.

1.2.2. Solid Isotropic Material with Penalization
A year after reporting on the homogenization method, Bendsøe [12] has replaced the hole distribution by a virtual density variable. This variable is kept constant per element in a mesh, and determines a material stiffness distribution over the domain. Inspired by the work with the homogenization method, another polynomial relation is applied. For a linear dependence of stiffness on density, one obtains a so-called “gray” material solution. Such topologies have been found earlier by Rossow and Taylor [41]. Higher polynomial orders represent so-called penalization. The solution is pushed towards a 0–1 design, or a design with exclusively void and fully solid material. This penalization scheme forms the basis of the SIMP method.

SIMP is widely applied in academics and industry, partly due to its relatively simple implementation. This simplicity has been demonstrated in various publications. A freely available web application has been provided and described by Tcherniak and Sigmund [61]. Well-known is the SIMP implementation in 99 lines of MATLAB code by Sigmund [50]. Andreassen et al. [4] have “updated” this code to 88 lines. Several aspects of the method are refined and a more efficient implementation is provided. SIMP has been applied to many research fields. Amongst many others, applications are reported on for compliant mechanisms (Sigmund [49]), geometrically nonlinear structures (Buhl et al. [20]) and reliability-based topology optimization (Kharmanda et al. [35]).

A descent method updates density variables for every iteration in SIMP. One simple descent method is the optimality criterion method, which has been discussed in detail by Zhou and Rozvany [63] and Zhou and Rozvany [64]. While this approach is simple to implement, its major drawback is the limitation to a single global constraint. Numerical methods may be used to alleviate this restriction. Popular choices are Convex Linearization (CONLIN) (Fleury [26]) and its derivative Method of Moving Asymptotes (MMA) (Svanberg [60]). Sigmund and Maute [53] have reported MMA to currently be a common choice for industry applications.

As a relatively simple proof of concept, topology optimization is often implemented for a minimum
compliance problem. Such a problem becomes concave when penalization is used. Without measures in place, the optimizer may converge on “local optima”. Petersson and Sigmund [37] have introduced the “continuation method”, which reportedly prevents this phenomenon. An unpenalized stiffness is applied when optimization starts. Penalization is then gradually introduced over subsequent iterations. A slightly modified version of this approach has been described by Allaire and Francfort [2] and Allaire and Kohn [3]. They define a fixed set of penalization powers, moving on to the next value every time optimization converges.

Rozvany [42] has argued that validation of topology optimization results is mostly based on visual comparison. In his opinion, it would be better to have a quantified standard. However, such a standard has not been adopted thus far.

1.2.3. Mesh sensitivity and checkerboard formation
The objective of a 0–1 design commonly follows from manufacturing considerations. One way of obtaining such a design is restricting the solution space to these two values only. However, Stolpe and Bendsøe [58] have shown that discrete density treatments are impractical. SIMP works with a “relaxed” design space instead. Intermediate values are allowed, but penalized. As a result, the optimal topology is free from intermediate densities to the maximum possible extent. Sigmund and Petersson [54] have discussed how this leads to a problem of mesh dependency: there is always a finer-scale topology that is “more optimal”. Mathematically speaking, a solution is required to be independent of the discretization used. The optimization problem is therefore termed “ill-posed”.

A major issue with methods like SIMP is the formation of topologies resembling checkerboards. Such topologies represent a specific type of mesh dependency. They have been studied by Díaz and Sigmund [23]. “Checkerboards” contain solid material regions of single element width, connected by “corner hinges”. The analysis method does not account for the infeasibility of such hinges, which is shown to be the reason for the “checkerboarding” phenomenon. Jog and Haber [33] have published a theoretical framework for the issue. They limit their treatment to 2D cases, but numerical experiments by Beckers [10] have shown checkerboard formation for 3D problems as well. Petersson and Sigmund [37] have stated that methods which resolve mesh dependency, typically dampen checkerboards.

1.2.4. Checkerboard prevention
Formation of checkerboards is commonly prevented through the application of “filters”. Inspired by image processing, a sensitivity filter has been introduced by Sigmund [48]. An extra step is added to the optimization process, in which sensitivities for all density variables are modified. Each sensitivity is adjusted as a weighted average of its surrounding values. The weight is inversely proportional to the distance between elements. This filtering approach has been criticized as being heuristic and lacking proof of the existence of a solution. Sigmund and Petersson [54] have discussed how this leads to a problem of mesh dependency: there is always a finer-scale topology that is “more optimal”. Mathematically speaking, a solution is required to be independent of the discretization used. The optimization problem is therefore termed “ill-posed”.

A major issue with methods like SIMP is the formation of topologies resembling checkerboards. Such topologies represent a specific type of mesh dependency. They have been studied by Díaz and Sigmund [23]. “Checkerboards” contain solid material regions of single element width, connected by “corner hinges”. The analysis method does not account for the infeasibility of such hinges, which is shown to be the reason for the “checkerboarding” phenomenon. Jog and Haber [33] have published a theoretical framework for the issue. They limit their treatment to 2D cases, but numerical experiments by Beckers [10] have shown checkerboard formation for 3D problems as well. Petersson and Sigmund [37] have stated that methods which resolve mesh dependency, typically dampen checkerboards.
However, the methods are different in nature. Moreover, filters introduce a minimum length scale and hence mesh insensitivity. Obtaining a similar effect for “patching” is possible, but requires the design mesh to be kept constant as the analysis mesh is refined. The “resolution” of the design then no longer improves.

1.2.5. Isogeometric analysis

Numerical analysis in solid mechanics traditionally involves FEA. The analysis method is therefore also applied for topology optimization. However, CAD is used for design. FEA relies on Lagrange polynomials as basis functions, while CAD depends on NURBS. The difference is inconvenient, since it causes the need for a re-meshing step when going from design to analysis or optimization. In an attempt at unification, Hughes et al. [31] have introduced IGA, an isoparametric analysis method using NURBS basis functions.

The IGA method has been thoroughly explained in the book by Cottrell et al. [22]. Many applications have been reported on as well, for instance in the fields of turbulence (Akkerman et al. [1]), fluid-structure interaction (Bazilevs et al. [8]), incompressibility (Auricchio et al. [5]) and shells (Benson et al. [15]).

Piegl and Tiller [38] and Rogers [40] have treated NURBS in detail. To construct NURBS, B-spline basis functions are first defined through the Cox-De Boor recursion formula. NURBS are then obtained by combining weighted rationals of these basis functions. Geometry is defined through three variables: knots, control points and weights.

NURBS have several distinguishable geometric properties. They may be used to exactly fit circular paths. The continuity of NURBS equals the polynomial order, but is lowered by knot repetition. Control points are not interpolated by NURBS, which has important implications for implementation of IGA. Refinement possibilities have been discussed by Cottrell et al. [21]. Besides h-refinement and p-refinement, which are similar to their counterparts in FEA, NURBS allow for a combination termed k-refinement.

Patches of NURBS surfaces and volumes are formed through tensor products of basis functions. The resulting grid-like layout is therefore said to have a “tensor product structure”. A consequence is that refinement always affects the whole patch. Local refinement has been achieved by Bazilevs et al. [9] and Scott et al. [46] through the use of T-splines. These T-splines have been discussed by Sederberg et al. [47] and generally do not have a tensor product structure. A drawback is that T-splines make implementation of IGA more complex.

Obtaining an element structure for NURBS is complicated by the lack of control point interpolation. Borden et al. [16] have introduced Bézier extraction as a solution. The same method has been applied to T-splines by Scott et al. [45]. Through a set of extraction operators, every knot span is related to one and the same Bézier element. Such an element is made up of Bernstein polynomials, as has been described in more detail by Farouki [25]. Bézier curves interpolate their exterior control points, and geometry depends on quantities local to the element only. The use of this property brings IGA closer to FEA. Differences between their implementations are then confined to the so-called “shape routine”. Numerical integration for FEA has been discussed in detail by Hughes [30]. For IGA with Bézier extraction, the procedure is analogous. However, the prescribed amounts of quadrature points are inefficient in this case. This has been explained by Hughes et al. [32]. The discussion has been extended by Auricchio et al. [6]. When integrating knot spans, use may be made of the continuity of NURBS. In many cases, this leads to “under-integration” being sufficient. Schillinger et al. [44] have combined the approach with Bézier extraction.

Hassani et al. [29] have provided an implementation of IGA within SIMP, using element-wise constant densities. The results for elementary minimum compliance problems visually match those obtained with FEA elsewhere. Applying IGA reportedly results in a checkerboard-free topology, even for coarse meshes. However, a “noise-cancellation technique” is mentioned despite these claims. This technique is not further elaborated on.

Density has been represented through a fully separate B-spline mesh by Qian [39]. Disconnecting the design mesh from the analysis mesh leads to checkerboard-free topologies. It is similar to “patching” methods. This approach is presented as being a “B-spline filter”. IGA is explored as analysis method alongside FEA. The density values are chosen as element-wise constant, as nodal variables or at quadrature points. Minimum length scale control is not achieved, leading to mesh sensitivity. Those topics are recommended for potential future research, with robust formulations (Wang et al. [62]) as a
suggested option. Given the lack of control point interpolation for NURBS, the combination of IGA and “nodal” densities is especially interesting. However, a detailed treatment is absent.

1.2.6. Discussion of literature
A method for topology optimization that is popular in academics and industry is SIMP. It is thoroughly described in the literature, which proves valuable when constructing an implementation. As descent method, authors choose between the optimality criterion method and numerical methods like MMA. These latter methods are a necessity when considering multiple constraints. The continuation method is a straightforward tool for ensuring global convergence. Visual comparison is a standard method for validating topology optimization results.

Common issues with SIMP are mesh sensitivity and the formation of checkerboards. Mesh sensitivity relates to the tendency of ever smaller microstructure to form as the mesh is refined. By checkerboards, element-wide regions of solid material are meant. These regions are connected through corner hinges. The literature presented does not mention a connection between checkerboards and the element-wise constant density representation being used. However, visual inspection suggests a relation. Additional research is warranted.

A method typically applied for preventing checkerboards is filtering of either sensitivities or densities. These quantities are averaged over a neighborhood of a preset radius. This introduces a minimum length scale for the final topology, achieving mesh insensitivity as well. Despite its wide application, the approach has heuristic components. A more mathematically sound alternative may be found in “patching”. One collects various analysis elements in design patches. Unlike filters, this does not trivially result in minimum length scale control. A design remains sensitive to the mesh used. It is nonetheless interesting to further study the effect of a non-heuristic method on the final topology.

IGA blends CAD and FEA. The concept may be extended to topology optimization. A choice must be made between NURBS and T-spline basis functions. This is a choice between ease of implementation and local refinement possibilities, respectively. Regardless of the basis chosen, Bézier extraction introduces an element structure. Changes in implementation with respect to FEA are confined to the so-called “shape routine”. Large parts of existing FEA codes are reusable, which is both efficient and attractive from the perspective of industry adoption. For an optimal implementation, the continuity of NURBS may be used to derive efficient quadrature rules.

Some articles mention the combination of IGA and topology optimization. Thorough treatments remain unavailable, especially if one considers a continuous density approximation. A dedicated study on the matter would be a valuable contribution to the research field.

1.3. Research goals and report layout
Section 1.2 identifies several opportunities for research, based on current scientific literature. The relation between checkerboard formation and element-wise constant densities has not yet been studied. Checkerboards are typically countered using filtering, a method that is often considered heuristic. SIMP itself is implemented using FEA, while IGA allows for re-using a CAD mesh. The aim of the research presented in this report may be stated as follows:

Implement SIMP using a consistent density representation, and evaluate its effect on checkerboard formation. Study the replacement of filters with a separated design and analysis mesh. Use isogeometric analysis with Bézier extraction throughout.

Density representation is consistent when it utilizes the same basis functions as the geometry and displacement approximations. Such a representation is continuous and replaces the element-wise constant densities traditionally implemented for SIMP.

Providing a simple SIMP implementation means it should be no more complex than is necessary to study the desired modifications. Restricting the optimization routine to relatively simple physical problems might limit its applicability. The implementation should nonetheless serve as a proof of concept for a consistent density representation and separated design and analysis meshes. Choices for implementation aspects are listed in Table 1.1.

Comparison of the optimization results is based on visual inspection and compliance values. This is in line with current practice, as reported on in the literature. Although numbers of elements and checkerboard prevention parameters are easily varied, the intention is not to carry out elaborate pa-
Table 1.1: Choices affecting an implementation of the optimization routine. The aim is to make it no more complex than necessary to study the intended features. Aspects relating to the physical problem are presented first, after which implementation details are listed. Note that although these are kept simple, the continuation method is retained as it is straightforward to implement.

<table>
<thead>
<tr>
<th>aspect</th>
<th>choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical dimensions</td>
<td>2</td>
</tr>
<tr>
<td>objective functional</td>
<td>minimum compliance</td>
</tr>
<tr>
<td>constraints</td>
<td>single volume constraint</td>
</tr>
<tr>
<td>loading</td>
<td>static, no body force</td>
</tr>
<tr>
<td>strain assumption</td>
<td>infinitesimal strain theory</td>
</tr>
<tr>
<td>constitutive relation</td>
<td>homogeneous, isotropic, plane stress</td>
</tr>
<tr>
<td>boundary conditions</td>
<td>traction, homogeneous displacement</td>
</tr>
<tr>
<td>basis functions</td>
<td>NURBS</td>
</tr>
<tr>
<td>integration</td>
<td>FEA quadrature rules</td>
</tr>
<tr>
<td>descend method</td>
<td>simple gradient method</td>
</tr>
<tr>
<td>local optima handling</td>
<td>continuation method</td>
</tr>
</tbody>
</table>

rameter studies. Even without those, insight gained from the results should make a contribution to the academic literature.

The layout of this report is as follows. Chapter 2 presents various mathematical and physical preliminaries necessary to implement SIMP. Chapter 3 discusses a classical SIMP implementation, followed by the variations suggested in this section. A thorough comparison of MBB beam results is provided in chapter 4, along with additional examples of optimized topologies. Chapter 5 lists conclusions and recommendations. The IGA code used to generate the optimization results is validated in appendix A.
Mathematical descriptions in isogeometric topology optimization

Implementation of SIMP within an isogeometric framework requires various mathematical and physical concepts. These concepts are presented in this chapter. The notation used is clarified as it appears. Additional references are provided when material is not treated in section 1.2. As outlined in Table 1.1, the presentation is limited to two dimensional minimum compliance problems.

Section 2.1 gives the mathematical formulation of a minimum compliance problem. The problem is discretized for isogeometric analysis in section 2.2. Section 2.3 explains numerical integration and Bézier extraction. The descent method is presented in section 2.4.

2.1. Formulation of the minimum compliance problem

Mathematical expressions for the minimum compliance problem are derived in this section. It is split up between a presentation of balance equations and a treatment of compliance.

2.1.1. Balance equations

Multiple quantities are typically defined in solid mechanics. Start out with strain tensor $\epsilon$. In line with Table 1.1, it is related to displacement $u$ through infinitesimal strain theory:

$$\epsilon = \nabla u + \nabla u^T$$

Where $\nabla$ is the Cartesian gradient operator, the gradient taken in physical space. Strain is defined as the symmetric part of the displacement gradient. It may be related to stress $\sigma$ through the generalized Hooke's law as:

$$\sigma = C \epsilon$$

With $C$ a fourth-order stiffness tensor. In the homogeneous, isotropic plane stress case, it has both major and minor symmetry. Making use of the major symmetry, (2.1) and (2.2) may be combined as:

$$\sigma = C \nabla u$$

On a boundary, stress is related to traction $t$ by normal vector $n$:

$$t = \sigma n$$

Consider a domain $\Omega \in \mathbb{R}^d$, where $d$ is the amount of physical dimensions. An example is shown in Figure 2.1. It might undergo a body force $b$, plus be subjected to a displacement $u$ on $\Gamma_D$ and a traction $t$ on $\Gamma_N$. However, by Table 1.1:

$$\bar{b} = 0$$

$$\bar{u} = 0$$
The balance of linear momentum on $\Omega$ for static cases is then written as:

$$\begin{align*}
\text{div} \boldsymbol{\sigma} &= 0 \quad \text{on } \Omega \\
\boldsymbol{u} &= 0 \quad \text{on } \Gamma_D \\
t &= \bar{t} \quad \text{on } \Gamma_N
\end{align*}$$

(2.7)  (2.8)  (2.9)

In addition, balance of angular momentum prescribes that:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T$$

(2.10)

It is not possible to find an analytical solution to (2.7)–(2.9). Instead, one typically attempts to satisfy the balance equations in a weak sense. Define an arbitrary "weight function" $\boldsymbol{w}$ that is similar to $\boldsymbol{u}$. Multiplying (2.7) by $\boldsymbol{w}$ and integrating over $\Omega$:

$$\int_{\Omega} \text{div} \boldsymbol{\sigma} \cdot \boldsymbol{w} \, d\Omega = 0$$

(2.11)

Consider expanding a gradient of a symmetric second order tensor $\mathbf{A}$ acting on a vector $\mathbf{b}$:

$$\text{div} (\mathbf{A} \mathbf{b}) = \text{div} \mathbf{A} \cdot \mathbf{b} + \mathbf{A} \cdot \nabla \mathbf{b}$$

(2.12)

This may be used to rewrite (2.11):

$$\int_{\Omega} \mathbf{\sigma} \cdot \nabla \boldsymbol{w} \, d\Omega = \int_{\Gamma} \mathbf{\sigma} \boldsymbol{w} \cdot \mathbf{n} \, d\Gamma = \int_{\Gamma} \mathbf{t} \cdot \boldsymbol{w} \, d\Gamma$$

(2.13)

Apply the divergence theorem, (2.10) and (2.4):

$$\int_{\Omega} \mathbf{\sigma} \cdot \nabla \boldsymbol{w} \, d\Omega = \int_{\Gamma} \mathbf{\sigma} \boldsymbol{w} \cdot \mathbf{n} \, d\Gamma = \int_{\Gamma} \sigma \mathbf{w} \cdot \mathbf{n} \, d\Gamma$$

(2.14)

Since $\boldsymbol{w}|_{\Gamma_D} = 0$ and because of (2.9), this may be rewritten as:

$$\int_{\Omega} \mathbf{\sigma} \cdot \nabla \boldsymbol{w} \, d\Omega = \int_{\Gamma_N} \mathbf{t} \cdot \boldsymbol{w} \, d\Gamma$$

(2.15)

Or with (2.3):

$$\int_{\Omega} \mathbf{c} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{w} \, d\Omega = \int_{\Gamma_N} \mathbf{t} \cdot \boldsymbol{w} \, d\Gamma$$

(2.16)
2.1. Formulation of the minimum compliance problem

Which is called the *weak formulation* of the balance of linear momentum. Condition (2.9) is incorporated in a “natural” sense. It is therefore called a *natural boundary condition*, as opposed to *essential boundary condition* (2.8). Weak form (2.16) is the starting point for Galerkin’s method applied in section 2.2.

2.1.2. Minimum compliance

Several objective functionals have been considered for topology optimization. An objective functional is “the quantity to optimize for”. By Table 1.1, the treatment below is limited to minimum compliance. It is the most common choice in introductory texts. Compliance $D$ is defined as:

$$ D = \int_{\Gamma_N} \bar{t} \cdot u \, d\Gamma - \int_{\Gamma_D} t \cdot \bar{u} \, d\Gamma \quad (2.17) $$

Because of the minus-sign, minimizing $D$ implies minimizing the first term, while maximizing the second. Both terms contain an expression of work. In words, minimizing compliance may be stated as:

*Require prescribed tractions to do as little work as possible, and at the same time require maximum work for accomplishing prescribed displacements.*

As mentioned in Table 1.1, assumption $\bar{u} = 0$ holds. This simplifies (2.17) to:

$$ J = D|_{\bar{u}=0} = \int_{\Gamma} t \cdot u \, d\Gamma \quad (2.18) $$

An alternative form is often presented. Applying (2.4) and (2.10):

$$ J = \int_{\Gamma} \sigma n \cdot u \, d\Gamma = \int_{\Gamma} \sigma^T u \cdot n \, d\Gamma = \int_{\Gamma} \sigma u \cdot n \, d\Gamma \quad (2.19) $$

Then using the divergence theorem and (2.12):

$$ J = \int_{\Omega} \text{div}(\sigma u) \, d\Omega = \int_{\Omega} \text{div} \sigma \cdot u \, d\Omega + \int_{\Gamma} \sigma \cdot n u \, d\Omega \quad (2.20) $$

As the static case is being studied, (2.7) reduces this to:

$$ J = \int_{\Omega} \sigma \cdot \nabla u \, d\Omega \quad (2.21) $$

In which (2.1) may be substituted. Due to the symmetry of $\sigma$ as per (2.10) and properties of the dot product, the result becomes:

$$ J = \int_{\Omega} \sigma \cdot \epsilon \, d\Omega \quad (2.22) $$

Compliance is equal to twice the total strain energy in the domain, $E = \frac{1}{2} \int_{\Omega} \sigma \cdot \epsilon \, d\Omega$.

The actual minimization problem may be formulated as:

$$ \text{find } \rho^* \text{ such that } \forall \rho, \ J(\rho^*) \leq J(\rho) \quad (2.23) $$

$$ \text{and } \int_{\Omega} \rho^* \, d\Omega \leq V \quad (2.24) $$

And either definition (2.17) or (2.22) may be used for $J$. Density variable $\rho$ and volume constraint (2.24) are treated in chapter 3. It must be understood that during minimization, weak form (2.16) of the balance equations needs to be satisfied.
2.2. Isogeometric discretization

For isogeometric analysis, physical fields are represented in the same way geometry is. This holds for displacements, but may also apply to the density field (section 3.3). Use is made of NURBS basis functions. This section presents NURBS as generalizations of B-splines. Important characteristics are treated, as well as the extension to surfaces. An actual isogeometric formulation of the optimization problem is derived at the end.

2.2.1. B-splines

Prior to deriving NURBS, B-splines must be considered. Both are curves that map a coordinate $\xi$ in parameter space to a location $x$ in physical space. In this case, physical space is made up of domain $\Omega$ in Figure 2.1. A B-spline is defined as a linear combination:

$$ B(\xi) = \sum_{a=1}^{n} N_{a,p}(\xi) P_a $$

Functions $N_{a,p}$ are B-spline basis functions. They are defined through the Cox-De Boor recursion formula:

$$
\begin{align*}
N_{a,p}(\xi) &= \frac{\xi - \xi_a}{\xi_{a+p} - \xi_a}N_{a,p-1}(\xi) + \frac{\xi_{a+p+1} - \xi}{\xi_{a+1} - \xi_a}N_{a+1,p-1}(\xi) \\
N_{a,0}(\xi) &= \begin{cases} 1 & \xi \in [\xi_a, \xi_{a+1}) \\ 0 & \xi \notin [\xi_a, \xi_{a+1}) \end{cases}
\end{align*}
$$

The B-spline and its basis depend on several parameters:

- Polynomial order $p$ has implications similar to that of Lagrange polynomials in FEA. It determines the complexity of geometry that may be approximated using the basis, as well as the continuity of basis functions.

- The shape of the basis is determined by a set of non-decreasing knots $\xi_k$. These knots define parameter space. They are collected in knot vector $\Xi$. The subspace between two adjacent knots is called a knot span. A knot vector is said to be open if its first and last knot have multiplicity $p + 1$. They are then repeated $p + 1$ times:

$$
\Xi_{\text{open}} = [\xi_0, \ldots, \xi_0, \xi_1, \ldots, \xi_{n-1}, \xi_{n-1}, \ldots, \xi_N]_{p+1}
$$

Only open knot vectors are used in this report. A knot vector is termed uniform if there is equal spacing between all successive knots. Knot vectors may be scaled and translated without effect on the basis.

- Let a knot vector contain $l$ knots. The amount $n$ of corresponding basis functions is then:

$$
n = l - p - 1
$$

- The physical layout of a B-spline is determined by control points $P_a$. They represent positions in physical space and therefore have $d$ components. Although control points function similarly to nodes in FEA, they are generally not interpolated by the NURBS curve. Subsection 2.2.2 elaborates on the matter.

An example of B-spline basis functions is shown in Figure 2.2.

2.2.2. Important characteristics of B-splines

B-splines have several important characteristics, with direct implications for IGA and topology optimization. They are as follows:
2.2. Isogeometric discretization

Figure 2.2: B-spline basis functions for polynomial order 2 and knot vector [0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5]. Note that this knot vector is open but not uniform. The effects of the latter property may be observed in the figure.

1. B-spline basis functions span an amount of knot spans equal to \( p + 1 \). This is named the support of the basis functions. It may be observed from Figure 2.2. Note that some functions appear to have a smaller span. This is because the knot at \( \xi = 4 \) is repeated (has multiplicity 2), essentially creating a zero-width knot span.

2. The sum of all basis function values at one parametric location always equals one:

\[
\sum_{a=1}^{n} n_{a,p}(\xi) \equiv 1
\]  

(2.30)

This is commonly termed the partition of unity property. Together with the span property of basis functions, this has an important consequence for B-splines representing scalar fields. Such B-splines may be seen as having one-dimensional control points. Consider a case where the field values must be non-zero and constant over one knot span. This is only possible if the field is non-zero over at least \( 2p + 1 \) knot spans. See Figure 2.3. The effective span may be reduced through knot multiplicity, similarly to Figure 2.2.

3. The continuity of B-splines is equal to their polynomial order. Each time a knot is repeated, the continuity of the curve is locally reduced by 1 order. Note that the continuity of the basis remains the same. Figure 2.2 shows the effect. Open knot vectors define a B-spline that is “discontinuous” at its ends.

4. A B-spline cannot follow an exactly circular path (while a NURBS can). This is displayed in Figure 2.4.

5. B-splines do not interpolate their control points. In other words, control points are generally not positioned on the curve. This may be observed in Figure 2.4. It is caused by multiple basis functions being non-zero at knot span borders, the dotted lines in Figure 2.2.

These properties have consequences for the derivations further on in this chapter, and for the implementations described in chapter 3.

2.2.3. Knot insertion

B-splines can be refined through several methods. Knot insertion has similarities to h-refinement in FEA, while order elevation is the counterpart to p-refinement. A combination of the two, called k-refinement, is unique to IGA. Only knot insertion is used in this report.

Consider inserting a knot \( \tilde{\xi} \in [\xi_k, \xi_{k+1}] \) into \( \Xi \). The new knot vector \( \Xi' \) becomes:

\[
\Xi' = [\xi_0, ..., \xi_k, \tilde{\xi}, \xi_{k+1}, ..., \xi_n]
\]  

(2.31)
Mathematical descriptions in isogeometric topology optimization

Figure 2.3: 1D B-splines for order 2 knot vector [0, 0, 0, 1, 2, 3, 4, 5, 5, 5]. A knot span wide value of 1 (solid line) is achieved through control points [0, 0, 1, 1, 1, 0, 0], which leads to the curve being non-zero in \( 2p + 1 = 5 \) knot spans. The narrowest curve (dashed line) is achieved through control points [0, 0, 1, 0, 0, 0], but reaches a lower value.

Figure 2.4: A B-spline (solid) can only approximate circular geometry, while a NURBS (dashed) can exactly fit it. Control points are shown as well. Neither curve interpolates its interior control point.
The goal is then to construct a new set of $n+1$ control points $\tilde{P}_a$, such that the B-spline does not change. This is done as follows:

$$\tilde{P}_a = \begin{cases} P_1 & a = 1 \\ \alpha_a P_a + (1 - \alpha_a) P_{a-1} & 1 < a < n+1 \\ P_n & a = n+1 \end{cases}$$

(2.32)

Coefficients $\alpha_a$ are chosen as:

$$\alpha_a = \begin{cases} 1 & 1 \leq a \leq k-p \\ \frac{\xi - \xi_a}{\xi_{a+p} - \xi_a} & k-p+1 \leq a \leq k \\ 0 & a \geq k+1 \end{cases}$$

(2.33)

Again, the basis changes, but the curve does not.

Subsection 2.2.2 describes how the continuity of a B-spline is locally reduced by increased knot multiplicity. The procedure above may be used to achieve this.

**2.2.4. NURBS**

Figure 2.4 shows how a B-spline cannot exactly fit a circular path, while a NURBS can. The definition of NURBS is similar to that of B-splines:

$$N(\xi) = \sum_{a=1}^{n} R_{a,p}(\xi) P_a$$

(2.34)

Now with basis functions $R_{a,p}$ according to:

$$R_{a,p}(\xi) = \frac{N_{a,p}(\xi) w_a}{W(\xi)} = \frac{N_{a,p}(\xi) w_a}{\sum_{a=1}^{n} N_{a,p}(\xi) w_a}$$

(2.35)

NURBS basis functions are rationals of B-spline basis functions (2.26) and weights $w_a$. Function $W(\xi)$ is called the weight function. All B-spline characteristics outlined in subsection 2.2.2 are shared with NURBS. The obvious exception is that NURBS can represent circular geometry.

Knot insertion is not applied to NURBS directly. For every NURBS $N(\xi)$, one may define a B-spline $B^p(\xi)$ with an equal polynomial order and knot vector. Let every B-spline control point $P_a$ be formed from the corresponding NURBS control point and weight through:

$$P_a^p = \left( \begin{array} {c} w_a P_{a,1} \\ w_a P_{a,2} \\ \vdots \\ w_a P_{a,d} \end{array} \right)$$

(2.36)

Where $P_{a,i}$ is a physical component $i$ of $P_a$. B-spline $B^p(\xi)$ is known as the projective B-spline belonging to NURBS $N(\xi)$. Its control points exist in projective space, which has one dimension more than the NURBS physical space. A geometrical representation may be constructed as well [22]. The procedure for knot insertion with NURBS is then as follows:

1. Transform the NURBS into its projective B-spline using (2.36).
2. Apply knot insertion to the projective B-spline using (2.32) and (2.33).
3. Transform the refined projective B-spline back into a NURBS using the “inverse” of (2.36).

**2.2.5. B-spline and NURBS surfaces**

Definitions for B-splines and NURBS are easily extended to surfaces (and solids). Take two sets of B-spline basis functions: $N_{a,p}(\xi)$ with $a = 1, 2, \ldots, n$, and $M_{b,q}(\eta)$ with $b = 1, 2, \ldots, m$. A B-spline surface is then defined as:

$$S^B(\xi, \eta) = \sum_{a=1}^{n} \sum_{b=1}^{m} N_{a,p}(\xi) M_{b,q}(\eta) P_{a,b}$$

(2.37)
With control points $P_{a,b}$ that may be gathered in a “net” $\hat{P}$:

$$\hat{P} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,m} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n,1} & P_{n,2} & \cdots & P_{n,m} \end{bmatrix}$$ (2.38)

Like the curve, the surface does not interpolate these control points. Collect basis functions $N_{a,p}(\xi)$ and $M_{b,q}(\eta)$ in vectors $\hat{N}$ and $\hat{M}$ respectively:

$$\begin{align*} \hat{N} &= \begin{pmatrix} N_{1,p}(\xi) \\ N_{2,p}(\xi) \\ \vdots \\ N_{n,p}(\xi) \end{pmatrix} \\ \hat{M} &= \begin{pmatrix} M_{1,p}(\xi) \\ M_{2,p}(\xi) \\ \vdots \\ M_{m,p}(\xi) \end{pmatrix} \end{align*}$$ (2.39)

Equation (2.37) may then equivalently be written as:

$$S^B(\xi, \eta) = \left( \hat{N} \otimes \hat{M} \right) \cdot \hat{P}$$ (2.40)

A 2D basis is obtained through a tensor product of 1D basis functions. B-spline surfaces are said to have a tensor product structure. It allows for efficient computer implementation, since 1D basis function values can be computed separately. However, the structure also makes refinement propagate in the mesh. See chapter 4 for examples of the tensor product structure.

B-spline surfaces may also be defined as a linear combination of 2D basis functions. Let “global index” $A$ be related to indexes $a$ and $b$ as:

$$A = n(b - 1) + a$$ (2.41)

B-spline surface (2.42) are then equivalently defined as:

$$S^B(\xi, \eta) = \sum_{A=1}^{nm} N_{A,p,q}(\xi, \eta) P_A$$ (2.42)

With basis functions:

$$N_{A,p,q}(\xi, \eta) = N_{a,p}(\xi) M_{b,q}(\eta)$$ (2.43)

$$P_A = P_{a,b}$$ (2.44)

A NURBS surface is defined analogously:

$$S^B(\xi, \eta) = \sum_{A=1}^{nm} R_{A,p,q}(\xi, \eta) P_A$$ (2.45)

With basis functions:

$$R_{A,p,q}(\xi, \eta) = \frac{N_{a,p}(\xi) M_{b,q}(\eta) w_A}{\sum_{A=1}^{nm} N_{a,p}(\xi) M_{b,q}(\eta) w_A}$$ (2.46)

NURBS surfaces have a tensor product structure as well, again affecting implementation and refinement.

Further extension to solids is straightforward. Generally speaking, a B-spline/NURBS surface or solid is considered a “patch”. Different patches may be connected to create complex geometry. In that case, connection equations are necessary for analysis and optimization. In this report, only single-patch NURBS surfaces are used.
2.2.6. Isogeometric formulation of optimization problem

To facilitate numerical implementation, one must obtain discrete variants of (2.16) and (2.22). NURBS have been described as defining geometry in the previous subsections. They map a parameter \( \xi \) to a physical location \( x \). Other fields may be approximated using NURBS as well. As the equations that follow are to be used for computer implementation, matrix notation is adopted.

Discrete approximations for displacements and the trial function must be introduced first. For ease of notation, define the parameter pair \( \xi = (\xi, \eta) \). Let geometry be specified through NURBS surface \( N(\xi) \):

\[
N(\xi) = \sum_{A=1}^{nm} R_{A,p,q}(\xi) P_A
\]  

(2.47)

Through \( N \), fields \( u \) and \( w \) are a function of the parametric coordinate as well:

\[
u(x) = u(N(\xi))
\]  

(2.48)

\[
w(x) = w(N(\xi))
\]  

(2.49)

They may be approximated using basis functions \( R_{A,p,q} \):

\[
u(N(\xi)) \approx \sum_{A=1}^{nm} R_{A,p,q}(\xi) u_A
\]  

(2.50)

\[
w(N(\xi)) \approx \sum_{A=1}^{nm} R_{A,p,q}(\xi) v_A
\]  

(2.51)

Where vector-valued coefficients \( u_A \) and \( v_A \) are named control variables, analogous to control points \( P_A \). The dependence of \( u \) and \( w \) on physical position \( x \) becomes implicit. A coarse NURBS control point grid is often sufficient for describing geometry. However, accurate approximation of displacement requires many control points. For that reason, NURBS are typically refined before doing analysis or optimization. The procedure is explained in section 2.3.

Gradients of these fields must be computed as well. At this point, it is convenient to introduce Voigt notation. The stress and strain matrices are represented as vectors of their unique, non-zero components:

\[
sigma = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{pmatrix}
\]  

(2.52)

\[
\varepsilon = \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix}
\]  

(2.53)

Similar notation is used for gradients:

\[
\nabla u = \begin{pmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_2}{\partial x} \\ \frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x} \end{pmatrix}
\]  

(2.54)

Which is discretized through substituting (2.50):

\[
\nabla u \approx \sum_{A=1}^{nm} B_A u_A
\]  

(2.55)

With matrix \( B_A \) defined as:

\[
B_A = \begin{bmatrix} \frac{\partial R_A}{\partial x} & 0 \\ 0 & \frac{\partial R_A}{\partial y} \end{bmatrix}
\]  

(2.56)
Note that for readability, the index pair \((p, q)\) is suppressed for \(R\). The derivatives of the basis functions are not computed directly. They are instead evaluated using Bézier extraction, as explained in section 2.3. A similar derivation holds for \(\nabla w\).

When Voigt notation is used, stiffness tensor \(C\) is usually replaced by a matrix \(D\). For homogeneous, isotropic, plane stress cases it is written as:

\[
C \rightarrow D = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}
\]  

(2.57)

In which Young’s modulus \(E\) and Poisson’s ratio \(\nu\) are used.

One may now substitute discrete quantities in the weak formulation of the balance equations. Rewriting (2.16):

\[
\int_D \left( \sum_{A=1}^{nm} B_A u_A \right) \cdot \left( \sum_{\beta=1}^{nm} B_\beta v_\beta \right) \, d\Omega = \int_{\Gamma_N} \bar{t} \cdot \left( \sum_{\beta=1}^{nm} R_\beta v_\beta \right) \, d\Gamma
\]  

(2.58)

The order of integration and summation may be changed in this case. Matrix \(D\) is symmetric while coefficients \(u_A\) and \(v_\beta\) are constant. Equation (2.58) can thus be rearranged as:

\[
\sum_{\beta=1}^{nm} v_\beta^T \left[ \int_A^{nm} B_\beta^T D B_A \, d\Omega \right] u_A = \sum_{\beta=1}^{nm} v_\beta^T \left[ \int_{\Gamma_N} R_\beta \, d\Gamma \right] B = 1, 2, \ldots, nm
\]  

(2.59)

Trial function \(w\) is chosen arbitrarily. Coefficients \(v_\beta^T\) are therefore arbitrary as well. For that reason, (2.59) can be satisfied in only one way. Individual terms in both summations over index \(B\) must be set equal. A system of equations results:

\[
\sum_{A=1}^{nm} \left( \int_A^{nm} B_\beta^T D B_A \, d\Omega \right) u_A = \int_{\Gamma_N} R_\beta \, d\Gamma \quad B = 1, 2, \ldots, nm
\]  

(2.60)

This system may conveniently be rewritten as a matrix equation. Introduce indexes \(P\) and \(Q\):

\[
P = d(A - 1) + i \quad (2.61)
\]

\[
Q = d(B - 1) + j \quad (2.62)
\]

The system becomes:

\[
K \vec{u} = \vec{f}
\]  

(2.63)

**Stiffness matrix** \(K\) of size \((dnm \times dnm)\) is symmetric and sparse. It has components:

\[
K_{PQ} = e_i^T \int_A^{nm} B_\beta^T D B_A \, d\Omega e_j
\]  

(2.64)

With \(e_i\) and \(e_j\) being the physical unit vectors. **Displacement vector** \(\vec{u}\) has size \((dnm \times 1)\) and components:

\[
\vec{u}_P = e_i^T u_A
\]  

(2.65)

And **force vector** \(\vec{f}\) has size \((dnm \times 1)\) and components:

\[
\vec{f}_P = \int_{\Gamma_N} R_\beta \, d\Gamma
\]  

(2.66)

After applying homogeneous displacement boundary condition (2.8), system (2.63) is typically solved numerically. Section 2.3 discusses the procedure.

A discrete expression of compliance is found in a largely similar way. Take (2.22) and apply (2.3) and (2.1):

\[
J = \int_{\Omega} \nabla \vec{u} \cdot \nabla \vec{u} \, d\Omega
\]  

(2.67)
2.3. Numerical integration through Bézier extraction

Use is again made of the symmetry of $\sigma$. Inserting discrete quantities yields:

$$ J = \int_{\Omega} D \left( \sum_{a=1}^{nm} B_A u_A \right) \cdot \left( \sum_{B=1}^{nm} B_B u_B \right) \, d\Omega $$

(2.68)

Or:

$$ J = \sum_{A=1}^{nm} \sum_{B=1}^{nm} u_A^T \left( \int_{\Omega} B_A^T D B_B \, d\Omega \right) u_B $$

(2.69)

Which may be written in terms of the stiffness matrix and displacement vectors defined above as:

$$ J = \hat{u}^T K \hat{u} $$

(2.70)

This can be reworked to a form more similar to (2.18). Applying (2.63) and commutativity of the dot product:

$$ J = \hat{u}^T f = f^T \hat{u} $$

(2.71)

These equations are used to compute the compliances of topologies in chapter 3.

2.3. Numerical integration through Bézier extraction

This section first presents a numerical integration scheme. NURBS are shown to complicate its use. A solution is found in Bézier curves and Bézier extraction, which are outlined next. Bézier extraction creates an element structure. It is shown how integration may be carried out per element.

2.3.1. Numerical integration through Gaussian quadrature

Numerical integration is carried out using Gaussian quadrature in traditional FEA. Function $f(x)$ is integrated numerically as follows:

$$ \int_{-1}^{1} f(x) \, dx = \sum_{i=1}^{\hat{n}} f(\tilde{x}_i) \tilde{w}_i $$

(2.72)

In which $\tilde{x}_i$ and $\tilde{w}_i$ are called integration points and integration weights respectively. These weights should not be confused with weights used to define NURBS basis functions in section 2.2.4. Sufficient integration points are necessary to achieve accurate integration. Given $\hat{n}$ integration points, polynomial order $p$ of $f(x)$ may at most be:

$$ p = 2\hat{n} - 1 $$

(2.73)

Specific procedures exist to determine exact locations and weights of integration points [30]. The concept is easily extended to multiple dimensions. Let a second parametric direction require $\hat{m}$ integration points $\tilde{y}_j$ with weights $\tilde{w}_j$. One may write:

$$ \int_{-1}^{1} \int_{-1}^{1} f(x, y) \, dx \, dy = \sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} f(\tilde{x}_i, \tilde{y}_j) \tilde{w}_i \tilde{w}_j $$

(2.74)

A grid of integration points is formed. Weights at every location are calculated as the product of the corresponding unidirectional weights.

Gaussian quadrature must be applied to NURBS, since computation of stiffness matrix (2.64) and force vector (2.66) involves integrating basis functions. FEA shape functions are the same for every element. As a result, their integration point values have to be computed only once. Subsection 2.2.2 explains how NURBS basis functions vary per element, depending on the knot vector. That property holds in both parametric directions in the case of surfaces. Integration over the domain or boundary is therefore more computationally involved. The remainder of this sections presents Bézier extraction as a solution.
2.3.2. Bézier curves and Bernstein polynomials

Prerequisite to the concept of Bézier extraction is the Bézier curve. It is in fact a B-spline with an open knot vector and no interior knots:

$$\Xi_{\text{Bezier}} = [0, \ldots, 0, 1, \ldots, 1]$$

(2.75)

The Bézier curve may formally be defined as:

$$\mathbf{Z}(\tilde{\xi}) = \sum_{a=1}^{n} B_{a,p}(\tilde{\xi}) \mathbf{P}_a$$

(2.76)

With Bernstein polynomials as basis functions:

$$\begin{align*}
B_{a,p}(\tilde{\xi}) &= (1 - \tilde{\xi})B_{a,p-1}(\tilde{\xi}) + \tilde{\xi}B_{a-1,p-1}(\tilde{\xi}) \\
B_{a,0}(\tilde{\xi}) &= 1
\end{align*}$$

(2.77, 2.78)

Bernstein polynomials of order \(p = 3\) are shown in Figure 2.5a. Note how at the endpoints, only one basis function is nonzero and reaches value 1. Bézier curves interpolate their exterior control points. Bernstein polynomials are all supported on the same interval \([0, 1]\), facilitating efficient implementation of quadrature as treated in subsection 2.3.4.

Derivatives of Bernstein polynomials may easily be computed. They are defined in terms of lower order Bernstein polynomials:

$$\frac{dB_{a,p}}{d\xi} = p (B_{a-1,p-1}(\tilde{\xi}) - B_{a,p-1}(\tilde{\xi}))$$

(2.79)

2.3.3. Bézier extraction

It is possible to represent every B-spline knot span as a Bézier curve. The procedure is called Bézier extraction. Every interior knot must be given multiplicity \(p\) through knot insertion. Take Figure 2.2 as example. Order 2 basis functions for knot vector \(\Xi = [0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5]\) are shown. Knots that have to be inserted are \(\Xi_{\text{insert}} = [1, 2, 3]\).
The procedure for inserting a single knot is the starting point. It is outlined in subsection 2.2.3. Let (2.32) be used to insert a knot \( \bar{\xi}_i \). One may consider the equivalent matrix multiplication:

\[
\begin{bmatrix}
\bar{P}_1 \\
\bar{P}_2 \\
\vdots \\
\bar{P}_{n+1}
\end{bmatrix} = C_i^T \begin{bmatrix}
P_1 \\
P_2 \\
\vdots \\
P_{n}
\end{bmatrix}
\] (2.80)

\[
C_i = \begin{bmatrix}
\alpha_1 & 1 - \alpha_2 & 0 & \cdots & 0 \\
0 & \alpha_2 & 1 - \alpha_3 & 0 & \cdots & 0 \\
0 & 0 & \alpha_3 & 1 - \alpha_4 & 0 & \cdots & 0 \\
0 & \cdots & 0 & \alpha_n & 1 - \alpha_{n+1}
\end{bmatrix}
\] (2.81)

Where it must be understood that \( n \) is the amount of control points before inserting \( \bar{\xi}_i \). Repeated insertion then requires repeated application of (2.80) with (2.81).

Let Bézier extraction now require insertion of \( m \) knots. Gather the set of \( n \) original B-spline control points \( P_a \) in a \((n \times 2)\) matrix \( \hat{P} \):

\[
\hat{P} = \begin{bmatrix}
P_1 \\
P_2 \\
\vdots \\
P_n
\end{bmatrix}
\] (2.82)

A set of \( n + m \) Bézier control points \( P^b_a \) is eventually constructed. Let those be collected in an \(((n + m) \times 2)\) matrix \( \hat{P}^b \):

\[
\hat{P}^b = \begin{bmatrix}
P_1^b \\
P_2^b \\
\vdots \\
P_{n+m}^b
\end{bmatrix}
\] (2.83)

Set (2.83) and (2.82) are then related through:

\[
\hat{P}^b = C^T \hat{P}
\] (2.84)

\[
C^T = C_m^T C_{m-1}^T \cdots C_1^T
\] (2.85)

Matrix \( C \) is named the extraction operator. The procedure above is not the most efficient way of obtaining \( C \) [16]. However, section 3.2 shows how Bézier extraction is only applied in a pre-optimization step. Equations (2.81) and (2.85) are therefore appropriate for this report.

For every knot span, extraction operators relate B-spline basis functions to Bernstein polynomials as well. Gather the set of \( n \) B-spline basis functions \( N_{a,p}(\xi) \) in a \((n \times 1)\) vector \( \hat{N} \):

\[
\hat{N} = \begin{bmatrix}
N_{1,p}(\xi) \\
N_{2,p}(\xi) \\
\vdots \\
N_{n,p}(\xi)
\end{bmatrix}
\] (2.86)

If \( t \) knot spans \( s \) are present, collect \( t \) sets of \( p + 1 \) Bernstein polynomials \( B_{a,p}(\xi) \) in \(((n + m) \times 1)\) vector \( \hat{B} \):

\[
\hat{B} = \begin{bmatrix}
B_{1,p}(\xi) \\
B_{p+1,p}(\xi) \\
\vdots \\
B_{1,p}(\xi) \\
B_{p+1,p}(\xi)
\end{bmatrix}^T \quad s = 1 \\
\vdots \\
\end{bmatrix}^T \quad s = t
\] (2.87)

Definition (2.76) may be extended to create a chain of Bézier curve segments that is identical to the B-spline. The following relation holds:

\[
B(\xi) = Z(\xi) = (\hat{P}^b)^T \hat{B} = (C^T \hat{P})^T \hat{B} = \hat{P}^T C \hat{B}
\] (2.88)
But $B(\xi) = \hat{P}^T \hat{N}$, and so:

$$\hat{N} = C \hat{B}$$  \hspace{1cm} (2.89)

As an example, consider the knot span shown in Figure 2.5b. Its basis functions can be constructed from the Bernstein polynomials in Figure 2.5a.

Extraction operators may be applied to NURBS as well. Gather a set of $n$ NURBS basis functions $R_{\alpha \beta}(\xi)$ in a $(n \times 1)$ vector $\hat{R}$, analogously to (2.86):

$$\hat{R} = \begin{pmatrix} R_{1,\beta}(\xi) \\ R_{2,\beta}(\xi) \\ \vdots \\ R_{n,\beta}(\xi) \end{pmatrix}$$  \hspace{1cm} (2.90)

Now define both a vector $\hat{w}$ and a diagonal matrix $\hat{W}$ containing $n$ NURBS weights $w_{\alpha}$:

$$\hat{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$  \hspace{1cm} (2.91)

$$\hat{W} = \begin{bmatrix} 0 & \cdots & 0 \\ w_1 & \ddots & \vdots \\ \vdots & \ddots & 0 \end{bmatrix}$$  \hspace{1cm} (2.92)

Definition (2.35) may then equivalently be written as:

$$\hat{R} = \frac{1}{\hat{w}^T \hat{N}} \hat{W} \hat{N} = \frac{1}{\hat{w}^T CB} \hat{W} \hat{C} \hat{B}$$  \hspace{1cm} (2.93)

An expression suitable for computer implementation.

A similar procedure is followed for NURBS surface basis functions. Let $\hat{M}$ be a set of B-spline basis functions in a second parametric direction $\eta$. Two distinct sets of Bernstein polynomials are extracted from $\hat{N}$ and $\hat{M}$:

$$\hat{N} = C^\xi B^\xi$$  \hspace{1cm} (2.94)

$$\hat{M} = C^\eta B^\eta$$  \hspace{1cm} (2.95)

As elaborated on in subsection 2.2.5, 2D basis functions are created from the tensor product of 1D basis functions. These functions must be gathered in a $((n \times m) \times 1)$ vector $\hat{N}^S$. The following convention may be adopted:

$$\hat{N}^S = \begin{pmatrix} (\hat{N} \otimes \hat{M})_{1,1} \\ \vdots \\ (\hat{N} \otimes \hat{M})_{n,1} \\ (\hat{N} \otimes \hat{M})_{1,2} \\ \vdots \\ (\hat{N} \otimes \hat{M})_{n,m} \end{pmatrix}$$  \hspace{1cm} (2.96)

Definition (2.46) may then be rewritten as:

$$\hat{R}^S = \frac{1}{\hat{w}^T \hat{N}^S} \hat{W} \hat{N}^S$$  \hspace{1cm} (2.97)

The combination of equations (2.94)–(2.97) enables computer implementation of NURBS surfaces. Basis functions are extracted in both parametric directions separately.

Stiffness matrix components (2.64) require derivatives of $\hat{R}^S$ with respect to both physical directions. They are computed from derivatives with respect to parametric directions through the chain rule:

$$\frac{\partial \hat{R}^S}{\partial x} = \frac{\partial \hat{R}^S}{\partial \xi} \frac{\partial \xi}{\partial x} = \frac{\partial \hat{R}^S}{\partial \xi} \left( \frac{\partial x}{\partial \xi} \right)^{-1}$$  \hspace{1cm} (2.98)
Since NURBS are used to describe geometry, one has:

\[ \frac{\partial x}{\partial \xi} = \hat{P} \frac{\partial \hat{R}^S}{\partial \xi} \]  

(2.99)

With:

\[ \frac{\partial \hat{R}^S}{\partial x} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \]  

(2.100)

\[ \frac{\partial \hat{R}^S}{\partial \xi} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \]  

(2.101)

\[ \frac{\partial x}{\partial \xi} = \begin{bmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{bmatrix} \]  

(2.102)

Due to the tensor product structure, derivatives \( \frac{\partial \hat{R}^S}{\partial \xi} \) may be computed in a relatively straightforward way. In view of (2.94)–(2.96), derivatives of the 2D B-spline basis may be obtained through:

\[ \frac{d\hat{N}}{d\xi} = C^\xi \frac{dB^\xi}{d\xi} \]  

(2.103)

\[ \frac{d\hat{M}}{d\eta} = C^\eta \frac{dB^\eta}{d\eta} \]  

(2.104)

And:

\[ \frac{\partial \hat{N}^S}{\partial \xi} = \begin{pmatrix} \frac{d\hat{N} \otimes \hat{M}}{d\xi} \\ \vdots \\ \frac{d\hat{N} \otimes \hat{M}}{d\xi} \end{pmatrix} \]  

\[ \frac{\partial \hat{N}^S}{\partial \eta} = \begin{pmatrix} \frac{d\hat{N} \otimes \hat{M}}{d\eta} \\ \vdots \\ \frac{d\hat{N} \otimes \hat{M}}{d\eta} \end{pmatrix} \]  

(2.105)

Application of the quotient rule then yields:

\[ \frac{\partial \hat{R}^S}{\partial \xi} = \hat{W} \frac{\partial \hat{N}^S}{\partial \xi} \hat{w}^x \hat{N}^S - \hat{N} \hat{w}^x \hat{N}^S \frac{\partial \hat{N}^S}{\partial \xi} \]  

\[ = \hat{W} \frac{\hat{N}^S}{\hat{w}^x \hat{N}^S} \frac{\partial \hat{N}^S}{\partial \xi} \]  

(2.106)

And analogously:

\[ \frac{\partial \hat{R}^S}{\partial \eta} = \frac{\hat{W}}{\hat{w}^y \hat{N}^S} \left( 1 - \frac{\hat{N}^S}{\hat{w}^y \hat{N}^S} \hat{w}^y \right) \frac{\partial \hat{N}^S}{\partial \eta} \]  

(2.107)

Derivatives of a NURBS surface basis function are then implemented using (2.94)–(2.96) and (2.98)–(2.107).

The procedures above are not the only way to apply Bézier extraction. The method presented is in line with Algorithm 3 [16]. As an alternative, one may aim to express all equations in terms of Bézier basis functions, control points and even weights. B-spline/NURBS counterparts would no longer appear. One then defines Bézier elements that are close to traditional elements in FEA. As is shown in chapter 3, this is not required for the implementations described in this report. Not even when desiring to confine changes to the “shape function routine”, as outlined in chapter 1. The approach would lead to unnecessary overhead and is not further discussed or adopted here.
2.3.4. Element structure and integration

Bézier extraction operators relate NURBS knot spans to sets of Bernstein polynomials. An element structure may be created, allowing for integration. The method is applied for NURBS surfaces in this report.

Consider a knot span \( c \) in \( \xi \) direction. An element extraction operator \( C^{e,el}_\xi \) is obtained from components of the global operator \( C_{ij} \) as:

\[
C^{e,el}_\xi = \begin{bmatrix}
C^{el}_\xi,s & \cdots & C^{el}_\xi,s \\
\vdots & \ddots & \vdots \\
C^{el}_\xi,e & \cdots & C^{el}_\xi,e
\end{bmatrix}
\] (2.108)

With:

\[
s = (c - 1)p + 1 \quad (2.109)
\]
\[
e = cp + 1 \quad (2.110)
\]

NURBS knot spans are then considered elements. Supported B-spline basis functions in \( \xi \) direction may be obtained as:

\[
\hat{N}^{el}_e = C^{e,el}_\xi \hat{B}^\xi
\] (2.111)

An analogous approach holds for the \( \eta \) direction. Note that regardless of the element, basis functions are always related to one and the same set of Bernstein polynomials. Localization is “captured” in the element extraction operator \( C^e \). The notion of element extraction operators may be used to localize the equations in subsection 2.3.3. Those are then used to obtain NURBS basis functions and derivatives on every element.

Considering Bernstein polynomial definition (2.77)–(2.78), a new set of parametric coordinates \( \tilde{\xi} = (\tilde{\xi}, \tilde{\eta}) \) is defined. They exist in what is called the parent space. Bernstein polynomials are defined over a Bézier parent element. Element extraction operators \( C^e \) map this parent element to the different NURBS knot spans in parametric space. As discussed in subsection 2.2.1, the NURBS surface is then mapped to domain \( \Omega \) in physical space. An illustration is provided in Figure 2.6.

Gaussian quadrature as described in subsection 2.3.1 may be applied to Bézier parent elements. Equation (2.74) holds for an interval \([-1, 1]\) in both directions. An extra mapping can be avoided if (2.74) is redefined over parent space:

\[
\int_{\tilde{\xi}} \int_{\tilde{\eta}} f(\tilde{\xi}, \tilde{\eta}) \, d\tilde{\xi} \, d\tilde{\eta} = \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{\tilde{m}} f(\tilde{\xi}_i, \tilde{\eta}_j) \tilde{w}_i \tilde{w}_j
\] (2.112)

With integration points corrected for the change of coordinates:

\[
\tilde{\xi}_i = \frac{\xi_i + 1}{2} \quad (2.113)
\]
\[
\tilde{\eta}_j = \frac{\eta_j + 1}{2} \quad (2.114)
\]

And integration weights adapted for the new interval length:

\[
\tilde{w}_i = \frac{\tilde{w}_i}{2} \quad (2.115)
\]
\[
\tilde{w}_j = \frac{\tilde{w}_j}{2} \quad (2.116)
\]

However, integration must be carried out over the physical domain. This requires a change of domain from the parent space. A Jacobian term must be included in the integrals:

\[
\int^1_{-1} \int^1_{-1} f(\xi, \eta) \, d\xi \, d\eta = \sum_{i=1}^{n} \sum_{j=1}^{m} f(\xi_i, \eta_j) w_i w_j
\] (2.117)
2.3. Numerical integration through Bézier extraction

Figure 2.6: Three spaces relevant to implementation. A Bézier parent element (parent space) is mapped to a NURBS knot span (parameter space). The NURBS surface is in turn mapped to domain $\Omega$ (physical space).

With:

$$\frac{\partial \xi}{\partial \tilde{\xi}} = \begin{bmatrix} \frac{\partial \xi}{\partial \xi} & \frac{\partial \xi}{\partial \eta} \\ \frac{\partial \xi}{\partial \xi} & \frac{\partial \xi}{\partial \eta} \end{bmatrix}$$  \hspace{1cm} (2.118)

Subsection 2.2.1 implies that knot vectors may be translated and scaled without effect on a curve. The same holds for surfaces. Let $S^U$ be a NURBS surface for which knot vectors lie within interval $[0, 1]$. One has:

$$\left(\frac{\partial \xi}{\partial \tilde{\xi}}\right)^S = I$$  \hspace{1cm} (2.119)

With $I$ the $(2 \times 2)$ identity matrix. Knot vectors may be modified such that they fit the description of $S^U$. This simplifies the computation of Jacobian $J^S(\tilde{\xi}, \tilde{\eta})$. In any case, integrals over element $\Omega^{\text{e}\ell}$ are computed as:

$$\int_{\Omega^{\text{e}\ell}} f(\xi, \eta) \, d\Omega = \sum_{i=1}^{m} \sum_{j=1}^{n} f(\xi_i, \eta_j) J^S(\xi_i, \eta_j) \tilde{w}_i \tilde{w}_j$$  \hspace{1cm} (2.120)

Which allows for integration of the basis functions and derivatives defined in subsection 2.3.3. Principally, (2.120) can be used for carrying out boundary integrals as well. However, a simplification is possible. On the boundary, a NURBS surface may be constructed from 1D basis functions. This follows from (2.40) and (2.30). The feature is easier to recognize when only open knot vectors are used, as is the case in this report. Take the boundary at $\eta = 0$. Only one $\eta$-basis function is active and has value 1. As a result:

$$(N \otimes \tilde{M})^{\eta=0} = N \otimes \begin{pmatrix} 1 \\ 0 \\ : \\ 0 \end{pmatrix} = \hat{N}$$  \hspace{1cm} (2.121)
Table 2.1: Numerical arrays used for assembly [30] [22]. The ID array is typically used to incorporate homogeneous displacement boundary conditions.

<table>
<thead>
<tr>
<th>acronym</th>
<th>name</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC</td>
<td>NURBS coordinates</td>
<td>Identifies 1D components of 2D global basis functions.</td>
</tr>
<tr>
<td>IEN</td>
<td>element nodes</td>
<td>Identifies global indexes of basis functions local to elements.</td>
</tr>
<tr>
<td>ID</td>
<td>destination</td>
<td>Identifies system equations for global basis functions.</td>
</tr>
</tbody>
</table>

A similar relation holds for other boundaries and for NURBS. Element boundary integrals may then be evaluated as:

\[
\int_{\Gamma^e} f(\xi) \, d\Gamma = \sum_{i=1}^{n_i} f(\xi_i) \left| \frac{\partial x}{\partial \xi}(\xi_i) \right| \bar{w}_i \\
\int_{\Gamma^e} f(\eta) \, d\Gamma = \sum_{j=1}^{n_j} f(\eta_j) \left| \frac{\partial x}{\partial \eta}(\eta_j) \right| \bar{w}_j
\]

Such that 1D basis functions may be integrated.

Stiffness matrix \( K \) and force vector \( f \) may now be computed element by element, such that system (2.63) is built. One gathers which basis functions are supported on which element. This information is stored in several numerical arrays, as listed in Table 2.1. The following steps are to be taken:

1. Refine the NURBS surface to the desired resolution using the procedures outlined in subsection 2.2.3. Knots are inserted in both parametric directions separately, with multiplicity 1.
2. Obtain an appropriate number of integration points, based on the polynomial orders. Adjust them to the Bézier parent element using (2.113)–(2.116).
3. Compute values of Bernstein basis functions and their derivatives for every integration point. See subsection 2.3.2.
4. Determine element extraction operators for all elements. Also identify which basis functions, control points and weights belong to each element, using arrays IEN and INC.
5. Use the equations in subsection 2.3.3 to compute supported NURBS basis function values and derivatives for every integration point in every element.
6. Determine element contributions to all components \( K_{ij} \) through (2.64) with (2.120) and to components \( f_i \) through (2.66) with (2.122) or (2.123).
7. Sum up all contributions and build system (2.63) using the ID array. Components relating to displacement boundary condition (2.8) are typically dropped.

System (2.63) is then solved numerically. A direct method is used for this report. The process of summing up components and forming the system is called assembly. Element integration, assembly and solution methods are described at length in the literature [30] [22]. Once NURBS basis function and derivative values for the integration points are known, step 6–7 may be carried out repeatedly. They can be used to evaluate other integrals as well (see subsection 3.3.2).

### 2.4. Descent method

Optimization methods in general optimize a design field for an objective functional. Chapter 3 introduces density as the design field. The objective functional is minimum compliance, as explained in section 1.3. Its mathematical problem is treated in section 2.1.

One starts out with an initial choice for the design field. Based on analysis, it is updated iteratively such that the objective functional decreases per iteration. These updates are carried out through a descent method. This section presents the specific descent method used for the current implementation.
2.4. Descent method

In line with section 1.3, it is kept simple. Optimization might take longer as a result, but the simplicity does not interfere with evaluating the proposed SIMP modifications. With appropriate measures for avoiding local optima as discussed in subsection 3.2.1, the optimal topology is eventually produced.

The steepest descent method is outlined first, which is then modified to incorporate resource constraint (2.24).

2.4.1. Steepest descent

The descent methods typically applied for topology optimization in the literature are gradient descent methods [11] [50] [4]. An elementary form of a gradient descent method is the steepest descent method [18] [7]. Given a d-dimensional vector space and a scalar field \( f(x) \), a gradient \( g(x) \) may be defined as:

\[
g(x) = \nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d} \right)
\]  

(2.124)

The gradient vector represents the direction of biggest increment in field \( f \) within the vector space.

Let \( f \) now be the objective functional to be minimized. Components of \( x \) then parametrize the design field. They must be updated by \( \Delta x \) in a "direction" opposite to the gradient \( g \). Magnitude \( |\Delta x| \) is often determined by a chosen step size \( \alpha \). For every optimization iteration \( k \), one may formally write:

\[
x^{k+1} = x^k + \Delta x^k \\
\Delta x^k = -\alpha g^k
\]  

(2.125) (2.126)

An illustration of the gradient and the design field update for \( d = 2 \) is shown in Figure 2.7.

Note that \( \alpha \) equals magnitude \( |\Delta x| \) when gradient \( g \) is normalized. An appropriate choice for \( \alpha \) is necessary. Chosen too low, it will make optimization require an unnecessarily large amount of iterations. When too high, the optimizer might repeatedly overshoot, preventing convergence. More advanced methods, like a line search, require additional analysis. To keep the implementation simple (as outlined in section 1.3), \( \alpha \) is chosen constant. Similar approaches are common in the topology optimization literature [50] [4].

2.4.2. Use of a modified gradient

This subsection considers density specifically. Let the density field be parametrized by \( \rho \). In line with Table 1.1, \( \rho^{k+1} \) is subjected to a single global resource constraint:

\[
V(\rho^{k+1}) \leq \bar{V}
\]  

(2.127)
2. Mathematical descriptions in isogeometric topology optimization

Where the constraint is normalized to the range $[0, 1]$:

$$0 \leq V(\rho^{k+1}) \leq 1$$  \hspace{1cm} (2.128)

As explained in chapter 3, a local constraint is active for every $\rho_i$ as well:

$$0 \leq \rho_i^{k+1} \leq 1 \quad i = 1, 2, \ldots, N$$  \hspace{1cm} (2.129)

The descent method must account for both types of constraints. This is achieved by modifying the gradient in the steepest descent scheme introduced in subsection 2.4.1:

$$\rho_i^{k+1} = \begin{cases} 0 & \rho_i^k + \Delta \rho_i^k < 0 \\ \rho_i^k + \Delta \rho_i^k & 0 \leq \rho_i^k + \Delta \rho_i^k \leq 1 \\ 1 & \rho_i^k + \Delta \rho_i^k > 1 \\ \end{cases} \quad i = 1, 2, \ldots, N$$  \hspace{1cm} (2.130)

$$\Delta \rho_i^k = -\alpha (g_i^k + \lambda^k)$$  \hspace{1cm} (2.131)

Where local constraint (2.129) is actively enforced. Global constraint (2.127) is satisfied through a proper choice of $\lambda^k$. To find it, an inner loop of iterations $l$ is used:

$$\lambda_i^{k,l+1} = \lambda_i^{k,l} + \Delta \lambda$$  \hspace{1cm} (2.132)

$$\Delta \lambda = \begin{cases} \alpha^{-1} \left( V(\rho^{k+1}) - \overline{V} \right) & V(\rho^{k+1}) > \overline{V} \\ 0 & V(\rho^{k+1}) < \overline{V} \end{cases}$$  \hspace{1cm} (2.133)

The loop is terminated once $\Delta \lambda = 0$. Remainder $\Delta \lambda$ is repeatedly determined based on an estimate. If density could be represented by a scalar value $\rho$ with $V = V(\rho)$, $\Delta \lambda$ would be such that global constraint (2.127) is matched exactly. The less accurate this description, the more iterations $l$ are needed to calculate $\lambda^k$. In practice this means that as optimization proceeds, the necessary amount of iterations $l$ increases.

As instructed in subsection 2.4.1, a constant value is used for $\alpha$. That way, the actual change in density scales with the magnitude of the gradient. The rationale behind this is as follows. At an optimum, the gradient vanishes. Since optimization must convergence on this point, the magnitude of density updates has to approach zero as well.

Finding a proper $\alpha$ value is often a matter of experiments. For this report, the following estimate is computed in the first iteration:

$$\alpha = \frac{|\rho^1|}{|g^1|}$$  \hspace{1cm} (2.134)

Gradient $g^1$ is scaled such that it has the same magnitude as density vector $\rho^1$.

Detailed implementation of the descent method described in this subsection depends on the representation of the density variable. The matter is discussed in chapter 3.
Variations on SIMP implementation

This chapter focuses on SIMP and the variations proposed in section 1.3. As said, SIMP is a widely used topology optimization method in academia and industry today. Features that have traditionally been part of the method are discussed in section 3.1. They rely on the concepts described in Chapter 2. Section 3.2 derives the steps required to create a classical SIMP implementation. A mathematically consistent density representation is introduced in section 3.3. Section 3.4 outlines the use of different mesh refinements for design and analysis.

3.1. Classical concepts within SIMP

This section presents various concepts that are classically part of a SIMP implementation. SIMP as a density method and its penalization scheme are introduced first. They are followed by the introduction of element-wise constant densities, with a derivation of the gradient. The checkerboard phenomenon is discussed as well. Prevention is possible through the density filter, which is treated last.

3.1.1. Solid Isotropic Material with Penalization

For any objective functional, topology optimization has traditionally been about “where to put material”. At every location inside a domain, one has exactly two options. There should either be material, or there should be nothing. This is commonly referred to as a 0–1 design. It leads to a discrete design space. Successful numerical implementation with such spaces is only possible in very limited cases [58].

It is desirable to use a continuous design field instead. Such a field may then be incorporated in the continuous equations derived in section 2.1. Several approaches exist for making a continuous design field approximate a 0–1 design. The most common method presently applied is SIMP [11].

SIMP is in fact a material model. It modifies stiffness tensor $\mathbb{C}$ in (2.2) to:

$$\mathbb{C} = \rho^\theta \mathbb{C}_0$$

(3.1)

In which $\mathbb{C}_0$ is the original, unmodified stiffness tensor. Scalar field $\rho \in [0,1]$ is the density field. It is the design field in optimization. For that reason, SIMP is sometimes said to be a density method [53]. Power $\theta$ is the penalization power. A standard value that is used throughout the report is $\theta = 3$. Since $0 \leq \rho \leq 1$, intermediate densities have a penalized effect on stiffness. Combined with a volume constraint, the optimizer may be expected to avoid the use of such densities as much as possible. Penalization can be made inactive by setting $\theta = 1$. One may then for instance model variable thickness sheets [53]. The two scenarios are illustrated in Figure 3.1.

Note: the penalization power is typically indicated with symbol $p$. This report uses $\theta$ to avoid confusion with the polynomial order of NURBS, as described in subsection 2.2.1.

The implementation of (3.1) and specific penalization powers has a direct influence on the nature of the continuous problem. This is treated in subsection 3.1.4. Before substitution within discrete systems (2.63) and (2.71), a discretization for the density variable $\rho$ must be adopted. Options for the latter are discussed in subsection 3.1.2 and subsection 3.3.1.
3. Variations on SIMP implementation

3.1.2. Piecewise constant densities

From a viewpoint of numerical implementation, one must introduce discrete approximations of the density field. This enables the substitution in equations derived in subsection 2.2.6. A consistent approach would be to approximate using NURBS basis functions, as is already done for displacement. However, it is common to specify constant density values for each element \([12][53]\). This approach is referred to as an element-wise constant density approximation or element densities in this report.

Let \(\rho_{e1}\) be an element density. The discrete variant of (3.1) relates to matrix \(D\) and is written per element as:

\[
D_{e1} = (\rho_{e1})^\theta D_0
\]  

(3.2)

Where \(D_{e1}\) is the element \(D\) matrix and \(D_0\) is again the original, full material \(D\) matrix. Element densities \(\rho_{e1}\) allow for an element-wise definition of stiffness matrix components. This is convenient, since subsection 2.3.4 explains how stiffness matrix \(K\) is built element by element. Substitute (3.2) in (2.64):

\[
K_{pq}^{e1} = e_i^T \int_{\Omega^{e1}} B_i^T (\rho_{e1})^\theta D_0 B_q \, d\Omega e_j = (\rho_{e1})^\theta (K_0^{e1})_{pq}
\]  

(3.3)

For which \((K_0^{e1})_{pq}\) is an element stiffness matrix component calculated for “full material” \(\rho = 1\). The optimizer changes element densities after every iteration. Although this means stiffness matrix \(K\) must be re-computed, integrals do not have to be re-evaluated. A multiplication suffices, which allows for a faster implementation. Element stiffness matrices scale with element densities. As can be observed from (2.70), similar implications hold for the determination of compliance. This relative simplicity is a major advantage of element-wise constant densities.

The element-wise constant density approximation is used when calculating total volume as well. Resource constraint (2.24) is then made dimensionless by normalizing this total volume. In a continuous sense, one has:

\[
V = \frac{\int_{\Omega} \rho \, d\Omega}{\int_{\Omega} d\Omega}
\]  

(3.4)

Let all element densities be collected in a vector \(\hat{\rho}_{e1}\). One approximates (3.4) as:

\[
V(\hat{\rho}_{e1}) = \frac{\sum_{e1} \int_{\Omega^{e1}} \rho_{e1} \, d\Omega}{\sum_{e1} \int_{\Omega^{e1}} d\Omega} = \frac{\sum_{e1} \rho_{e1} v_{0e1}^{e1}}{\sum_{e1} v_{0e1}^{e1}}
\]  

(3.5)
3.1. Classical concepts within SIMP

Where $V_{e1} = \int_{\Omega_{e1}} d\Omega$ must be pre-computed for each element. Repeated calculation of volume is also simplified through the use of element densities. This holds for integrations involving $\rho_{e1}$ in general.

3.1.3. Discrete gradient derivation

Optimization routines involve the implementation of a descent method. Section 2.4 explains the choice for modified steepest descent. It requires computation of the gradient. Such a gradient indicates how the objective functional responds to changes in the design field.

A relatively simple approach for obtaining the gradient is typically adopted in the literature \[12\] [50]. Rather than starting from a continuous compliance formulation, the derivation starts with discrete form (2.71). Currently, compliance $J$ is a function of a finite set of element densities $\rho_{e1}$. Gradients $g_{e1}$ are therefore computed as partial derivatives:

$$g_{e1} = \frac{\partial J}{\partial \rho_{e1}} = f^T \frac{\partial \hat{u}}{\partial \rho_{e1}}$$

(3.6)

The relation between $\hat{u}$ and $\rho_{e1}$ is implicit. It is determined through the solution of system (2.63), and partial derivative $\frac{\partial \hat{u}}{\partial \rho_{e1}}$ cannot be computed analytically. Given the potential size of displacement vectors, finite difference approximations and similar are no feasible options either. Instead, one adopts the so-called adjoint method.

Define an arbitrary adjoint vector $\hat{u}^*$. Let its size be equal to that of $\hat{u}$. The aim is to augment discrete compliance formulation (2.71) without changing the value of $J$. System (2.63) may be utilized:

$$\tilde{J} = J = f^T \hat{u} - (\hat{u}^*)^T (K\hat{u} - f)$$

(3.7)

With $\tilde{J}$ the augmented compliance. Note that it equals $J$ regardless of $\hat{u}^*$. The gradient may now be re-formulated as:

$$g_{e1} = \frac{\partial J}{\partial \rho_{e1}} = \frac{\partial \tilde{J}}{\partial \rho_{e1}}$$

$$g_{e1} = f^T \frac{\partial \hat{u}}{\partial \rho_{e1}} - \left( \frac{\partial \hat{u}^*}{\partial \rho_{e1}} \right)^T (K\hat{u} - f) - (\hat{u}^*)^T \left( \frac{\partial K}{\partial \rho_{e1}} \hat{u} + K \frac{\partial \hat{u}}{\partial \rho_{e1}} \right)$$

(3.8)

Where (2.63) has now been used to drop a term. Given the symmetry of $K$, the first remaining term may also be dropped if $\hat{u}^*$ satisfies:

$$K\hat{u}^* = f$$

(3.9)

Which is the same restriction as imposed by (2.63). In conclusion, derivative $\frac{\partial \hat{u}}{\partial \rho_{e1}}$ cancels if one lets the adjoint vector equal the displacement vector:

$$\hat{u}^* = \hat{u}$$

(3.10)

The minimum compliance problem is said to be self-adjoint. A simple expression follows for the gradient:

$$g_{e1} = -\hat{u} \frac{\partial K}{\partial \rho_{e1}} \hat{u} = -\theta (\rho_{e1})^{\theta-1} \hat{u} K_{e1}^{\rho_{e1}} \hat{u}$$

(3.11)

Since $\rho_{e1}$ only influences stiffness matrix components with contributions from its corresponding element. Full material stiffness matrix $K_0$ is defined analogously to (3.3). Given a known displacement vector, expression (3.11) is straightforward to compute. The gradient is always negative. Increasing density leads to a decrease in compliance, which matches intuition. Dependence on $\rho$ is explicit, but disappears when $\theta = 1$. 

3. Variations on SIMP implementation

3.1.4. The checkerboard phenomenon

This report treats minimum compliance topology optimization with a SIMP model. Without penalization, $\theta = 1$, this is a convex problem [59]. “Local optima” do not exist. A (modified) steepest descend method with appropriate step size will always converge on the optimal topology. Such topologies are demonstrated in subsection 4.1. For higher penalization powers $\theta > 1$, the problem becomes concave. A direct consequence is that the “optimum” found will depend on the optimization procedure. Final topology also becomes dependent on the mesh used.

An issue that typically appears is the formation of checkerboards. These topologies consist of single elements filled with material, placed in a checkerboard pattern. The elements are connected through so-called corner hinges, which cannot exist in practice. Checkerboard topology is illustrated in Figure 3.2.

The general understanding is that checkerboards result from the numerical analysis method [23]. FEA over-estimates stiffness of these structural compositions. It does not account for corner hinges being unrealistic. Related properties of IGA are evaluated in subsection 4.1.

Subsection 1.2.6 discusses the relation between checkerboards and the element-wise constant density representation introduced in subsection 3.1.2. Element densities lead to inherently discontinuous topologies. As is obvious from Figure 3.2, checkerboards represent such a topology. Section 3.3 proposes implementation of a continuous density approximation. By stepping away from element densities, checkerboards in their classical form are excluded from the solution space.

3.1.5. Density filters

Formation of checkerboards must be prevented. Several methods have been studied for this purpose. An approach commonly applied is density filtering [19]. It is often considered heuristic, but introduces a notion of “minimum length scale”.

The traditional implementation

Density filtering is an extension of sensitivity filtering [54]. It is inspired by anti-alias filters, which smoothen bitmaps in image processing. Anti-aliasing is typically used to prevent “pixelated” appearances, for instance after resizing operations. One might draw a parallel with the checkerboard topology seen in Figure 3.2.

The method is connected to the use of element densities as outlined in subsection 3.3.1. Take an element $c$ with corresponding density $\rho_e^1$. Now define a collection $N^c$ of elements $i$ with density $\rho_e^1$.

For every $\rho_e^1$, a physical density $\tilde{\rho}_e^1$ is introduced according to:

$$\tilde{\rho}_e^1 = \frac{1}{\sum_{i \in N^c} H_{ci}} \sum_{i \in N^c} H_{ci} \rho_i^1$$  \hspace{1cm} (3.12)

This represents a weighted average with weight $H_{ci}$. Region $N^c$ is described in the literature as a circle [4]. Its radius is $r_{\min}$ and its center coincides with the centroid of $c$. Weight $H_{ci}$ is supposed to be inversely proportional to the distance between the centroids of $c$ and $i$. The setup is illustrated in Figure 3.3a.
3.1. Classical concepts within SIMP

(a) Ideal, theoretical circle. True for regular meshes.  
(b) The circular shape is lost on irregular meshes.

Figure 3.3: The circle of influence for traditional implementations of the density filter.

Traditional implementations assume a regular grid, however [50] [4]. This means all elements are equal in size and shape. Rather than drawing a circle in physical space, only the indexes of the elements are considered. Weight $H_{cl}$ is computed as:

$$H_{cl} = \max \left(0, r_{min} - |c - i| \right)$$

(3.13)

The circular shape of $N_c$ is lost on irregular grids. This is demonstrated in Figure 3.3b.

Definition (3.13) implies that $r_{min}$ is expressed in terms of a number of elements as well. It is often made a function of refinement, and therefore given an interpretation of physical length. A minimum length scale for the topology is thereby introduced. As a direct consequence of density filtering, "structural members" have a minimum thickness. This is claimed to be beneficial for manufacturing [54] and leads to mesh insensitivity. Note how Figure 3.3b reveals that the minimum thickness is dependent on physical location for irregular meshes.

A choice must be made for the implementation of physical densities $\tilde{\rho}_{el}^{i1}$. The approach used here consists of the following steps for every iteration $\rho_{el}^{i1}$ [4]:

1. For all current $\rho_{el}^{i1}$, determine $\tilde{\rho}_{el}^{i1}$.
2. Compute "physical gradients" $\bar{g}_{el}^{i1} = \frac{\partial f}{\partial \tilde{\rho}_{el}^{i1}}$ according to the method outlined in subsection 3.1.3.
3. Use the chain rule to obtain gradients $g_{el}^{i1}$:

$$g_{el}^{i1} = \frac{\partial f}{\partial \rho_{el}^{i1}} = \sum_{i \in N_c} \frac{\partial \tilde{\rho}_{el}^{i1}}{\partial \rho_{el}^{i1}} \bar{g}_{el}^{i1} = \frac{1}{H_{el}} \sum_{i \in N_c} H_{el} \bar{g}_{el}^{i1}$$

(3.14)

Where the symmetry of both (3.13) and the definition of $N_c$ is used:

$$\frac{\partial \rho_{el}^{i1}}{\partial \rho_{el}^{i1}} = \frac{\partial \tilde{\rho}_{el}^{i1}}{\partial \rho_{el}^{i1}} \quad \forall c, i$$

(3.15)
4. Apply the descent method presented in section 2.4. Use $g_{e1}^e$ as gradient, but physical densities $\bar{\rho}_{e1}^e$ to determine volume.

Both (3.12) and (3.14) may be rewritten as pre-multiplication by a band matrix $H$. Its bandwidth is determined by $r_{\text{min}}$. Since $H$ is constant, it may be computed before optimization for an efficient implementation. [4]

As stated, the steps outlined above represent just one possible implementation of the density filter. One might alternatively replace the original densities $\rho_{e1}^e$ by $\bar{\rho}_{e1}^e$ entirely, for example. Gradients $\bar{g}_{e1}^e$ are then directly used in the descent method. Considerations like these underline the heuristic nature of the density filter. It does not represent a mathematically consistent approach. Besides the discussion on separated meshes in section 3.4, a robust reformulation of the optimization problem has been proposed in the literature. [51] [62].

**An improved implementation**

The aforementioned implementation of $N^c$ as an actual circle is possible for irregular grids as well. It requires extra computation, but prevents the situation illustrated in Figure 3.3b. Let all element centroids $\bar{x}_{e1}$ be precomputed according to:

$$\bar{x}_{e1} = \frac{\int x \, d\Omega}{\int d\Omega}$$ (3.16)

With integration carried out in all physical dimensions separately. Weight function $H_{e1}$ is then redefined as $H_{e1}^{i_1}$:

$$H_{e1}^{i_1} = \max(0, r_{\text{min}} - |\bar{x}_{e1} - \bar{x}_{i_1}|)$$ (3.17)

After which radius $r_{\text{min}}$ becomes a physical quantity. Keeping it constant ensures mesh insensitivity. The rest of the implementation remains the same. As explained in section 1.3, the focus in this report is on mathematically consistent formulations within SIMP. Even with (3.17), density filtering can be considered heuristic. This improved implementation is therefore not further studied. The traditional density filter implementation is used to generate the results in chapter 4.

### 3.2. Classical implementation of SIMP

This section outlines a straightforward, classical implementation of SIMP. It builds on the material presented in chapter 2 and section 3.1. The continuation method is specified, together with a convergence criterion and some general remarks for programming. A list of actual implementation steps is presented at the end.

#### 3.2.1. The continuation method and programming technique

A continuation scheme is used for the penalization power. It prevents convergence on “local optima” as discussed in section 1.3. The implementation used in this report is as follows. Penalization power $\theta$ equals $\theta^1$ in the first iteration. Over $N$ iterations, it is gradually increased to $\theta^f$. It subsequently remains constant. For iterations $i = 1, 2, \ldots$, this is written as:

$$\theta = \min_{\theta^1, \theta^f} \left( \theta^i + \frac{i-1}{N-1} (\theta^f - \theta^i), \theta^f \right)$$ (3.18)

Subsection 3.1.1 mentions a typical value $\theta^f = 3$. In line with subsection 3.1.4, $\theta^1 = 1$ ensures that optimization starts from a convex problem. By choosing $N = 21$, reduced penalization powers are used in the first 20 iterations.

Iteration stops when a convergence criterion is met. Several possibilities for such a criterion exist. Convergence criteria are typically not perfect, and a maximum number of allowed iterations should be used. Consider the end of iteration $i$. Let $\rho_{e1}^{c1}$ be the density value for element $c$. The following convergence criterion is used for this report:

$$|\rho_{e1}^{c1} - \rho_{e1}^{c1-1}| \leq 1 \times 10^{-3} \quad \forall c$$ (3.19)
This criterion is also found in example implementations [50] [4]. In line with the selection of the descent method in subsection 2.4.2, the simplicity of the criterion may lead to an excessive amount of iterations. However, a sufficiently high maximum does not change the final topology obtained. The criterion does not interfere with the main research focus, studying the SIMP modifications specified in section 1.3.

Figure 3.4a shows a typical compliance history for SIMP. After an initial rise due to the continuation method, compliance visually appears to decrease monotonously. It eventually appears to converge on a minimum. Figure 3.4b reveals that this is in fact not the case. A convergence criterion purely based on the sign of the compliance increment is therefore inappropriate.

Although programming is outside the scope of this report, some general remarks regarding technique are made. It is possible to create a relatively fast implementation using high-level programming languages. One should prevent having to do computations for each element sequentially. Take basis functions as an example. Their values for every element may be stored in one multi-dimensional array as:

\[ R[\text{element}, \text{function}, \text{location}] \]

Similar approaches can be used for other quantities. Software like MATLAB and Python’s numpy package efficiently operates on these arrays. By using it to calculate for all elements simultaneously, one automatically obtains a parallel implementation. Multi-core processors may then be used effectively.

In addition, stiffness matrix \( K \) and filter matrix \( H \) are sparse. Storing them accordingly saves system memory.

When programming, (3.1) is usually replaced by the alternate SIMP scheme [4]:

\[ C = C_{\min} + \rho^p (C_0 - C_{\min}) \quad (3.20) \]

With \( C_{\min} \) a very low stiffness value. It prevents the material from reaching absolute zero stiffness, which could otherwise complicate numerical solution of system (2.63). For readability, equation (3.1) is used throughout the report.

### 3.2.2. Implementation steps

A typical implementation starts with the following preparatory steps:

1. Determine the NURBS geometry and the loading situation.

2. Apply the desired refinement and determine sets of element extraction operators \( C^e_{\xi} \) and \( C^e_{\eta} \). See subsection 2.3.4.

3. Build the INC, IEN and ID arrays.

4. Determine integration points \( (\xi_i, \eta_j) \) and corresponding weights \( (\bar{w}_i, \bar{w}_j) \). Changes of domain have to be accommodated for through (2.113)–(2.116).
5. Compute Bernstein basis function values $B_{a,p}(\xi)$ and derivatives $\frac{dB_{a,p}}{d\xi}$ for all integration points. Use (2.77)–(2.79).

6. For every element, determine NURBS surface basis function values $\tilde{R}_{\text{el}}$, derivatives $\frac{dR_{\text{el}}}{dx}$ and Jacobians $f$. See subsection 2.3.3 and subsection 2.3.4.

7. Compute "full material" element stiffness matrix contributions $K^0_{\text{el}}$ with (3.3). Also calculate element force vector components $f^\text{el}$ according to (2.66).

8. Choose starting densities $\rho_{\text{el}}$ for every element. One typically sets all values equal to volume constraint $V$ in (2.24). An admissible, homogeneous density field is created that way. Alternatively, previously obtained density fields may be used.

9. Build and store the filter matrix $H$ according to (3.12) or (3.14).

The next steps are then carried out repeatedly:

10. Determine the appropriate penalization power $\theta$ for this iteration. Apply (3.18).

11. Calculate the actual element stiffness matrix contributions $K^\theta_{\text{el}}$ using (3.3). Assemble stiffness matrix $K$ and force vector $f$.

12. Solve matrix system (2.63) to obtain displacement $\hat{u}$.

13. Compute compliance $J_i$ using (2.70) and store it. Use (3.11) and pre-multiply by $H$ to obtain a filter-corrected gradient $g^\text{el}$ for each element.

14. Apply the descent method outlined in subsection 2.4.2. Determine physical densities through pre-multiplication by $H$ and calculate volume using (3.5).

15. Test for convergence with (3.19). Stop iterating when converged. Return to step 10 otherwise.

After the optimization loop terminates, some post-processing is typically involved. One may for instance store compliance history to disk. A plot of the final topology is usually constructed as well.

### 3.3. Consistent density representation

In virtually all articles available on SIMP, element-wise constant densities are adopted. Section 1.3 proposes the evaluation of a consistent density approximation. Subsection 2.2.6 indicates how displacement is represented using NURBS basis functions. As noted in subsection 3.1.2, consistency then requires the implementation of a continuous density field using NURBS as well. This section elaborates on the consistent density approximation, a corresponding consistent gradient derivation and implications for computer implementation.

#### 3.3.1. Continuous density

Subsection 3.1.4 identifies a potential link between element-wise constant densities and checkerboard formation. To evaluate this connection, density approximation (3.2) is replaced by a relation more in line with displacement representation (2.50). Consider a linear combination of basis functions:

$$\mathbf{D} = \left( \sum_{A=1}^{nm} R_A(\xi) \mathbf{\rho}_A \right) \mathbf{D}_0 \left( \mathbf{\bar{R}}^T \mathbf{\bar{R}} \right)^{\theta} \mathbf{D}_0 \quad (3.21)$$

Where density control variables and basis functions have been collected in vectors $\mathbf{\bar{p}}$ and $\mathbf{\bar{R}}$ respectively. This eases notation, analogously to section 2.3. Stiffness matrix components are evaluated as:

$$K_{PQ} = e_i^T \int_{\Omega_{\text{el}}} \left( \mathbf{\bar{p}}^T \mathbf{\bar{R}} \right)^{\theta} B_{ij} B_{jP} B_{lQ} d\Omega e_l \quad (3.22)$$
3.3. Consistent density representation

Contrary to (3.3), this may not be simplified to a multiplication with pre-determined full-material stiffness matrix components. Subsection 3.3.3 treats how implementation can still be done efficiently.

Volume determination (3.4) is expanded using a representation similar to (3.21):

\[ V = \frac{\int \rho \hat{R} \, d\Omega}{\int \hat{R} \, d\Omega} \]  

(3.23)

Which is evaluated using the methods described in subsection 2.3.4.

3.3.2. Consistent gradient derivation

The current section focuses on a mathematically consistent, continuous density representation. A gradient is therefore consistently derived from continuous formulation (2.18) as well. To illustrate the difference, a derivation started from discrete formulation (2.71) is also provided. Such a derivation is more in line with the common practice for element densities presented in subsection 3.1.3. The results of both derivations are compared at the end of this subsection.

Starting from the continuous formulation

An adjoint method is used to derive a partial derivative of discrete formulation (2.71) in subsection 3.1.3. In its continuous formulation (2.18), compliance \( J \) is a functional of density field \( \rho \). Its gradient cannot be defined through a partial derivative. The adjoint method must be applied in a continuous sense instead [27] [24].

Consider a scalar-valued function \( f \) with vector argument \( x \). The directional derivative in direction \( n \) is calculated through:

\[ \nabla_n f = \langle \nabla f, n \rangle = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i} n_i \]  

(3.24)

Where the inner product has been expanded in components. Tensor \( \nabla f \) is subsequently recognized as the gradient \( g \) of \( f \):

\[ g = \nabla f \]  

(3.25)

Inner products are defined for functions (or fields) on \( \Omega \) as well. Quite similarly to the sum in (3.24), one has:

\[ \langle f(x), g(x) \rangle = \int_{\Omega} f(x) g(x) \, d\Omega \]  

(3.26)

The equivalent to directional derivative (3.24) then reads:

\[ \delta J = \int_{\Omega} g \delta \rho \, d\Omega \]  

(3.27)

Where \( \delta J \) measures the first order change in compliance due to a change in density \( \delta \rho \). In this context, gradient \( g \) is called the functional derivative. The aim is to find an expression for it.

A change in compliance \( \delta J \) may be expanded using (2.18). One may write:

\[ \delta J = \int_{\Gamma} \delta t \cdot u + t \cdot \delta u \, d\Gamma \]  

(3.28)

For which \( \delta t \) and \( \delta u \) can be related to \( \delta \rho \) through:

\[ \delta t = \frac{\partial t}{\partial \rho} \delta \rho \]  

(3.29)

\[ \delta u = \frac{\partial u}{\partial \rho} \delta \rho \]  

(3.30)

Analytical expressions for these partial derivatives cannot be obtained. The situation is in this sense similar to that of subsection 3.1.3. One again continues through application of the adjoint method.
Let a constraint \( c(\rho) \) be constructed from (2.7):
\[
  c(\rho) = \div \left( C \nabla u \right) = 0
\]  
(3.31)

Using an arbitrary adjoint field \( u^*(\rho) \), functional \( J \) is augmented to an equal functional \( \tilde{J} \):
\[
  \tilde{J} = J + \langle c, u^* \rangle = \int_{\Gamma} t \cdot u \, d\Gamma + \int_{\Omega} \div \left( C \nabla u \right) \cdot u^* \, d\Omega
\]  
(3.32)

Divergence expansion (2.12) is rewritten as:
\[
  \div A \cdot b = \div \left( Ab \right) - A \cdot \nabla b
\]  
(3.33)
\[
  A \cdot \nabla b = \div \left( Ab \right) - \div A \cdot b
\]  
(3.34)

Now apply (3.33), traction definition (2.4) and the divergence theorem to (3.32). One obtains:
\[
  \tilde{J} = \int_{\Gamma} t \cdot (u + u^*) \, d\Gamma - \int_{\Omega} C \nabla u \cdot \nabla u^* \, d\Omega
\]  
(3.35)

A change in augmented functional \( \tilde{J} \) is expanded as well:
\[
  \delta \tilde{J} = \delta J = \int_{\Gamma} \left[ \delta t \cdot (u + u^*) + t \cdot (\delta u + \delta u^*) \right] \, d\Gamma
\]
\[
  - \int_{\Omega} \left[ \frac{\partial C}{\partial \rho} \nabla u \cdot \nabla u^* \delta \rho + C \nabla \delta u \cdot \nabla u^* + C \nabla u \cdot \delta u^* \right] \, d\Omega
\]  
(3.36)

One should note that the order of applying \( \nabla(\cdot) \) and \( \delta(\cdot) \) may be interchanged. This is because they represent derivatives in the physical and design space respectively. In topology optimization, these spaces are independent. Stiffness change \( \delta C \) has been expanded implicitly according to:
\[
  \delta C = \frac{\partial C}{\partial \rho} \delta \rho
\]  
(3.37)

Because \( \frac{\partial C}{\partial \rho} \) is a partial derivative for which an analytical expression can be obtained. Two terms in (3.36) may be expanded by using (3.34) and the major symmetry of \( C \):
\[
  \int_{\Omega} C \nabla \delta u \cdot \nabla u^* \, d\Omega = \int_{\Omega} C \nabla u^* \cdot \nabla \delta u \, d\Omega = \int_{\Gamma} t^* \cdot \delta u \, d\Gamma - \int_{\Omega} \div \left( C \nabla u^* \right) \cdot \delta u \, d\Omega
\]  
(3.38)
\[
  \int_{\Omega} C \nabla u \cdot \nabla \delta u^* \, d\Omega = \int_{\Gamma} t \cdot \delta u^* \, d\Gamma - \int_{\Omega} \div \left( C \nabla u \right) \cdot \delta u^* \, d\Omega
\]  
(3.39)

The term in (3.39) containing \( \div \left( C \nabla u \right) \) cancels out due to (3.31). Quantities that are prescribed on the boundary cannot change during optimization. Hence:
\[
  \delta t|_{\Gamma_N} = 0
\]  
(3.40)
\[
  \delta u|_{\Gamma_D} = 0
\]  
(3.41)

Expression (3.36) is simplified using (3.38)–(3.41):
\[
  \delta \tilde{J} = \int_{\Gamma_N} \left[ \delta t \cdot (u + u^*) \right] \, d\Gamma + \int_{\Gamma_D} (u + u^*) \cdot \delta t \, d\Gamma
\]
\[
  - \int_{\Omega} \left[ \frac{\partial C}{\partial \rho} \nabla u \cdot \nabla u^* \delta \rho + C \nabla \delta u \cdot \nabla u^* + C \nabla u \cdot \delta u^* \right] \, d\Omega
\]  
(3.42)
The undesired variations drop when choosing $u^*$ such that:

\[
\begin{aligned}
\text{div}(\mathbb{C} \nabla u^*) &= \text{div} \sigma^* = 0 \quad \text{on } \Omega \\
\begin{bmatrix}
\mathbb{C} \\
\nabla u^*
\end{bmatrix} &= 0 \quad \text{on } \Gamma_D \\
t^* &= t \quad \text{on } \Gamma_N
\end{aligned}
\] (3.43)

Which is analogous to (2.7)–(2.9). And so:

\[u^* = u\] (3.46)

As in subsection 3.1.3, the problem is self-adjoint. Variation $\delta J$ is expressed as:

\[\delta J = - \int_{\Omega} \frac{\partial C}{\partial \rho} \nabla u \cdot \nabla \rho \, d\Omega \] (3.47)

Or with SIMP approximation (3.1):

\[\delta J = - \int_{\Omega} \theta \rho^{\theta-1} c_0 \nabla u \cdot \nabla \rho \, d\Omega \] (3.48)

Comparison with (3.27) then yields an expression for the gradient:

\[g = -\theta \rho^{\theta-1} c_0 \nabla u \cdot \nabla \rho \] (3.49)

No infeasible numerical computations are required for evaluation. The gradient is also negative definite again. It has an explicit dependence on $\rho$ as long as $\theta > 1$.

Alongside the displacement and density fields, continuous gradient (3.49) must be approximated using basis functions $R_A$:

\[g = \hat{g}^T \hat{R} \] (3.50)

Vector $\hat{g}$ contains gradient control variables. Each entry corresponds to a density in $\hat{\rho}$. Computation of these entries is similar to finding vector components, displayed in Figure 3.5. Expand (3.50) using (3.49):

\[-\theta \rho^{\theta-1} c_0 \nabla u \cdot \nabla \rho = \sum_{A=1}^{nm} g_A R_A \] (3.51)

Introduce matrices $B$ and $D$ as defined in subsection 2.2.6. With (3.21), one has:

\[-\theta (\hat{\rho}^T \hat{R})^{\theta-1} D_0 \left( \sum_{c=1}^{nm} B_c u_c \right) \cdot \left( \sum_{b=1}^{nm} B_b u_b \right) = \sum_{A=1}^{nm} g_A R_A \] (3.52)

Or:

\[-\theta (\hat{\rho}^T \hat{R})^{\theta-1} \sum_{c=1}^{nm} \sum_{b=1}^{nm} u_c^T B_c B_b D_0 u_b = \sum_{A=1}^{nm} g_A R_A \] (3.53)

To find coefficients $g_A$, both sides of the equation are multiplied by an arbitrary basis function $R_B$ and integrated over domain $\Omega$. Interchanging both sides and rearranging, one obtains:

\[\sum_{A=1}^{nm} \int_{\Omega} R_A R_B \, d\Omega \, g_A = -\theta \sum_{c=1}^{nm} \sum_{b=1}^{nm} \int_{\Omega} u_c^T B_c B_b D_0 u_b \, d\Omega \] (3.54)

Given that this should hold for any basis function $R_B$, a system of equations results. This system is then written as a matrix equation:

\[A \hat{g} = b \] (3.55)
3. Variations on SIMP implementation

Figure 3.5: A vector and its components.

With:

$$A_{AB} = \int_{\Omega} R_A R_B \, d\Omega$$  \hspace{1cm} (3.56)

$$b_A = -\theta \sum_{c=1}^{nm} \sum_{d=1}^{nm} u_c^T \int_{\Omega} (\hat{R}^T \hat{R})^{q-1} R_A B_c^T D_b B_d \, d\Omega \, u_d$$  \hspace{1cm} (3.57)

Vector $\hat{g}$ contains the gradient control variables. Matrix $A$ is symmetric and sparse, just like stiffness matrix $K$. Solution of (3.55) is possible through various numerical procedures. The method applied for (2.63) may be used here as well.

**Starting from the discrete formulation**

For comparison with subsection 3.1.3, the derivation is started from discrete formulation (2.71) as well. One can define gradient values $g_A^d$ analogously to (3.11) as:

$$g_A^d = \frac{\partial J}{\partial \rho_A} = -\hat{u} \frac{\partial K}{\partial \rho_A} \hat{u}$$  \hspace{1cm} (3.58)

Both $g_A^d$ and $\rho_A$ now represent control variables. Stiffness matrix $K$ is defined component-wise through (3.22). It can be implemented in (3.58). The result is equal to:

$$g_A^d = -\theta \sum_{c=1}^{nm} \sum_{d=1}^{nm} u_c^T \int_{\Omega} (\hat{R}^T \hat{R})^{q-1} R_A B_c^T D_b B_d \, d\Omega \, u_d$$  \hspace{1cm} (3.59)

Which is in turn equal to the components of $b$, as defined in (3.57). This means that the gradient control variables as derived from the continuous and discrete formulations are equal if matrix $A$, defined through (3.56), is the identity matrix:

$$g_A^d = g_A \hspace{1cm} \text{if} \hspace{1cm} A_{AB} = \int_{\Omega} R_A R_B \, d\Omega = \delta_{AB}$$  \hspace{1cm} (3.60)

With $\delta_{AB}$ the Dirac delta function. Draw the analogy with vectors and the inner product again. Relation (3.60) then holds if the basis $R_A$ is orthonormal. This is generally not the case for NURBS. For mathematical consistency, gradient values must be computed through the solution of (3.55). That approach is applied to generate the results in chapter 4.

**Note:** gradient descend methods are “forgiving” when it comes to computation of the gradient. If the “direction” of the density updates is sufficiently close to the steepest descent direction, the topology will eventually approach its optimum. Preliminary tests show no visual difference between topologies obtained with either approximation of the continuous gradient.
3.3.3. Implementation of consistent densities

The use of a consistent density representation has consequences for the implementation steps outlined in subsection 3.2.2. Steps that change are 7–9, 11 and 13–14.

An important change is made to step 7. In the case of a consistent density approximation, recomputation of the stiffness matrix requires re-integration. This is outlined in subsection 3.3.1. Determination of a “full material” element stiffness matrix $K_{p0}^{d}$ is not useful. However, repeated integration can still be made efficient. Consider determining $K_{p0}^{d}$ through (3.22) and Gaussian quadrature. As mentioned in subsection 3.2.4, this is done element by element. Incorporating (2.120), one has for every element:

$$K_{p0}^{d} = e^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \left( \rho^{T} \tilde{R} \right)^{\theta} B_{A}^{T} D_{B} B_{B} f^{i} \right)_{\xi=\tilde{\xi}, \eta=\tilde{\eta}} \tilde{w}_{i} \tilde{w}_{j} e_{j}$$  \hfill (3.61)

One restricts sum $\rho^{T} \tilde{R}$ to basis functions with support on the element. The product $B_{A}^{T} D_{B} B_{B}$ is constant during optimization and may now be pre-determined for every integration point in a new step 7.

Step 11 changes to accommodate (3.61). Terms in the sum for every integration point, after which the actual summation follows. While the process is relatively straightforward, it is more computationally involved than the multiplications in the element-wise constant case.

Few alterations are made to step 8. Analogously to element densities, one may set all initial density control variables equal to volume constraint $\tilde{V}$. The overall constraint is then met because of the NURBS partition of unity property, listed in subsection 2.2.2.

The filters defined in subsection 3.1.5 are inherently related to element densities. There is no matrix $H$ to be obtained in step 9. However, matrix $A$ for the gradient calculation also remains constant throughout optimization. For that reason, it may be pre-computed here using (3.56).

Actual determination of the gradient in step 13 changes as well. Vector $b$ is built from (3.57), after which gradient control variables are found by solving (3.55). The descent method application in step 14 does not change. A gradient value is still available for each density value. In the absence of filtering, matrix $H$ is no longer used in either step. Density control variables may also directly be used for the convergence criterion in step 15.

Although many steps are altered, the overall process is highly similar. The consistent density approximation leads to an increase in the number of computations and hence to a longer computation time. However, the lack of a density filter application partially compensates.

3.4. Separate design and analysis mesh refinements

This section describes the use of an analysis mesh that is obtained by further refining the design mesh. The approach forms a mathematically sound alternative to (density) filtering as discussed in subsection 3.1.5. Its study is one of the central goals in this report, as mentioned in section 1.3. After both meshes are defined, they are used to redefine the calculation of displacements and gradients. Modifications to the implementation steps in subsection 3.2.2 are presented as well.

3.4.1. Defining the design and analysis mesh

The immediate effect of filtering is that densities are not calculated for each element independently. Subsection 3.1.5 mentions that filters may be considered heuristic in several ways. One may see the equations and derivations in chapter 2 as a “mathematical framework”. Filters cannot be implemented as an extension to that framework. Separate design and analysis meshes fulfill the same purpose as filters, while staying within the mathematical framework. The method is referred to as “patching” in section 1.2.

Define two distinct but related meshes. The design mesh is constructed first, through the procedure outlined in subsection 2.3.4. It has elements $n_{e_{1},d}$, for which piecewise constant densities $\rho^{e_{1},d}$ (subsection 3.1.2) can be specified. Let $R_{A}^{d} = R_{A_{B},d}^{d}$ be the $N$ NURBS basis functions defined over the design mesh. The consistent density representation treated in subsection 3.3 is then redefined as:

$$\rho(\xi^{d}) \approx \sum_{A=1}^{N} \rho_{A} R_{A}^{d}(\xi^{d})$$  \hfill (3.62)
3. Variations on SIMP Implementation

Variations on SIMP implementation design mesh

Figure 3.6: An example of distinct design and analysis meshes. Densities are defined on the former, while displacements are approximated using the latter.

Volume calculation (3.5) or (3.23) is changed accordingly.

An analysis mesh is obtained by further refining the design mesh. If the elements in it are $\Omega^e_{1,a}$, it holds that:

$$\Omega^e_{1,a} \subset \Omega^e_{1,d} \subset \Omega$$  \hfill (3.63)

Analysis elements are used to approximate displacements, analogously to subsection 2.2.6. Take $R^a_B = R^a_{B,p,q}$ as the $M$ NURBS basis functions over the analysis mesh. One may write:

$$\mathbf{u}(\xi^a) \approx \sum_{B=1}^{M} \mathbf{u}_B R^a_B (\xi^a)$$ \hfill (3.64)

Due to the way the meshes are constructed, $M$ is a multiple of $N$. An example of distinct design and analysis meshes is shown in Figure 3.6.

### 3.4.2. Modified displacement calculation

Displacement control variables $\mathbf{u}_B$ are obtained through the solution of system (2.63). Stiffness matrix $\mathbf{K}$ and force vector $\mathbf{f}$ are computed per element, as explained in subsection 2.3.4. Given the two sets of basis functions and their accompanying parent spaces, one must use either design or analysis elements for this purpose. This subsection will proceed using analysis elements. The possibility of using design elements is discussed in subsection 3.4.4.

Consider the element-wise constant density case first. Following derivations in subsection 2.2.6 with (3.64), definition (2.66) for $f_p$ changes to:

$$f_p = \int_{\Gamma_N} R^a_B \mathbf{t}_i \, d\Gamma$$ \hfill (3.65)

Evaluation using analysis elements is relatively straightforward, since there is no dependence on design mesh quantities. With element densities $\rho^{e1,d}$, definition (3.3) for $K_{pq}$ becomes:

$$K^{p1,a}_{pq} = e^T_{\Omega} \int_{\Omega^{e1,a}} (\mathbf{B}^a_{\theta})^T (\rho^{e1,d})^\theta \mathbf{D}_\theta \mathbf{B}^a_{\theta} \, d\Omega e_j = (\rho^{e1,d})^\theta (K^{p1,a}_{\theta})_{pq}$$ \hfill (3.66)

Which is only slightly more complicated to evaluate. One has to identify the correct design element $\Omega^{e1,d} \supset \Omega^{e1,a}$ and use the corresponding density $\rho^{e1,d}$.

The procedure is more complex for a consistent density approximation. Rewriting (3.22) with (3.62) yields:

$$K_{pq} = e^T_{\Omega} \int_{\Omega} (\hat{\rho}^T \hat{e}^T) (\mathbf{B}^a_{\theta})^T \mathbf{D}_\theta \mathbf{B}^a_{\theta} \, d\Omega e_j$$ \hfill (3.67)
3.4. Separate design and analysis mesh refinements

This may be computed on an analysis element analogously to (3.61) as:

$$K_{PQ}|_{e_{l,a}} = d^T \sum_{i=1}^{n} \sum_{j=1}^{m} \left[ (\hat{\rho}^T \hat{\mathbf{R}}^\theta) \mathbf{D}_\theta \mathbf{B}_\theta^* \mathbf{f}^t \right]_{\xi^2 = \xi^2_i, \eta^2 = \eta^2_j} \hat{w}_i \hat{w}_j \mathbf{e}_f$$  \hspace{1cm} (3.68)

Figure 3.7: Finding the location of analysis integration points within design parent space. They do not coincide with integration points used when integrating in design space.

Jacobian $f^t$ incorporates the physical integration domain $\Omega^{e_{1,a}}$. One uses the analysis parent space with integration points $(\xi^2_i, \eta^2_i)$. Design basis function values $R^d_\theta$ must be evaluated at the same physical locations. This means that corresponding points must be found in design parent space. It is a geometric consideration, as is seen in Figure 3.7. Within a design element, let analysis elements be indexed in the $\xi$ direction by $l = 1, 2, \ldots, R$. For the $\eta$ direction, let them be indexed by $j = 1, 2, \ldots, S$. A desired location $(\xi^d, \eta^d)$ is then calculated as:

$$\xi^d|_{\xi^2 = \xi^d_i} = \frac{(l - 1) + \xi^2_i}{R}$$  \hspace{1cm} (3.69)

$$\eta^d|_{\eta^2 = \eta^d_i} = \frac{(j - 1) + \eta^2_i}{S}$$  \hspace{1cm} (3.70)

Where use is made of the fact that both parent spaces span $[0, 1]$. Basis functions values $R^d_\theta$ for these points are determined through the procedures described in subsection 2.3.3. Components (3.68) may then be evaluated.

### 3.4.3. Modified gradient calculation

One needs a gradient value for every density variable. Subsection 3.4.1 defines densities on the design mesh. Gradients must therefore be specified on the design mesh as well. Their determination proceeds in a way similar to that in subsection 3.4.2. The case of element-wise constant densities is again fairly straightforward. With separate meshes, (3.11) is re-evaluated as:

$$g^{e_{1,d}} = -\hat{u} \frac{\partial K}{\partial \rho^{e_{1,d}}} \hat{u} = -\theta \left( \rho^{e_{1,d}} \right)^{\theta-1} \sum_{e_{l,a} \in e_{1,d}} \hat{u} K^{e_{1,a}}_0 \hat{u}$$  \hspace{1cm} (3.71)

Where the summation must be understood to be over all analysis elements within a design patch, $\Omega^{e_{1,a}} \subset \Omega^{e_{1,d}}$. These are the elements for which the stiffness matrix contribution is influenced by $\rho^{e_{1,d}}$. Expressions for consistent gradient control variables are obtained by modifying system (3.55). Analogously to subsection 3.4.2, matrix $A$ and vector $b$ are obtained by integrating over analysis elements.
In view of (3.62), definition (3.56) for $A_{AB}$ is modified to:

$$A_{AB} = \int_{\Omega} R_A^3 R_B^3 \, d\Omega$$  (3.72)

Additionally incorporating (3.64), definition (3.57) for components $b_A$ is rewritten as:

$$b_A = -\theta \sum_{\zeta=1}^{nm} \sum_{\eta=1}^{nm} u_c^T \left( \nabla^T R^d \right)^{\gamma-1} R_A^3 \left( B_c^A \right)^T D_b B_b^A \, d\Omega \, u_D$$  (3.73)

Equations (3.72) and (3.73) are computed similarly to (3.61). The respective results are:

$$A_{AB} \big|_{e_{1,a}} = \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} \left[ R_A^3 R_B^3 J_c^A \right]_{\xi^e = \xi^t, \eta^e = \eta^t} \tilde{w}_i \tilde{w}_j$$  (3.74)

$$b_A \big|_{e_{1,a}} = -\theta \sum_{\zeta=1}^{nm} \sum_{\eta=1}^{nm} u_c^T \left( \nabla^T R^d \right)^{\gamma-1} R_A^3 \left( B_c^A \right)^T D_b B_b^A \, d\Omega \, u_D$$  (3.75)

Both of which may be evaluated using the integration point mapping procedure depicted in Figure 3.7. Subsection 3.4.2 elaborates on the matter.

### 3.4.4. Integral evaluation over design elements

Domain (and boundary) integrals are evaluated over analysis elements in subsection 3.4.2 and subsection 3.4.3. Integration may in fact be carried out over any subdivision of the domain. The only condition is that sufficient refinement is used to accurately apply Gaussian quadrature, outlined in subsection 2.3.1. However, using the design or analysis mesh is convenient. This is due to the basis functions being defined on them and the mechanism provided by Bézier extraction, explained in subsection 2.3.3. The current subsection discussed evaluation of integrals over design elements.

One should find points $(\xi^a, \eta^a)$ in analysis parent space that physically correspond to design space integration points $(\tilde{\xi}^d, \tilde{\eta}^d)$, using Figure 3.8. An extra challenge compared to Figure 3.7 is that for every design integration point, one needs to find the corresponding analysis element. Adopt the same convention as in subsection 3.4.2. In direction $\xi$, let analysis elements be indexed within a design element by $I = 1, 2, \ldots, R$. Use $J = 1, 2, \ldots, S$ for the $\eta$ direction. The element is then selected according to:

$$I = \lceil \tilde{\xi}^d R \rceil$$  (3.76)

$$J = \lceil \tilde{\eta}^d S \rceil$$  (3.77)

For which the fact that both parent spaces span $[0, 1]$ is used again. Ceiling operator $\lceil \cdot \rceil$ rounds up to the nearest integer. The coordinates are computed as:

$$\xi^a |_{\xi^e = \tilde{\xi}^d} = R \mod \frac{\tilde{\xi}^d}{R}$$  (3.78)

$$\eta^a |_{\eta^e = \tilde{\eta}^d} = S \mod \frac{\tilde{\eta}^d}{S}$$  (3.79)

Where modulus operator $\mod \cdot$ calculates the remainder of a division.

Using design elements for integration requires an extra step when compared to the use of analysis elements. For that reason, integration over analysis elements is chosen for generating the results in chapter 4.

**Note:** Preliminary experiments show no visual differences between topologies generated with either integration procedure.
3.4.5. Implementation of separate mesh refinements

The implementation steps in subsection 3.2.2 are changed to accommodate the use of two meshes. Changing steps are 2–9, 11 and 13–14.

Step 2 and step 3 are effectively applied twice. One first refines to obtain a mesh of design elements $\Omega_{\text{el,d}}$, after which arrays INCd and IENd are computed for design. As the ID array relates to assembly of stiffness matrix $K$ (subsection 2.3.4), it has no purpose for the design mesh. Refinement is carried out once more, to obtain the mesh of analysis elements $\Omega_{\text{el,a}}$. The INCa, IENa and ID arrays are then determined for analysis.

Integration point calculation is amended in step 4 and step 5. The same holds for the determination of Bernstein basis function and derivative values. One also processes locations $(\xi^d, \eta^d)$ of analysis integration points in design parent space. Use is made of (3.69) and (3.70). Basis function values $R^d_a$ and $R^d_b$ need to be determined separately in step 6.

Step 7 and step 11 are adjusted for the displacement calculation outlined in subsection 3.4.2. In a similar way, step 8 and step 13 change to accommodate the gradient determination of subsection 3.4.3. As the split meshes approach replaces filtering, no $H$ matrix is computed for the latter step. Along the same lines, step 9 is canceled and step 14 is modified.

Although most steps change, the procedure is conceptually similar to that presented in subsection 3.2.2. Relative complexity mainly lies in the correct implementation of the procedure described in subsection 3.4.2 that is shown in Figure 3.7. The computational effort created by a refined analysis mesh is greater than that of pre-multiplication by a filter matrix.

Note that the procedure outlined above may be combined with a consistent density approximation and hence with the steps of subsection 3.3.3. One may even apply both a density filter and separate meshes. For readability, the procedure is omitted here. It mostly involves the same steps as listed in this subsection, without the modifications that relate to removing the filter. An example result is provided in section 4.2.

One should further remark that filters may be modified to properly account for irregular meshes, as shown in subsection 3.1.5. This is not trivially possible for separate meshes. Filters can also be used to create mesh insensitivity, by keeping $\rho^{\text{el,n}}$ physically constant. The only way two obtain a similar feature for split meshes, is to keep the design mesh fixed. In that case, the “resolution” of the topology no longer improves upon refinement.
Chapter 3 discusses SIMP and the two variations central to this report. This chapter presents topologies generated by these methods, such that their uses and effects may be evaluated. Values selected for material properties and optimization parameters are listed in Table 4.1. The material corresponds to aluminum. Loading is chosen such that a $1 \text{ cm}^2$ cross-section bar under unidirectional load undergoes 0.1% strain. It is specifically mentioned when topologies are obtained with deviating parameter values.

In line with the research goals outlined in section 1.3, this chapter aims to compare various concepts. A parameter study falls outside its scope. Various refinement levels and checkerboard prevention parameters are used, but chosen such that optimizations do not take excessive amounts of run time.

Section 4.1 demonstrates the concavity of penalized SIMP. It also evaluates the effect of a consistent density approximation. Section 4.2 describes topologies resulting from the use of filters and split meshes. The combination is discussed as well.

4.1. Effect of continuous densities on checkerboard formation

Chapter 1 identifies a potential link between checkerboard formation and element-wise constant densities. As part of the main goal in this report, section 3.3 presents a consistent, continuous density representation. This section studies its implications for SIMP, and its effect on checkerboard formation in particular.

The domain considered belongs to the so-called Messerschmidt-Bölkow-Blohm (MBB) beam. It is a common test case for topology optimization [50]. Because of symmetry, only one half is considered, shown in Figure 4.1a. Note the sliders on the left side, resulting from the symmetric cut-off. Figure 4.1b displays the NURBS domain used. It is generated from order 2 open knot vectors:

$$\Xi = \eta = [0, 0, 0, 1, 1, 1]$$

Table 4.1: Parameters used for optimization, unless otherwise stated. Material properties and loading data correspond to a physically realistic use case of aluminum.

<table>
<thead>
<tr>
<th>parameter</th>
<th>symbol</th>
<th>value</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young’s modulus</td>
<td>$E$</td>
<td>70</td>
<td>GPa</td>
</tr>
<tr>
<td>Poisson’s ratio</td>
<td>$\nu$</td>
<td>0.35</td>
<td>-</td>
</tr>
<tr>
<td>net load</td>
<td>$F$</td>
<td>7</td>
<td>kN</td>
</tr>
<tr>
<td>polynomial orders</td>
<td>$p, q$</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>volume fraction</td>
<td>$\bar{V}$</td>
<td>0.3</td>
<td>-</td>
</tr>
<tr>
<td>final penalization power</td>
<td>$p^f$</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>maximum iterations amount</td>
<td>$i_{\text{max}}$</td>
<td>1000</td>
<td>-</td>
</tr>
</tbody>
</table>
Comparative analysis of SIMP variations

(a) Geometry and loading.

(b) NURBS domain for 8x4 elements.

Figure 4.1: Problem setup for the (half) MBB beam.

Control points are evenly distributed over range $[0,2] \times [0,1]$:

$$\hat{P} = \begin{bmatrix}
(0, 0) & (1, 0) & (2, 0) \\
(0, \frac{1}{2}) & (1, \frac{1}{2}) & (2, \frac{1}{2}) \\
(0, 1) & (1, 1) & (2, 1)
\end{bmatrix} \quad (4.2)$$

All weights are set to $w = 1$. Note that after refinement, the outermost control points are placed closer to each other. This is because the procedure outlined in subsection 2.3.4 inserts knots with multiplicity 1. In contrast, the first and last knots in the open knot vectors have multiplicity $p + 1$. All elements have equal size nonetheless, in this rectangular case.

Discussion of visual optimization results

Subsection 3.1.4 argues how checkerboards are only formed for penalization powers $\theta > 1$. A minimum compliance problem is convex for $\theta = 1$. This conclusion is supported by Figure 4.2. Zones with intermediate densities are clearly visible. The isogeometric topology optimization routine reproduces unpenalized designs obtained with FEA.

Section 1.2 mentions how these topologies may be interpreted as metal plates with varying thickness. As such, they are valuable results. Topologies created with a consistent density approximation and with element-wise constant densities are similar. This holds both in terms of visual result and when comparing calculated compliance values. An advantage of the consistent density case is the smoothness of the result.

Checkerboard formation for element-wise constant densities is widely discussed in the literature [54]. Figure 4.3 shows that IGA changes checkerboard formation. Classical patches of "solid material
4.1. Effect of continuous densities on checkerboard formation

(a) Consistent density approximation. 999 iterations (max), 0.072 compliance.

(b) Element-wise constant density approximation. 999 iterations (max), 0.070 compliance.

Figure 4.2: Unpenalized ($\theta = 1$) MBB beam optimization results. 80×40 elements. Areas with intermediate densities and clear boundaries are visible.
elements connected through corner hinges are present. However, one also observes other infeasible design regions. Material appears in disconnected grid and line patterns.

Finer checkerboards form as the mesh is refined, and the overall topology changes. Even if checkerboards were feasible designs, the solution would not be mesh independent. Section 1.2 discusses how classical checkerboards are caused by a deficiency in the analysis method. Dedicated research should establish whether this holds for a consistent density representation as well.

Figure 4.4 illustrates the effect of a consistent density approximation on checkerboards. The use of a continuous density field does not prevent checkerboard formation. Despite the restriction to continuous density variation, the optimizer produces unrealistic topologies. Like in Figure 4.3, these topologies change upon refinement.
4.1. Effect of continuous densities on checkerboard formation

(a) 60×30 elements, 999 iterations (max), 0.095 compliance.

(b) 80×40 elements, 999 iterations (max), 0.108 compliance.

(c) 120×60 elements, 999 iterations (max), 0.109 compliance.

Figure 4.3: Penalized (θ = 3) MBB beam optimization results for an element-wise constant density approximation. Classical and non-classical types of checkerboard formation are observed. The phenomenon does not disappear upon refinement.
Comparative analysis of SIMP variations

(a) 60×30 elements, 442 iterations, 0.211 compliance.

(b) 80×40 elements, 560 iterations, 0.171 compliance.

(c) 120×60 elements, 999 iterations (max), 0.144 compliance.

Figure 4.4: Penalized ($\theta = 3$) MBB beam optimization results for a consistent density approximation. Checkerboard-like topologies still form. They do not disappear with refinement.
Table 4.2: Comparison of optimization results for the MBB beam with and without penalization. *: Maximum amount of iterations. **: These are single runs, so run time is only an indication.

<table>
<thead>
<tr>
<th>penal. power</th>
<th>density approx.</th>
<th>elements</th>
<th>iterations</th>
<th>compliance</th>
<th>run time**</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 element</td>
<td>80×40</td>
<td>999*</td>
<td>0.070</td>
<td>0:19:48</td>
<td></td>
</tr>
<tr>
<td>consistent</td>
<td>80×40</td>
<td>999*</td>
<td>0.072</td>
<td>0:29:34</td>
<td></td>
</tr>
<tr>
<td>3 element</td>
<td>60×30</td>
<td>999*</td>
<td>0.095</td>
<td>0:09:12</td>
<td></td>
</tr>
<tr>
<td>element</td>
<td>80×40</td>
<td>999*</td>
<td>0.108</td>
<td>0:21:31</td>
<td></td>
</tr>
<tr>
<td>element</td>
<td>120×60</td>
<td>999*</td>
<td>0.109</td>
<td>1:14:52</td>
<td></td>
</tr>
<tr>
<td>consistent</td>
<td>60×30</td>
<td>442</td>
<td>0.211</td>
<td>0:29:51</td>
<td></td>
</tr>
<tr>
<td>3 consistent</td>
<td>80×40</td>
<td>560</td>
<td>0.171</td>
<td>1:11:43</td>
<td></td>
</tr>
<tr>
<td>consistent</td>
<td>120×60</td>
<td>999*</td>
<td>0.144</td>
<td>5:10:33</td>
<td></td>
</tr>
</tbody>
</table>

Comparison of optimization data
Table 4.2 presents data on the optimization results in this section. Compliances for both unpenalized cases are comparatively close. Penalization slightly raises compliance values for the element-wise constant density cases. This may be explained as the drive towards “0–1 designs” effectively limiting the design space. Moreover, penalization increases compliance when intermediate densities are still present. In the case of element densities, refinement changes compliance by a relatively small amount.

More variation exists between compliance values for the consistent density case. The values are also significantly higher. Both observations may be explained by item 2 of subsection 2.2.2. NURBS require a minimum span to change between values 0 and 1. The property is illustrated in Figure 2.3. A penalized, consistent density representation with NURBS leads to necessary presence of intermediate densities. Higher compliance values are obtained as a result.

Subsection 2.3.4 specifies that knots are inserted with multiplicity 1 for refinement. Higher multiplicity would decrease the necessary span for NURBS to go from 0 to 1, as discussed in subsection 2.2.2. However, this would also decrease the NURBS continuity. Control variables and required computation time would increase. Future research may seek polynomial orders and knot multiplicity values such that compliance results and computational efficiency are simultaneously optimized.

One may also note that a consistent density approximation leads to a large increase in run time. This is partly caused by the additional computations involved (subsection 3.3.3), but also by a consistently higher number of required iterations. The latter may be improved by researching a better descent method like MMA [60]. A more suitable convergence criterion would also be valuable.

4.2. Checkerboard prevention through split meshes
Several checkerboard prevention schemes are applied and evaluated in this section.

Discussion of visual optimization results
Section 3.4 presents split design and analysis meshes as a mathematically consistent alternative to the density filter. The approach is combined with a consistent density representation and shown for various refinements in Figure 4.5. Note that the number of elements in both the design and analysis mesh are varied.

In agreement with the notes on “patching” in section 1.2, the use of separate meshes diminishes the formation of checkerboards. Small checkerboard regions can still be observed.

Intermediate densities are widely present. As discussed in section 4.1, this is a necessary consequence of density representation using NURBS. The consideration of increasing knot multiplicity may occur for this case as well.

As seen in Figure 4.5, the combination of consistent density representation with separated meshes produces two visual anomalies. Some structural members appear to be split up in multiple smaller bars. Other members seem to have an oscillating thickness. The latter characteristic can also be observed in an earlier study with NURBS density representation [29]. However, the report on that study omits implementation details. Additional research is necessary to establish the cause of these phenomena, and eliminate their occurrence together with residual checkerboards.
Figure 4.5 shows that refining the design mesh does not diminish the occurrence of these anomalies. The same holds for refining the analysis mesh, both independently and with respect to the design mesh.

The separate meshes scheme is combined with the element-wise constant density approximation to produce Figure 4.6. Checkerboard formation again diminishes, but is not eliminated. Note that a significant zone of intermediate densities is present in Figure 4.6b. This is likely to be the consequence of slow convergence, combined with the maximum amount of iterations being used. Apart from minor variations, topologies appear similar for different refinement levels. The differences are smaller than those observed in Figure 4.5. Most notably, the splitting of structural members does not occur here. Thickness oscillation is difficult to assess because of the piecewise constant nature of the topology.

Classical density filtering is shown in Figure 4.7 for comparison. As discussed in subsection 3.1.5, it is only defined for the element-wise constant density approximation. Refinement level and filter radius $r_{\text{min}}$ are varied. Resulting topologies contain structural members of significantly greater thickness than those in Figure 4.5 and Figure 4.6. This even holds when the minimum effective filter radius $r_{\text{min}} = 2$ is used.

Section 1.2 mentions how some authors consider minimum thickness to be advantageous from a manufacturing viewpoint. Given technologies such as additive manufacturing, this is debatable. The topologies obtained by using separated meshes provide more detail and better resemble classical Michell trusses [43]. Density filtering also forms an intermediate density “transition zone”, a feature made worse by larger radii $r_{\text{min}}$. One may note that when using IGA, density filter results appear similar to those obtained with FEA [50].

For completeness sake, separate design and analysis meshes are combined with the density filter in Figure 4.8. The result is visually similar to the application of only density filtering, Figure 4.7. As $r_{\text{min}} = 2$ is the minimum value for the filter to be active, one may consider density filtering to be dominant.
4.2. Checkerboard prevention through split meshes

(a) Refinement factor 2, 60×30 elements, 999 iterations (max), 0.160 compliance.

(b) Refinement factor 2, 80×40 elements, 999 iterations (max), 0.139 compliance.

(c) Refinement factor 3, 60×30 elements, 999 iterations (max), 0.151 compliance.

Figure 4.5: Penalized ($\theta = 3$) MBB beam optimization results for a consistent density approximation. The analysis mesh is refined with respect to the design mesh. Checkerboard formation diminishes. Small topology features are formed.
Comparative analysis of SIMP variations

(a) Refinement factor 2, 60×30 elements, 999 iterations (max), 0.110 compliance.

(b) Refinement factor 2, 80×40 elements, 999 iterations (max), 0.114 compliance.

(c) Refinement factor 3, 60×30 elements, 999 iterations (max), 0.113 compliance.

Figure 4.6: Penalized ($\theta = 3$) MBB beam optimization results for an element-wise constant density approximation. The analysis mesh is refined with respect to the design mesh. Checkerboard formation diminishes. Small topology features are formed.
Figure 4.7: Penalized ($\theta = 3$) MBB beam optimization results for an element-wise constant density approximation. The density filter is applied, producing structural members of relatively high thickness.
Figure 4.8: Penalized ($\theta = 3$) MBB beam optimization results for an element-wise constant density approximation. The analysis mesh is refined with respect to the design mesh and the density filter is applied. The result visually resembles the application of only a density filter, which is dominant. Refinement factor $2$, $r_{\text{min}} = 2$, $60 \times 30$ elements, 511 iterations, 0.184 compliance.
4.3. Additional optimization examples

This section studies additional geometry and loading cases, with the purpose of reinforcing the arguments made in section 4.1 and section 4.2. The choice of individual examples is arbitrary. Both the element-wise constant and consistent density representations are used, along with split meshes and filtering. To keep the presentation concise, different (design) refinements are omitted, along with different parameters for the checkerboard prevention schemes. Optimization parameters are kept as in Table 4.1.

The first case studies a linearly varied, distributed load. In previous sections, only a concentrated load is used. Geometry is a simple square, formed with knot vectors:

\[ \Xi = \mathcal{H} = [0, 0, 0, 1, 1, 1] \]  

(4.3)

Control points:

\[ \mathcal{P} = \left[ \begin{array}{ccc} (0, 0) & (1, 0) & (0, 1) \\ (0, \frac{1}{2}) & (\frac{1}{2}, \frac{1}{2}) & (1, \frac{1}{2}) \\ (0, \frac{1}{2}) & (\frac{1}{2}, 1) & (1, 1) \end{array} \right] \]  

(4.4)
And weights equal to 1. It is shown along with the loading situation and NURBS approximation in Figure 4.9. Results are obtained using 60×60 elements.

Figure 4.10 presents results obtained without a checkerboard prevention method. They are similar to the findings in section 4.1. Classical and non-classical checkerboard formation is again observed.

Topologies for respectively the density filter with element densities, split meshes with element densities and split meshes with consistent densities are shown in Figure 4.11. Filtering produces structural members with greater thickness, while both split mesh cases show more feature detail. Lack of convergence can again be concluded from the regions of intermediate density in the element density variant. Split-up of structural members as observed in Figure 4.5 is again present. Moreover, some of the smaller bars seem to disconnect. This confirms the phenomenon to be infeasible. One also observes thickness oscillation.
(a) Element densities, 999 iterations (max), $8.643 \times 10^{-5}$
(b) Consistent densities, 999 iterations (max), $9.612 \times 10^{-5}$

Figure 4.10: Distributed load case optimization results with checkerboards. 60×60 elements.
4. Comparative analysis of SIMP variations

(a) Element densities, density filter, 356 iterations, 
Element densities, separate meshes, 999 iterations, 
1.267 \times 10^{-4} \text{ compliance.}

(b) Element densities, separate meshes, 999 iterations, 
(\text{max}), 8.195 \times 10^{-5} \text{ compliance.}

(c) Consistent densities, separate meshes, 999 iterations, 
(\text{max}), 8.141 \times 10^{-5} \text{ compliance.}

Figure 4.11: Distributed load case optimization results. 60\times60 elements.
Figure 4.12 shows the optimization setup for a cantilever beam with a rounded support. While the previous example deals with a variation in loading, this case illustrates a change in geometry. The NURBS representation becomes slightly more complex. Required knot vectors are:

\[ \Xi = [0, 0, 0, 1, 1, 1] \]  
\[ \eta = [0, 0, 0, 1, 1, 2, 2, 2] \]  

Note that due to the choice for \( \eta \), the unrefined surface already contains two elements. Control points must be chosen as:

\[ \hat{P}_T = [ (0.00, 0.40) (0.10, 0.40) (0.10, 0.50) (0.10, 0.60) (0,0.60) \]  
\[ (0.00, 0.20) (1.05, 0.20) (1.05, 0.50) (1.05, 0.80) (0,0.80) \]  
\[ (0.00, 0.00) (2.00, 0.00) (2.00, 0.50) (2.00, 1.00) (0,1.00) ] \]  

And the weights as:

\[ \hat{W}_T = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} & 1 \end{bmatrix} \]  

Results with checkerboards are displayed for 80×80 elements in Figure 4.13. In this case, non-classical checkerboard variations dominate. Their pattern follows the irregular mesh illustrated in Figure 4.12b.

Figure 4.14 presents the various checkerboard prevention options. What may be observed is the tendency of features to “taper” outwards from the center. This is a consequence of the larger elements near the outer boundary as shown in Figure 4.12b. Subsection 3.1.5 and section 3.4 mention how the...
Comparative analysis of SIMP variations

(a) Element-wise constant density approximation, 956 iterations, $3.296 \times 10^{-1}$ compliance.

(b) Consistent density approximation, 999 iterations (max), $7.016 \times 10^{-1}$ compliance.

Figure 4.13: Round base geometry optimization results with checkerboard formation. 80×80 elements.
issue is related to the limited applicability of both filters and split meshes to irregular grids. As mentioned, a more advanced implementation will prevent the phenomenon for filters. Additional research might produce similar opportunities for the split meshes approach.

Topologies are similar to previous results. The filtered solution has structural members with greater thickness, while the separate mesh variants show more detail. Split structural members seem to be less of an issue than they are for the MBB beam and distributed load case. However, thickness oscillation is clearly observed. For this geometry especially, one may note how the split meshes approach leads to a topology that better resembles classical Michell trusses.
(a) Element-wise constant density approximation, density filter, 227 iterations, \(6.352 \times 10^{-1}\) compliance.

(b) Element-wise constant density approximation, analysis mesh refined with respect to design mesh, 999 iterations (max), \(3.934 \times 10^{-1}\) compliance.

(c) Consistent density approximation, analysis mesh refined with respect to design mesh, 999 iterations (max), \(4.361 \times 10^{-1}\) compliance.

Figure 4.14: Round base geometry optimization results. 80x80 elements.
4.3. Additional optimization examples

The third optimization example represents part of a wheel, loaded at its axis. Figure 4.15 contains the geometry, loading and NURBS domain. Knot vectors are:

\[ \Xi = H = [0, 0, 0, 1, 1, 1] \]  \hspace{1cm} (4.9)

With control points:

\[ \hat{P}^T = \begin{bmatrix} (0.050, 0.000) & (0.475, 0.000) & (1.000, 0.000) \\ (0.050, 0.050) & (0.475, 0.475) & (1.000, 1.000) \\ (0.000, 0.050) & (0.000, 0.475) & (0.000, 1.000) \end{bmatrix} \]  \hspace{1cm} (4.10)

And weights:

\[ \hat{W} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 \end{bmatrix} \]  \hspace{1cm} (4.11)

Checkerboard results for 60×60 elements are shown in Figure 4.16. Topologies are similar to the previous cases. Checkerboard prevention results presented in Figure 4.17 are also in line with previous findings. The difference in feature detail is observed, as well as the “taper” towards the outer border. Both split-up of structural members and member thickness oscillation are present.
(a) Element-wise constant density approximation, 999 iterations, \(4.517 \times 10^{-2}\) compliance.

(b) Consistent density approximation, 999 iterations, \(8.758 \times 10^{-2}\) compliance.

Figure 4.16: Wheel geometry optimization results with checkerboard formation. 60x60 elements.
4.3. Additional optimization examples

(a) Element-wise constant density approximation, density filter, 999 iterations (max), $1.159 \times 10^{-1}$ compliance.

(b) Element-wise constant density approximation, analysis mesh refined with respect to design mesh, 999 iterations (max), $4.836 \times 10^{-2}$ compliance.

(c) Consistent density approximation, analysis mesh refined with respect to design mesh, 999 iterations (max), $6.482 \times 10^{-2}$ compliance.

Figure 4.17: Wheel geometry optimization results. 60×60 elements.
Table 4.4: Comparison of optimization results for the various tested geometries. *: Maximum amount of iterations. **: These are single runs, so run time is only an indication.

<table>
<thead>
<tr>
<th>problem</th>
<th>c.b. prevention</th>
<th>density approx.</th>
<th>iterations</th>
<th>compliance</th>
<th>run time**</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBB</td>
<td>-</td>
<td>element</td>
<td>999*</td>
<td>1.083 x 10^-1</td>
<td>0:21:31</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>consistent</td>
<td>560</td>
<td>1.706 x 10^-1</td>
<td>1:11:43</td>
</tr>
<tr>
<td></td>
<td>filter</td>
<td>element</td>
<td>973</td>
<td>1.578 x 10^-1</td>
<td>0:20:09</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>element</td>
<td>999*</td>
<td>1.140 x 10^-1</td>
<td>3:13:15</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>consistent</td>
<td>999*</td>
<td>1.392 x 10^-1</td>
<td>5:44:35</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>element</td>
<td>999*</td>
<td>8.643 x 10^-5</td>
<td>0:32:02</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>consistent</td>
<td>999*</td>
<td>9.612 x 10^-5</td>
<td>2:14:22</td>
</tr>
<tr>
<td>Dist. load</td>
<td>filter</td>
<td>element</td>
<td>356</td>
<td>1.267 x 10^-4</td>
<td>0:11:12</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>element</td>
<td>999*</td>
<td>8.195 x 10^-5</td>
<td>3:58:17</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>consistent</td>
<td>999*</td>
<td>8.141 x 10^-5</td>
<td>7:30:41</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>element</td>
<td>956</td>
<td>3.296 x 10^-1</td>
<td>0:52:17</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>consistent</td>
<td>999*</td>
<td>7.016 x 10^-1</td>
<td>2:35:59</td>
</tr>
<tr>
<td>Round base</td>
<td>filter</td>
<td>element</td>
<td>227</td>
<td>6.352 x 10^-1</td>
<td>0:11:40</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>element</td>
<td>999*</td>
<td>3.934 x 10^-1</td>
<td>9:16:32</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>consistent</td>
<td>999*</td>
<td>4.361 x 10^-1</td>
<td>12:09:09</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>element</td>
<td>999*</td>
<td>4.517 x 10^-2</td>
<td>0:25:51</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>consistent</td>
<td>999*</td>
<td>8.758 x 10^-2</td>
<td>2:26:36</td>
</tr>
<tr>
<td>Wheel</td>
<td>filter</td>
<td>element</td>
<td>999*</td>
<td>1.159 x 10^-1</td>
<td>0:26:01</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>element</td>
<td>999*</td>
<td>4.836 x 10^-2</td>
<td>4:02:38</td>
</tr>
<tr>
<td></td>
<td>split meshes</td>
<td>consistent</td>
<td>999*</td>
<td>6.482 x 10^-2</td>
<td>6:19:19</td>
</tr>
</tbody>
</table>

Comparison of optimization data

Table 4.4 lists the data of the various optimization examples. Results for the MBB beam with 80x40 elements have been included for completeness. The lowest compliance is generally achieved for the element density case with checkerboard formation. As stated in section 4.2, this is an unrealistic result caused by a deficiency in the analysis method. The same cannot be observed for consistent densities, likely due to necessary inclusion of intermediate density as discussed in section 4.2.

Of the checkerboard prevention cases, filters lead to the highest compliance values. This again follows from the restricted design space created by minimum thickness specification and the intermediate density “transition zones”. For split meshes, the element densities lead to lower compliances, the distributed load case excepted. Additional research might establish a preference for either approach in specific optimization cases.

The maximum amount of allowed iterations is reached in almost every case, indicating a need for studies into descent methods and convergence criteria. It is theoretically possible that the maximum number of iterations used is too low. However, the aforementioned optimization aspects have been chosen for their simplicity. There likely is room for improvement. Only the density filter cases occasionally require relatively few iterations, which might again be related to their restricted design space.

Run times are generally considerably higher when using a consistent density approach. Although the method requires more computations, future research might look for a more efficient implementation. The use of split meshes also requires far more run time than the use of density filters (or using no checkerboard prevention). This is the price of using a design space that is less restricted.

Some conclusions are drawn. IGA changes the pattern of the checkerboards observed. One may state that consistent densities alone are insufficient to prevent checkerboard formation. In combination with the split meshes approach, they lead to anomalous phenomena. Split meshes produce more detailed topologies than density filters. As these topologies better resemble classical Michell trusses, they may be considered to be more insightful. Filtered results are visually similar to those classically obtained using SIMP with FEA. Both a consistent density representation and split meshes drive up run time.
This report presents an implementation of an isogeometric analysis tool for topology optimization. The study is restricted to the SIMP method for 2D minimum compliance problems. By utilizing the concept of Bézier extraction, changes from FEA to IGA are restricted to the so-called “shape routine”. Classical SIMP implementations are therefore largely re-usable.

Within SIMP, two main variations are explored. A consistent, continuous density approximation is used instead of the traditional element-wise constant density representation. Effects on checkerboard formation are studied. The change makes the optimization procedure more complex. It is especially the calculation of the design gradient that becomes mathematically more involved.

Checkerboard formation is traditionally prevented using a density (or sensitivity) filter. Filtering as a method is often considered heuristic, and adds an extra step to the optimization procedure. An alternative is found in using separate meshes for “design” (density definition) and “analysis” (displacement definition). The analysis mesh is then obtained by further refining the design mesh. Inclusion of these meshes is mostly straightforward, but requires extra care when performing Gaussian integration.

Analysis of these concepts leads to several conclusions and recommendations. They are gathered in this chapter.

### 5.1. Conclusions about isogeometric SIMP variations

These conclusions are drawn throughout this report.

1. A consistent, continuous density approximation changes checkerboard formation, but does not prevent it. The inability to generate discontinuous designs does not restrain the analysis tool from driving optimization towards checkerboard-like topologies. Furthermore, the continuity of NURBS enforces occurrence of intermediate densities.

2. The use of separate design and analysis meshes diminishes checkerboard formation. It does not completely prevent it. Compared to density filtering, this method results in more detailed topologies, that better resemble classical Michell trusses. Obtained compliance values are also lower.

3. IGA integrated within SIMP allows for the evaluation of arc-like sections, but produces non-classical types of checkerboard formation. When applying a density filter, results are highly similar to those typically obtained using SIMP with FEA.

4. Combination of IGA with consistent density approximation and split meshes produces anomalous features in the topology. Structural members are split up in several bars. These bars sometimes disconnect, demonstrating the infeasibility of this effect. Thickness of structural members also oscillates.

The demonstrated concepts provide various insights that are new to the scientific literature. At the same time, many observed phenomena call for additional research into their origins. Combined with a density filter, IGA may be substituted for FEA when creating industry applications. Use of a consistent density representation and separate meshes requires more research first.
5.2. Recommendations for future research

The following opportunities for further research present themselves over the course of this report.

1. Checkerboard formation occurs despite the use of a consistent density representation. One may evaluate whether this is due to the analysis method not incorporating the infeasibility of checkerboard designs, as is the case for checkerboards with element densities.

2. Research is required to explain the exact form of checkerboard formation that is observed. As non-classical checkerboards are produced by both density approximation strategies, it is likely that the form is related to IGA.

3. Density approximation using NURBS enforces occurrence of intermediate densities. One may seek a combination of NURBS polynomial orders and knot multiplicity that simultaneously optimizes intermediate density elimination and computational efficiency.

4. Further study may be conducted on the observed anomalies of split-up and oscillating thickness of structural members. They occur when combining IGA, a consistent density representation using NURBS and separate design and analysis meshes. The goal would be to eliminate them, together with residual checkerboard formation.

5. When using separate meshes, neither density approximation scheme consistently leads to better compliance values. It could be interesting to study the cause of this variation and see if one method can be preferred over the other for specific optimization problems.

6. Proper implementation of density filters allows them to introduce constant structural member thickness on irregular meshes. The same is not trivially the case for the separate meshes approach. A solution would be valuable.

7. It is desirable to have a better descent method and convergence criterion for the optimization procedure. The used maximum value of 999 iterations is reached in the majority of problems, with some results clearly showing lack of convergence. Although this value might be insufficiently high for non-filtered optimization, the simplicity of the descent method and convergence criterion used leaves opportunity for improvement. A starting point for research could be the widely used MMA [60].

8. The consistent density approximation requires more iterations and a significantly longer run time than its element-wise constant counterpart. A reason for this may be found in the higher number of computations required. Nonetheless, it would be beneficial to have a more efficient implementation.
Validation of IGA code

Validation for topology optimization is a debated topic in the literature [42]. Authors often resort to visual comparison for determining the quality of optimization results, as quantitative validation methods remain unavailable.

The IGA code of an optimizer may be validated more thoroughly. Stress and strain field results are compared to an analytical solution. Section A.1 presents the *Airy stress function* as a means of obtaining such a solution. The actual validation is carried out in section A.2. Table 4.1 lists the optimization parameters used.

### A.1. Airy stress function

This section briefly outlines the *Airy stress function*. More elaborate presentations are available in the literature [57].

Refer to Figure 2.1 and the assumptions listed in Table 1.1. In what follows, domain \( \Omega \) is considered to undergo zero body force. Restating (2.5):

\[
\mathbf{b} = 0
\]

One may then define an *Airy potential* \( \phi(x, y) \) that satisfies:

\[
\frac{\partial^4 \phi}{\partial x^4} + 2 \frac{\partial^4 \phi}{\partial x^2 \partial y^2} + \frac{\partial^4 \phi}{\partial y^4} = 0
\]

Consider now \( \Gamma_N \) with normal \( \mathbf{n} = (n_1, n_2) \) and traction \( \mathbf{t} = (t_1, t_2) \). Additional conditions on \( \phi \) are specified as:

\[
\frac{\partial^2 \phi}{\partial y^2} n_1 - \frac{\partial^2 \phi}{\partial x \partial y} n_2 = t_1
\]

\[
\frac{\partial^2 \phi}{\partial x^2} n_1 - \frac{\partial^2 \phi}{\partial x \partial y} n_2 = t_2
\]

If an Airy potential \( \phi \) satisfying (A.1) and (A.2)-(A.3) can be found, stresses may be calculated as:

\[
\sigma_{11} = \frac{\partial^2 \phi}{\partial y^2}
\]

\[
\sigma_{22} = \frac{\partial^2 \phi}{\partial x^2}
\]

\[
\sigma_{12} = -\frac{\partial^2 \phi}{\partial x \partial y}
\]

After which strains can be obtained through inversion of (2.2). To that end, introduce a compliance matrix:

\[
\epsilon = \frac{1}{E} \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} \sigma
\]

71
There is no analytical procedure for obtaining Airy stress potential $\phi$. In line with many methods for solving partial differential equations, one typically starts by choosing a generic form. The aim is then to adjust its parameters such that (A.1) and (A.2)-(A.3) are satisfied. One should note that the approach above is best suited to problems with traction prescribed boundaries. Boundaries with prescribed displacements may be incorporated as well, but only as an approximation.

A.2. Validation of implementation

Through an Airy stress function, the IGA code of a topology optimization routine may be validated. This validation is carried out in this section.$^1$

Total strain energy $U$ is defined as:

$$ U = \frac{1}{2} \int_{\Omega} \sigma \cdot \varepsilon \, d\Omega $$

(A.8)

Its integrand may be computed from an Airy stress potential. Consider the cantilever beam illustrated in Figure A.1a and the Airy potential:

$$ \phi = -\frac{3P}{4ab}xy + \frac{P}{4a^2b}y^3 $$

(A.9)

The potential satisfies (A.1). However, it only partially complies with (A.2)-(A.3). The displacement boundary condition cannot be satisfied in the upper and lower right corners. As mentioned in subsection A.1, Airy stress functions are better suited to traction boundary conditions. However, due to Saint-Venant’s principle, the solution far enough away from the boundary is left unaffected.

$^1$Presentation is based on [http://solidmechanics.org/Text/Chapter5_2/Chapter5_2.php](http://solidmechanics.org/Text/Chapter5_2/Chapter5_2.php).
Stresses are computed according to:

\[ \sigma_{11} = \frac{3P}{2a^3b} x y \]  
\[ \sigma_{22} = 0 \]  
\[ \sigma_{12} = \frac{3P}{4ab} \left( 1 - \frac{y^2}{a^2} \right) \]  

Strains are found through (A.7):

\[ \epsilon_{11} = \frac{1}{E} \sigma_{11} - \frac{nu}{E} \sigma_{22} = \frac{3P}{2Ea^3b} x y \]  
\[ \epsilon_{22} = \frac{v}{E} \sigma_{11} + \frac{1}{E} \sigma_{22} = -v \frac{3P}{2Ea^3b} x y \]  
\[ \epsilon_{12} = \frac{1}{E} \sigma_{12} = (1 + v) \frac{3P}{4Ea^3b} (a^2 - y^2) \]  

Which enables computation of the integrand in (A.8):

\[ \sigma \cdot \epsilon = \frac{9P^2}{16Ea^6b^2} \left[ 4x^2y^2 + (1 + \nu) (a^2 - y^2)^2 \right] \]  

Through (2.1) and (2.3), one has:

\[ \sigma \cdot \epsilon = \nabla \mathbf{u} \cdot \nabla \mathbf{u} \]  

Apply consistent gradient expression (3.49) with \( \theta = \rho = 1 \) for comparison. One may then write:

\[ g|_{\theta=\rho=1} = -\sigma \cdot \epsilon \]  

Note that \( \rho = 1 \) is required because it affects displacement \( \mathbf{u} \). Distribution (A.16) may directly be compared to the consistent gradient obtained through (3.55).

Model a cantilever beam with a NURBS surface that has polynomial order \( p = q = 2 \). Consider as knot vectors:

\[ \Xi = \Pi = [0, 0, 0, 1, 1, 1] \]  

Control points can be spread equally over the range \([0,4] \times [0,1] \):

\[ \mathring{\mathbf{P}} = \begin{bmatrix} (0,0) & (2,0) & (4,0) \\ (0,\frac{1}{2}) & (2,\frac{1}{2}) & (4,\frac{1}{2}) \\ (0,1) & (2,1) & (4,1) \end{bmatrix} \]  

All weights are set to \( w = 1 \). An example for 12×3 elements is shown in Figure A.1b.

Figure A.2 presents a comparison. The similarity is not immediately obvious. Strain energy peaks in the upper and lower right corners for both plots, driving the contour scale up. A plot of the difference between the two in Figure A.2c reveals that they are in fact similar. Figure A.3 shows plots of the optimization result for different refinements. One may observe how the "singularities" become more concentrated in the corners as mesh is refined. As stated above, the Airy stress function does not satisfy the boundary conditions in these corners. Fine plots as generated by the optimizer may be considered more accurate than the Airy stress function result.
A. Validation of IGA code

(a) Result obtained from Airy stress potential.

(b) Result obtained from optimization routine with $100 \times 25$ elements.

(c) Difference between the two.

Figure A.2: Validation results for a cantilever beam with consistent density representation.
A.2. Validation of implementation

(a) 50×12 elements.

(b) 200×50 elements.

Figure A.3: Validation results for a cantilever beam with consistent density representation. Obtained from optimization routine.
Validation of IGA code

(a) Result obtained from Airy stress potential.

(b) Result obtained from optimization routine with 100×25 elements.

(c) Difference between the two.

Figure A.4: Validation results for a cantilever beam with element-wise constant density representation.

Similar validation can be carried out for the element-wise constant density case. Gradient (3.11) is defined in terms of the full material element stiffness matrix \( K_e \). By (3.3), components of this matrix are calculated as integrals over the element. Product (A.16) may be compared if it is integrated over the element as well:

\[
g_e^{\text{el}} \big|_{\theta=\rho=1} = - \int_{\Omega} \sigma \cdot \epsilon \, d\Omega \tag{A.21}
\]

The comparison is shown in Figure A.4. It is similar to the results for the continuous density case. One observes minor deviation near the point of load application and in the corners, where analysis is least accurate. Comparing the results with that of Figure A.2 and Figure A.3, strain energy appears to be less concentrated in the corners. This conclusion is reinforced by the consideration of different optimization refinements in Figure A.5. It is explained by realizing that gradients are now “smeared out” over elements. The maximum gradient value is therefore not as high, which reduces the necessary contour scale.

The optimizer is shown to compute gradients in agreement with Airy stress potential theory. It is capable of better handling the singularities of strain energy in the upper and lower right corners. The IGA analysis component of the optimizer is successfully validated.
A.2. Validation of implementation

(a) 50×12 elements.

(b) 200×50 elements.

Figure A.5: Validation results for a cantilever beam with element-wise constant density representation. Obtained from optimization routine.
Glossary

**CAD** Computer Aided Design. 2, 4, 5

**CONLIN** Convex Linearization. 2

**FEA** Finite Element Analysis. 2, 4, 5, 10, 12, 17, 21, 30, 46, 61, 67, 68

**IGA** Isogeometric Analysis. 2, 4, 5, 10, 12, 30, 46, 49, 51, 57, 61, 67, 68, 70, 73

**MMA** Method of Moving Asymptotes. 2, 5

**NURBS** Non-Uniform Rational B-Splines. 2, 4–6, 10–15, 17, 19–24, 27, 28, 33, 34, 38–40, 45, 46, 49, 51, 56, 57, 59, 61, 63, 67, 68, 71

**SIMP** Solid Isotropic Material with Penalization. 1–7, 24, 27–30, 32–34, 37, 45, 51, 61, 67, 69


