2021

# ENHANCEMENT AND BENCHMARKING OF THE PRESCRIBED MATERIAL REDISTRIBUTION TOPOLOGY OPTIMIZATION SCHEME

Tjard Bätge
*University of Rhode Island*, tjard.baetge@web.de

# ENHANCEMENT AND BENCHMARKING OF THE PRESCRIBED MATERIAL REDISTRIBUTION TOPOLOGY OPTIMIZATION SCHEME

BY

TJARD BÄTGE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2021

MASTER OF SCIENCE THESIS

OF

TJARD BÄTGE

APPROVED:

Thesis Committee:

Major Professor David G. Taggart

Helio Matos

George Tsiatas

Brenton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2021

## ABSTRACT

A rising demand for lightweight structures and the advance of additive manufacturing led to an increased importance of topology optimization tools and research. One developed scheme is the "Prescribed Material Redistribution" (PMR) method, which heuristically solves minimum compliance problems under a single volume constraint. This method has not yet been fully evaluated and benchmarked.

In this research, the current standalone PMR code is enhanced by using the commercial finite element software Abaqus to improve performance as well as ease of use. The novel code is then validated for 2D and 3D cases. Subsequently, a benchmark is conducted to compare the PMR approach with the "Solid Isotropic Material with Penalization" (SIMP) and "Rational Approximation of Material Properties" (RAMP) formulations commercially implemented in Simulia's Tosca Structure. The benchmark framework and 6 sample cases are defined based on a literature review. The schemes' efficiency and solution quality are investigated quantitatively and visually-qualitatively in a parametric study on 306 test cases. The analysis includes checkerboarding, mesh distortion, mesh density, and different element formulations.

Hypotheses on the effects of individual parameters are checked, and reasonable settings for the methods are derived from the findings. The benchmark findings indicate that the novel PMR code is less efficient than the commercial SIMP and RAMP implementations and the initial academic PMR version. The ease of use and the functionality is, however, significantly enhanced. The PMR and the SIMP method generated visually better results than the RAMP scheme. Based on these findings, the commercial viability of the PMR method is discussed in reference to recent topology optimization literature.

## ACKNOWLEDGMENTS

There are many people I want to thank for their personal support during my studies at the University of Rhode Island. I would not have come to this point without your help, and I am very grateful for that, especially in times of a global pandemic.

First of all, I want to thank each member of my committee in the US and in Germany for their academic advice. In particular, I want to express my appreciation to my major professor, Dr. David Taggart. You have always provided me with extraordinarily helpful feedback to whatever problems I ran into and with all the resources I could ask for. Your research and teaching expertise is impressive, thank you.

I also want to express my gratitude to Matt Buglio and Felix Klempt. I highly appreciate the time and effort you invested in my research by assessing all the different topologies. Besides, I want to thank you for the mutual support during our hours together in the lab.

Additionally, I want to thank Peter Phelps. You are an inspiring source of information and your advice was always very valuable. Thank you for the insights into your work on the PMR scheme, which have helped me many times while conducting this research.

Lastly, I want to acknowledge all my friends and my family. I especially want to thank Sarah Dudinski. Thank you for all the hours you have put into reviewing my written work, which very much improved the quality of my writing.

## TABLE OF CONTENTS

Page

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

# Introduction

This chapter illustrates the motivation for this study and subsequently defines the research goals and questions. Furthermore, the methodological approach that is used to answer these questions is explained.

## 1.1 Motivation

Saving weight has almost always been an important objective of the mechanical design process. Global warming and increased requirements on the performance of mechanical structures, however, have recently boosted the demands for extreme light weight structures. Light weight machines are often more fuel-efficient or have a greater power to weight ratio and, thus, comply with these new needs. Weight can either be saved by using new advanced materials or by optimizing the design. This design variation can be an optimization in shape, size, or topology.

Topology optimization (TO) has recently become more relevant due to the popularity and the improvements of additive manufacturing techniques [5]. Additive manufacturing provides the ability to produce the often complex structures generated by the TO schemes [6]. Hence, the fast growing additive manufacturing field makes topology optimization more practically applicable [7]. The relevance of topology optimization research is reflected in the number of recent publications, e.g., on TO with neural networks in 2021 [8] or on TO of composite materials in 2018 [9].

Novel topology optimization methods have been developed as the general trend advanced. Each of these methods has advantages and disadvantages that are continuously addressed and overcome by further improvements to the existing schemes or the development of new TO approaches. Taggart, Dewhurst, Dobrot,

and Gill [3] developed the Prescribed Material Redistribution (PMR) method for minimum compliance problems under a single volume constraint in 2008. Unlike some dedicated and more general mathematical optimization approaches, the PMR code solves these problems with a rather heuristic approach by imposing beta function-like density distributions to each design cycle.

The most recent, academic version of the PMR code is a standalone code that integrates the finite element analysis into the TO process. The advantages and disadvantages of the PMR algorithm have never been fully evaluated. It is, however, perceived that the standalone implementation is slow and non-intuitive. As a result, this study aims to increase the performance and the ease of use of the scheme. Since commercial TO optimization software has focused on certain methods [10], different approaches need to be compared to the existing standards. Hence, a benchmark with academic and commercial software is conducted to evaluate the potential commercial viability of the enhanced PMR scheme.

### 1.2 Goals of the Research

There are two major goals of this research. The first goal is to translate the current MATLAB version of the Prescribed Material Redistribution (PMR) algorithm into the programming language Python and enhance it. The MATLAB algorithm lacks performance as well ease of use, particularly in the model creation, the process of solving the finite element equations, and the post-processing. A potential optimization of these drawbacks is achieved by using Python in combination with the commercial finite element software Abaqus.

The second goal of this research is to benchmark the PMR method, both quantitatively and qualitatively. Thus, the heuristic PMR approach is compared to available commercial software based on the well-investigated topology optimization schemes SIMP and RAMP. By interpreting the benchmark results, the goal

of this comparison is to answer the following questions. What influence has the variation of model parameters on the results of the different optimization schemes? What are the advantages and disadvantages of the PMR method and the enhanced implementation in reference to older code versions and the other two methods? What are the implications of the benchmark results on future research and potential commercial implementation of the PMR approach?

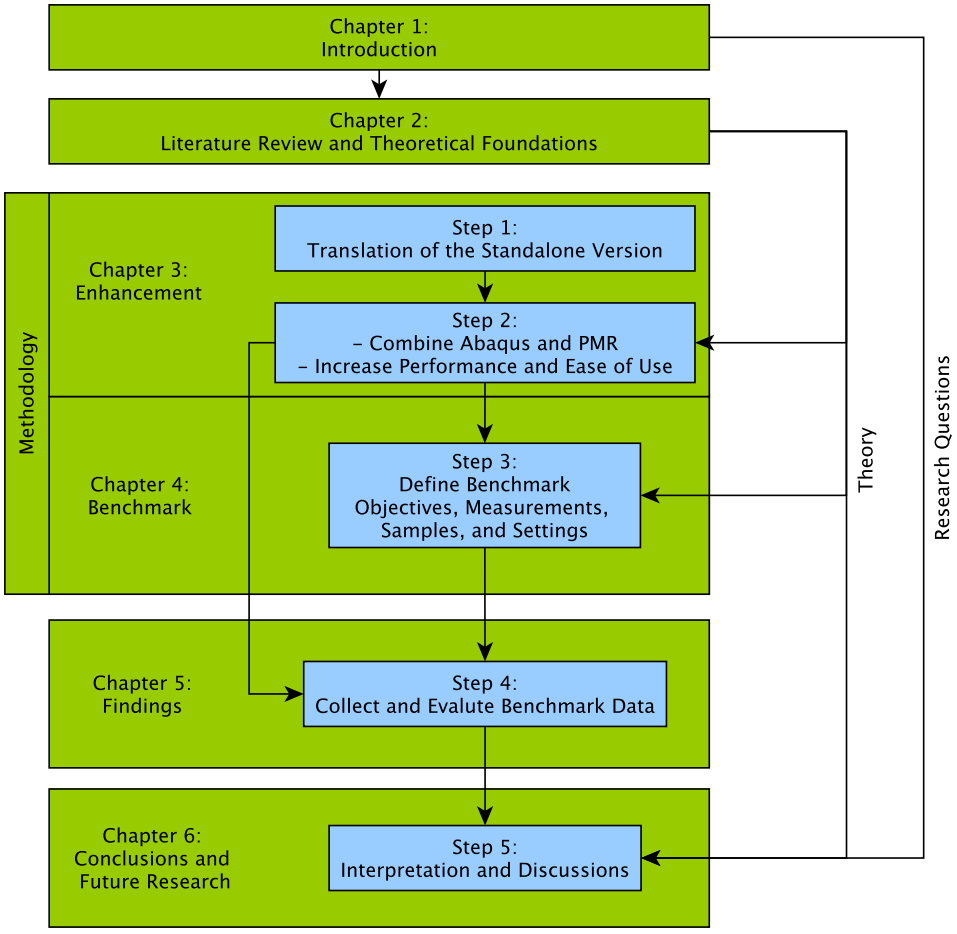


Figure 1: Flowchart of the Structure and the Methodological Approach

## 1.3 Methodology

The methodological approach of the research is divided into 5 steps. First, the MATLAB version of the PMR algorithm is translated into Python. Second,

Abaqus is then integrated into this code through the Abaqus Python interface and is utilized for a potential increase in performance and ease of use. Third, benchmark criteria and benchmark sample cases are derived from the literature and the particular benchmark settings for this research are defined. Fourth, the benchmark data is collected and evaluated based on the settings previously specified. Finally, the benchmark results are interpreted and discussed to answer the posted research questions. The structure of this thesis and the methodological approach is illustrated in Figure 1.

# CHAPTER 2

## Literature Review and Theoretical Foundations

The theoretical background of this research is explained in this chapter. First, both the basic principles of finite element analysis and topology optimization (TO) are summarized. Thereafter, selected TO schemes including the Prescribed Material Redistribution method are introduced in more detail. Finally, other TO benchmark studies and cases, the availability of TO software, and recent discussions within the TO research community are reviewed.

### 2.1 Finite Element Analysis

Finite Element Analysis (FEA) is a widespread and popular tool used in a variety of engineering disciplines. There are numerous publications and text books on the theory of FEA (e.g. [11, 12, 13, 14]). The comprehensive book by Bathe [12] will be used in this section as a reference for the fundamental theory and the corresponding equations. The reader is referred to any of these books for an extensive introduction because only a limited number of topics relevant to TO are discussed in this section.

The fundamental idea of finite element analysis is the approximation of a given real world design problem by a numerical solution. The continuous design domain is therefor split into discrete finite elements. A mathematical model describes the assembly of these coupled elements. This approach can be used to solve various physical problems such as fluid flow or heat transfer. In this research however, the focus is only on structural mechanics and static analysis in particular.

The finite element in the mechanical analysis generally consist of a set of nodes which can move in one-, two-, or three-dimensional space. The two-dimensional finite element meshes often contain either quadrilateral or triangular elements, the

three-dimensional models are constructed of hexahedral or tetrahedral elements. Figure 2 illustrates a simple rectangular 2D element. Boundary and loading conditions are imposed on the mesh of elements and the resulting system of equations is solved subsequently. The displacement of any point within the element is then interpolated using the nodal displacement values.

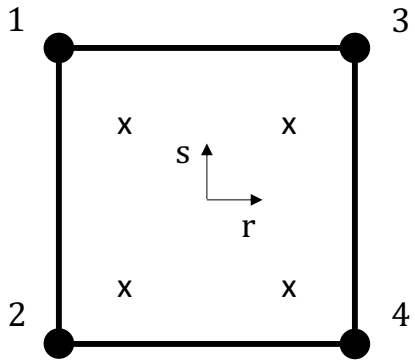


Figure 2: 4-node 2D Isoparamteric Continuum Element with 2x2 Integration

The interpolation or shape functions are commonly either linear or quadratic functions. In most two-dimensional analysis the quadrilateral elements have either four or eight nodes if linear or quadratic interpolation is used, and triangular elements have three or six nodes, respectively. There are, however, other element formulations with different number of nodes. The following equations are the interpolation functions for the 4-node 2D isoparametric continuum element:

$$h_1 = \frac{1}{4}(1-r)(1+s) \tag{1}$$

$$h_2 = \frac{1}{4}(1-r)(1-s) \tag{2}$$

$$h_3 = \frac{1}{4}(1+r)(1+s) \tag{3}$$

$$h_4 = \frac{1}{4}(1+r)(1-s) \tag{4}$$

The parameters $r$ and $s$ are the local coordinates within the isoparametric element and their values are in the interval from -1 to 1. The equations for quadratic

interpolation as well as the three-dimensional versions of these functions are given in [12]. These functions can be used to refine the analysis. Logan [11] describes two methods of refinement, the h- and the p-method. A designer can either use more and smaller elements in the FE mesh (h-method) or use higher-order polynomials for the interpolation function (p-method) to get more accurate results.

Finding a solution for the mechanical finite element equations requires solving multiple integrals. These integrals are often numerically solved utilizing Gauss quadrature. The Gauss integration approximates the exact solution of an integral of a function $f(x)$ by the weighted sum of values along $f$:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} \alpha_i f(x_i) \tag{5}$$

The optimal sampling weights $\alpha_i$ and the integration points $x_i$ are well-known and can be found in tables in the literature (e.g. [11, 12]). The integration order $n$ is a measurement for the accuracy of the approximation. Higher order integration is more exact. Equation 5 is valid for the one-dimensional case. However, the formula can be extended to two- or three-dimensional problems. Thus, for a two-dimensional element for example a 1x1, 2x2, 3x3, or higher Gauss integration could be used. Bathe [12] defines the term full integration as "the order that gives the exact matrices" and reduced integration as a lower order integration. For a linear four node quadrilateral element the full integration is of order 2x2 and for a quadratic eight node quadrilateral element this order is 3x3. Figure 2 shows the integration points for a 2x2 full integration marked as 'x'. The integration point for a reduced integration would lay in the center of the four-node element.

There are many software tools available to perform FEA for research and commercial purposes. The commercial tools Abaqus, ANSYS, COMSOL Multiphysics, LS-DYNA, and Nastran are among the most popular ones. In this particular study

Abaqus CAE [15] was used because of the availability of the research license. It can be assumed that a comparable study can be performed with any of these tools if the software offers a similar coding interface. The previously mentioned element types have specific internal names within Abaqus [16]. The four-node quadrilateral element in plane stress conditions is referred to as CPS4 and CPS4R (reduced integration) and the eight-node quadrilateral element is called CPS8 and CPS8R. The three- and six-node triangular elements are CPS3 and CPS6M.

## 2.2 Topology Optimization Schemes

Structural topology optimization (TO), as defined by Bulman, Sienz, and Hinton [17], is a means used by engineers to design a first practicable, structural topology in a reference domain for a certain given design problem. The goal of this optimization is the systematic and iterative redistribution of material inside the design domain. This process yields a structural topology which is defined as being optimal in some way. This tool is particularly important for the solution of complex problems. For easy problems, however, an experienced engineer or designer might be able to predict such an optimal topology [5]. Figure 3 illustrates a simple 2D design problem and an optimized solution (PMR algorithm) for a maximum stiffness/minimum compliance problem.

The fundamentals of today's structural topology optimization were established in a paper by Michell in 1904 [18]. Michell derived analytical solutions for minimum weight truss structure problems. Based on Michell's work, the first topology optimization schemes were developed. More recently, TO has experienced a rise of popularity because these original methods have become more sophisticated and new approaches have been developed [19]. An overview over the field's recent developments can be found, for example, in [19] and [20]. Simultaneously, the capacity of TO has been extended to other engineering disciplines and advanced

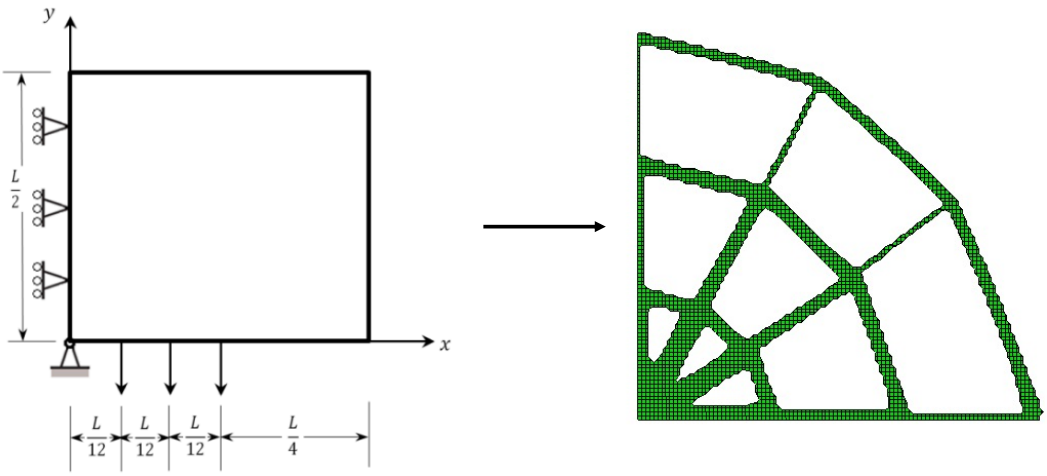


Figure 3: The Half Michell Arch Problem and an Optimized Solution

TO methods can, for example, deal with heat transfer, acoustics, fluid flows, or aeroelasticity problems [19]. Such problems including several physical phenomena are referred to as multi physics [21]. In this research only static, linear elasticity analysis with the objective of minimizing compliance under a single volume constraint will be investigated.

The general form of the minimum compliance problem with a single volume constraint can be formulated as follows:

$$\begin{aligned} min \quad & c = \mathbf{U}^{\mathbf{T}}\underline{\mathbf{K}}\mathbf{U} \\ s.t. \quad & \underline{\mathbf{K}}(\rho)\mathbf{U} = \mathbf{F} \\ & \int_{\Omega} \rho(\mathbf{x}) d\mathbf{x} = V_f \\ & \rho(\mathbf{x}) \in \{0,1\} \\ or \quad & \rho(\mathbf{x}) \in [0,1] \end{aligned} \tag{6}$$

The objective of this optimization model is to minimize the strain energy $c$, which is calculated from the displacement vector $\mathbf{U}$ and the global stiffness matrix $\underline{\mathbf{K}}$. The system is constrained by the load-displacement relationship "$\underline{\mathbf{K}}\mathbf{U} = \mathbf{F}$", the volume fraction $V_f$ constraint, and a binary or continuous definition of the

relative density $\rho$, which is the design variable. That is, material is allocated inside the domain $\Omega$ in a way that the overall structure has a minimum compliance or a maximum stiffness respectively. Simultaneously, the structure fills an exact fraction $V_f$ of the design domain's total volume.

The optimized topologies computed by TO schemes are often very complex structures. Some of these structures cannot be physically manufactured using traditional processes. Additive manufacturing, however, allows for the potential realization of structures with particularly high complexity [6]. Smaller limits for shape and complexity in additive manufacturing make these structures manufacturable. As a result, additive manufacturing as a growing field brings TO closer to application [7] and causes TO to gain more importance. However, the initially achieved solutions are not necessarily manufacturable [7] and often require manual interpretation [5]. Thus, there is a need for implementing manufacturing constraints, like build direction and distortion minimization, into TO software [5]. Manual changes to this initial design, like adding fillets, or further analysis are usually necessary for any (traditional) manufacturing technology.

The wide variety of topology optimization approaches available requires categorization of the different methods. Multiple studies that categorize TO schemes can be found in the literature. Bulman et al. [17] define three types of methods: h-methods, e-methods, and h/e-methods. Homogenization-type methods (h-methods) assume a homogeneous (artificial) material inside the domain that is iteratively redistributed. Evolutionary methods (e-methods) delete and add chunks of material (elements in a FEA) from and to the structure during the optimization process. The h/e-methods are hybrid methods combining approaches from both categories. Sigmund and Maute [22] differentiate between continuous methods, discrete methods, and Lagrangian and combined shape and topology approaches.

The discrete methods include the evolutionary schemes and the continuous methods include the density approaches, topological derivatives approaches, level set approaches, and the phase field approaches. The categorization used in this research follows a study by Deaton and Grandhi in 2014 [19]. Deaton and Grandhi divide the available TO schemes into 4 groups: 1) density-based methods including for example the popular SIMP and RAMP schemes, 2) hard-kill (evolutionary) methods, 3) boundary variation methods including level set and phase field approaches, and 4) a new biologically inspired method based on cellular division rules. Many of these methods are FE-based, which means that one or multiple finite elements can represent one discrete element in the topology optimization scheme [10]. In the following sections some of the most popular scheme are introduced in more detail.

### 2.2.1 Solid Isotropic Material with Penalization

The "Solid Isotropic Material with Penalization" (SIMP) method, first introduced by Bendsøe in 1989 [23], is probably the most widespread scheme to solve a topology optimization problem. It has been further investigated and named SIMP in a paper by Rozvany, Zouh, and Birker in 1992 [24].

The basic assumption of this method is that the naturally discrete integer optimization problem (void/dense regions) can be converted into a continuous problem where the (relative) density $\rho(\mathbf{x})$ of the structure is the design variable. This approach allows parts of the design domain $\Omega$ to have intermediate density values which can be interpreted as an "artificial material". That is, instead of using the binary density definition in eq. (6) the continuous definition is used (see also eq. (8) ("relax the zero-one constraints" [25])). All density-based TO schemes have this assumption in common. However, partially dense regions "cannot easily be manufactured" [22] and often discrete designs with either dense and void regions are

needed [21, 22]. For this reason, the SIMP scheme penalizes the material properties (in this case Young's modulus/stiffness tensor $\underline{\mathbf{E}}$) of intermediate density regions with a power law ($p \gg 1$) to force convergence to a quasi bimodal (void/dense) solution [23]:

$$\mathbf{E}_{ijkl}(\mathbf{x}) = |\rho(\mathbf{x})|^p * \overline{\mathbf{E}}_{ijkl} \tag{7}$$

where

$$V = \int_\Omega \rho(\mathbf{x}) d\mathbf{x}, \quad 0 \leq \rho(\mathbf{x}) \leq 1, \quad \mathbf{x} \in \Omega \tag{8}$$

This power law equation is specific to the SIMP formulation. The stiffness tensor $\overline{\mathbf{E}}_{ijkl}$ on the right side of Equation (7) contains the actual material properties, while the local stiffness tensor $\mathbf{E}_{ijkl}(\mathbf{x})$ for the artificial material is depending on the intermediate density $\rho$. Good convergence is achieved with the penalization parameter $p = 3$ [22]. The penalization equation is used to calculate $\underline{\mathbf{K}}$ in the load-displacement constraint in the general optimization problem (6). This problem is then often solved with the help of FEA and the "Method of Moving Asymptotes" (MMA), which was developed by Svanberg in 1987 [26]. Another way to penalize intermediate densities to obtain a quasi bimodal density field is introduced in the next section.

### 2.2.2 Rational Approximation of Material Properties

The "Rational Approximation of Material Properties" (RAMP) method is a density-based method, similar to the SIMP approach introduced in the previous section. It was proposed by Stolpe and Svanberg in 2001 as an alternative interpolation scheme for penalizing the intermediate densities [25]:

$$\mathbf{E}_{ijkl}(\mathbf{x}) = \frac{\rho(\mathbf{x})}{1 + q(1 - \rho(\mathbf{x}))} * \overline{\mathbf{E}}_{ijkl} \tag{9}$$

The RAMP formulation has a non-zero sensitivity in elements with a relative density of zero, which is an advantage over the SIMP formulation [19]. Figure 4 compares a linear material model, a material model using the SIMP, and a material model using the RAMP interpolation scheme. Similar figures can be found, for example, in [19], [27] or [25]. There are more approaches with different penalization laws in the literature. One example is the SINH method by Bruns, which is based on hyperbolic sine functions [27].

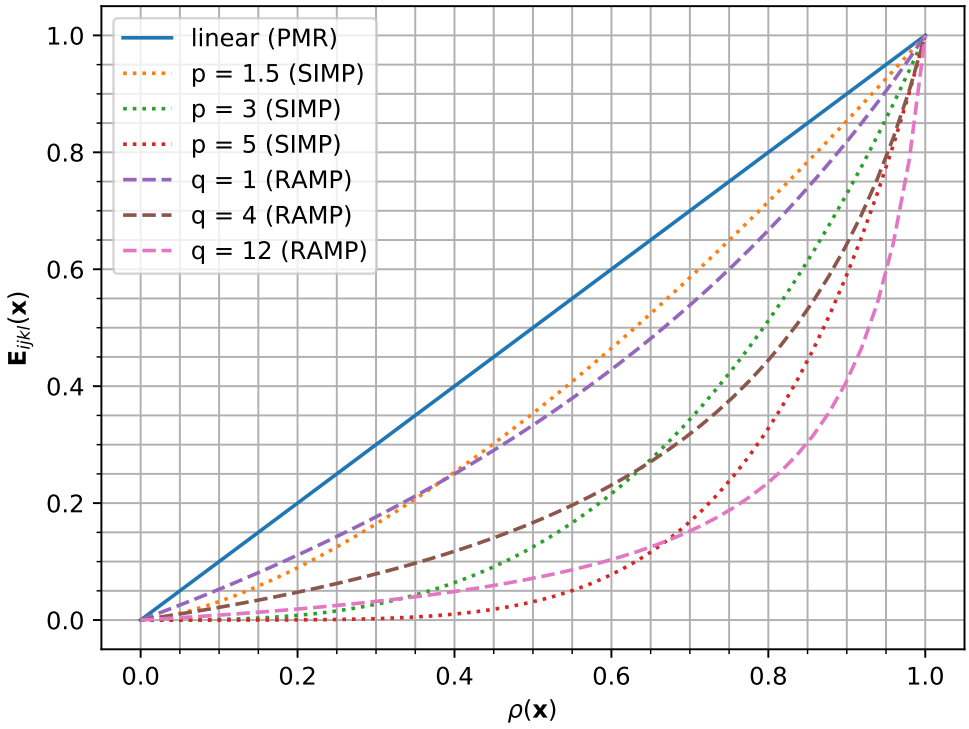


Figure 4: Penalization Laws for Intermediate Densities ($\overline{\mathbf{E}}_{ijkl} = 1$)

### 2.2.3 Evolutionary Structural Optimization

The "Evolutionary Structural Optimization" (ESO) method, developed by Xie and Steven in 1993 [28], follows a different approach than the density-based approaches introduced before. In this scheme, elements with a low criterion value

will be deleted from the design domain of the finite element analysis in an iterative manner. This criterion value is, for example, the strain energy density in case of the minimum compliance problem discussed in this research [19, 10]. The design variable is binary and states if an element exists (dense) or is deleted (void) (see first variable definition in eq. (6)). The bi-directional ESO (BESO) is an enhanced method in which elements can also be added to the structure, as described in an article by Querin, Steven, and Xie in 1998 [29].

Rozvany argues that the term "Evolutionary Structural Optimization" is inappropriate and suggests the term "Sequential Element Rejections and Admissions" (SERA) [10]. Sigmund and Maute argue that BESO should be perceived as a discrete update to the SIMP formulation (h-/e-methods) [22].

### 2.2.4 Level Set and Phase Field Topology Optimization

Unlike density-based and evolutionary-type methods, the fairly new level set and the phase field schemes do not change the topology on an elemental level, but rather vary the boundaries of the structure. The level set method was developed for a different purpose by Osher and Sethian in 1988 [30] and later utilized for TO by Wang, Wang and Guo in 2003 [31]. The topology of the structure in this method is described by a scalar function $\Phi(\mathbf{x})$, the level set function. The structure is represented by the positive values of $\Phi(\mathbf{x})$ and its boundary is defined by $\Phi(\mathbf{x}) = 0$. The boundary of the structure in the phase field approach is described by the transition region between two Phases A and B defined by a phase field function over the design domain [19]. A more detailed description of these methods, which are recently getting more popular, can exemplary be found in [31, 32, 19, 22].

### 2.2.5 Common Problems

There are a number of problems that commonly occur with (FE-based) topology optimization schemes. Shukla, Misra, and Kumar [33] name "checkerboard patterns, [...] mesh dependency, [...] local minima, and [...] singular topologies (for stress constrained problems)" as the most typical issues. Guest, Prévost, and Belytschko [34] similarly discuss checkerboarding, non-existence of solutions, and mesh dependency. Checkerboarding and mesh dependency are of particular interest for the benchmark study in this research.

Figure 5: Checkerboarding Example

The problem of checkerboarding is a discretization issue associated with the FEA model [10] and is named after the board game. When checkerboarding occurs, high- and low-density regions (elements) are positioned next to each other in a periodic order that resembles the checkerboard. These patterns are unintended, show an overestimated stiffness, and are hardly manufacturable [34]. Figure 5 shows a SIMP result of the Half Michell Arch example problem with checkerboard which was solved with the 88 line MATLAB code (see sections 2.5 and 5.1.3) [35]. One method of preventing checkerboarding is the use of higher order elements [33, 36].

## 2.3 Prescribed Material Redistribution

The "Prescribed Material Redistribution" (PMR) method was developed by Taggart, Dewhurst, Dobrot, and Gill at the University of Rhode Island in 2008 [3]. The scheme has been validated for two- and three-dimensional problems [1, 3] and has been patented [37]. The PMR algorithm has first been implemented as a user subroutine in the commercial FEA software Abaqus [3]. A more recent version is solely written in MATLAB with a simple FE solver. The PMR method is currently not available in commercial software.

The PMR scheme can be categorized as a density- and FE-based topology optimization scheme for compliance minimization under a single volume constraint. Unlike the SIMP or RAMP formulations, which facilitate the solution of an optimization problem, the PMR scheme is a fully heuristic method that imposes a specific density distribution for every iteration step to obtain a minimum weight structure with maximized stiffness. This prescribed distribution is based on a family of Beta (probability) functions [3].

Initially, a volume fraction is specified for the design domain. That is, it is defined how much volume of this domain should be dense (the actual structure) and how much volume should be void after the final iteration. Moreover, the loading and boundary condition are imposed on the domain to completely define the design problem. This can either be a 2D or a 3D problem.

The initial density distribution is unimodal partially-dense. Thus, all points in the design domain have the same relative density, which is equal to the volume fraction defined previously. This distribution is now gradually and smoothly transitioned to a bimodal distribution where a specific proportion of the design domain is void and the rest is fully dense according to the volume fraction. The

total mass of the material is conserved in this transition. The incomplete Beta function provides this described transition [1]:

$$F(\beta) = \beta_{inc}(\rho, r, s) = \frac{1}{B(r,s)} \int_0^{\rho} (\rho')^{r-1}(1-\rho')^{s-1} d\rho' \tag{10}$$

Where $\rho$ is the density and $B(r,s)$ is the Beta function which is computed using Gamma functions $\Gamma$:

$$B(r,s) = \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)} \tag{11}$$

The incomplete Beta function represents the cumulative distribution of the relative density. The distribution's parameters $r$ and $s$ are specifically chosen to ensure the smooth transition (not be confused with local finite element coordinates $r$ and $s$). Additionally, an artificial, non-dimensional time $t$ is introduced to parameterize the iterations. Figure 6 illustrates the transition from the initial, unimodal ($t = 0$) to the final, bimodal ($t = 1$) distribution of relative material density by showing a series of Beta distributions. The graph on the left shows the probability density functions and the graph on the right the cumulative distribution function.

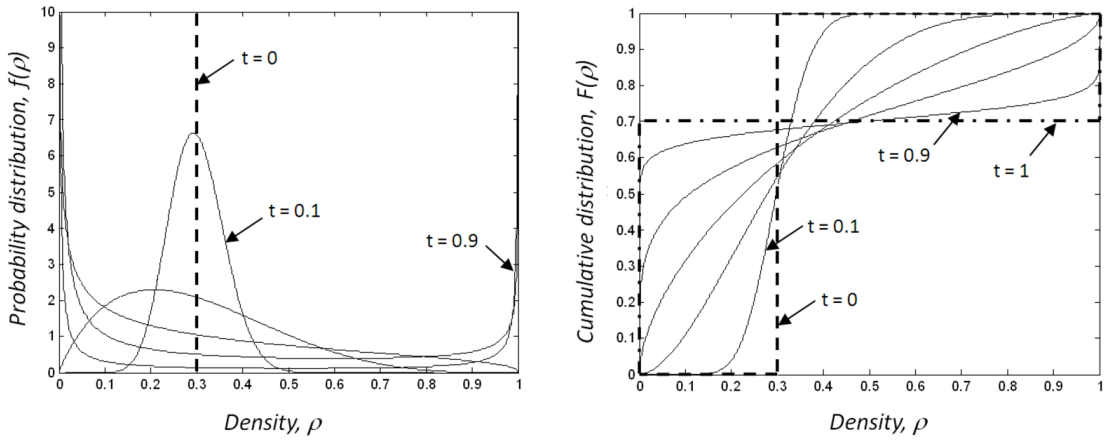


Figure 6: PMR Density Distribution using Beta Functions [1]

The density distribution is imposed on each node in the design domain in the descending order of its strain energy density value. That is, the strain energy values of each node, which result from the FE analysis, are sorted and the node with the highest strain energy is assigned the highest density value. Because a linear relation between stiffness and density is assumed, a higher density means that the node is stiffer. Thus, nodes with a higher strain energy in the FE analysis are stiffened and vice versa. A minimum density ($\rho_{min} = 10^{-2}$) is defined to prevent numerical singularities in the system matrices. This definition is also common practice for other TO approaches [22].

The PMR scheme has similarities to other TO approaches. Since the compliance problem is quite simple, there is always an increase in stiffness if material is added (see section 2.6). However, within all approaches utilizing this fact there are more efficient and less efficient methods [22]. The PMR method has also some similarity to a scheme developed by Guest et al. in 2004 [34]. This approach uses mesh independent nodal projection functions to impose minimum length constraints onto a TO scheme. The projection function used in the PMR method simply averages the nodal densities for calculating the elemental stiffness and, thus, is mesh dependent (see section 6.3).

## 2.4 Benchmarks of Topology Optimization Schemes

Various benchmark studies in the literature investigate the different methods introduced previously while focusing on separate objectives. Valdez et al. [38] carry out a meta study including 103 articles to merge the different 2D benchmark characteristics, like geometry or loading conditions, into unified problems. Subsequently, a benchmark using two versions of SIMP is conducted and solutions for minimum compliance problems and minimum volume problems with a stress constraint are presented. Other studies compare not only different SIMP schemes but

include also other density-based, evolutionary, or boundary variation approaches. Sigmund and Maute [22] claim that most of these TO methods, however, are very similar and are based on an almost identical sensitivity idea. Rojas-Labanda and Stolpe [39] choose another focus and benchmark different solution algorithms, for example the MMA scheme. There is no benchmark study including the PMR method known to the author.

The varied parameter, the criteria, and the sample case chosen in that studies are of particular interest to the benchmark in this research. A benchmark study by Bulman et al. [17] investigates the effects of FE mesh density, FE mesh distortion, and non-uniform initial density distributions on the TO results. These effects are measured qualitatively and quantitatively using the normalized strain energy. The influences of the mesh density and the mesh distortion as well as some parameters generic to the specific schemes (e.g. number of iterations) are investigated in this research. Additionally, four different standard benchmark cases are described in that study: clamped deep beam, Michell truss, Messerschmitt-Bölkow-Blohm (MBB) beam, and short cantilever beam. However, Sigmund and Maute [22] claim that there are only two standard benchmark cases: the MBB beam and the 5 by 8 cantilever. There are many more studies specifying different or new cases, e.g. [32] or [38]. Nonetheless, Deaton and Grandhi [19] state that analytical solutions can be found for particular examples and that these cases should generally be used. A series of these exact solution is, for example, given in [4, 40, 41, 42]. In section 4.3 the different example cases for this research are chosen based on this overview, are visually illustrated, and are defined in detail.

## 2.5 Available Topology Optimization Software

There is a variety of topology optimization software available for either research or commercial use. Reddy et al. [5] provide a list of both types of TO

software, which includes the commercial software packages “Altair OptiStruct”, “MSC Nastran”, “Vanderplaats Genesis”, and “SIMULIA Tosca”. The latter is used in this research because the software is implemented in the same commercial FEA software used for the PMR scheme (Abaqus). SIMULIA Tosca Structure's current version applies a sensitivity-based approach (SIMP or RAMP) and a MMA algorithm [43]. In earlier versions, Tosca might have been the only commercial software to use an evolutionary-type algorithm [10]. The software still offers a non-default option in the settings to delete soft elements in the manner of a hybrid method (see section 2.2.3).

Open access research software, which has been published for academic purposes, is also used in this benchmark study. Sigmund issued a simple 99-ine MATLAB code using the SIMP scheme for minimizing compliance problems in 2001 and the TopOpt group maintains updated versions available online for academic purposes [44, 35, 45]. Andreassen et al. [35] claim that the new, efficient 88-line code can solve an example problem with 7500 elements 100 times faster the the original 99-line version. These results are achieved with loop vectorization and memory preallocation. Challis [32] provided an 129-line level set MATLAB code in analogy to these SIMP versions in 2010. The TopOpt group also published an open source level set MATLAB algorithm more recently in 2020 [46].

## 2.6 Discussion within the Structural Optimization Community

There seems to be a division into separated groups within the topology optimization research community in which either SIMP, level set, or BESO schemes are preferred [22]. An illustrating example of this issue is given in an article by Rozany [10]. Rozvany names both, criticisms of ESO approaches and “attempts to defend ESO from the [...] criticisms”. However, Rozvany finally concludes that evolutionary approaches are heuristic, inefficient, and unreliable. Sigmund and Maute

[22] review the criticism that density-based schemes sometimes yield final results that still include intermediate densities. They argue that it is common practice to set a density threshold, when interpreting the design before manufacturing. A completely different criticism is raised by Reddy et al. [5], who criticize the lack of suitable manufacturing constraint in present commercial TO software.

One discussion within the community, which is particularly important to this research, is about how and what TO research is conducted. Sigmund and Maute [22] criticize visual assessments and the use of the word "optimal" if the convexity of the problem is not validated. The PMR method is a heuristic approach and, thus, it is reasonable in this research to use the terms "optimized" or "final result". Sigmund and Maute [22] emphasize the need for quantitative comparisons, when testing algorithms, and demand detailed disclosure of the benchmark settings and results. Moreover, Deaton and Grandhi [19] criticize authors that only test their new schemes on minimum compliance problems and not on more broad and practical frameworks ("multi physics" and manufacturing constraints). The compliance problem is very simple, all sensitivities are always negative, and any approach which adds material in some way will obtain a stiffer structure [19, 22]. Sigmund and Maute [22] state that they "feel that using rudimentary optimization algorithms is the wrong direction to take". For this reason, Sigmund and Maute propose a list of nine challenges for current TO research at the end of their article: efficiency, general applicability, multiple constraints, complex boundary conditions, independence on starting guess, few tuning parameters, mesh-independent convergence, ease of use, and alternatives to finite element analysis. The PMR version developed and benchmarked in this research was compared to each challenge on this list in the final discussion (see section 6.1).

# CHAPTER 3

# Enhancement

This chapter summarizes the first two steps of the methodological approach. First, the standalone MATLAB version of the PMR algorithm was translated into a standalone Python version. Second, Abaqus was integrated in this code and was utilized for a potential increase in performance and ease of use. In the following, the term "strain energy" is used in two different ways. One the one hand, strain energy refers to the nodal values of elastic strain energy density (per unit volume) and, on the other hand, the term stands for the total strain energy of the system.

## 3.1 Standalone PMR Algorithms

The initial MATLAB code, as well as the first translated Python version, are standalone PMR algorithms. That is, running these programs does not require additional software and corresponding licenses because the FE solver is integrated in the algorithm. This approach's advantages and disadvantages found before, during, and after the translation process are discussed in this section.

First, the MATLAB algorithm was analyzed to determine the general structure and to understand the fundamental functionality and its limitations. The integrated FE solver was of particular interest, because it was initially assumed to cause a poor performance and limit the design options. The analysis revealed that the solver mainly consists of assembling the global stiffness matrix, solving the stiffness equation ($\underline{\mathbf{K}}\mathbf{U} = \mathbf{F}$), and computing the strain energy. The scheme also exclusively uses 2x2 Gauss integration (2x2x2 for 3D) to evaluate the integrals. Furthermore, only four-node quadratic-shaped (2D) and eight-node cubic-shaped (3D) linear elements with the unitless edge length of 1 are available.

The limited pre- and post-processing options are another limitation of the standalone version. The pre-processing is limited to the definition of the design domain by the number of elements in $x$, $y$, and $z$ (for 3D) directions. Hence, the standalone scheme solely allows for rectangular design domains. Also, the loading and the boundary conditions have to be applied on the specific degree of freedom by defining the corresponding vectors directly. For 2D models, the post-processing options include stress, total strain energy, and density plots, visual feedback, and an export option. The visual feedback shown each iteration illustrates the current density distributions as a gray scale contour plot and provides information about the respective iteration and the objective function value. Following the main part of the algorithm, the result can be export as a STL file, which is a file format used in additive manufacturing [7].

After the analysis of the algorithm, the requirements for the Python version were defined and the algorithm was translated. The main requirement was to keep all features except the STL file export since the STL file was not relevant to the benchmark. The standalone Python version was solely an intermediate step to the final version with embedded FEA software that could potentially handle different elements and arbitrarily shaped design domains. Hence, there was no need to translate the export function. The process of translating the other functions was facilitated by the use of the Python package NumPy and SciPy [47, 48]. These packages have a high number of syntactic similarities to MATLAB code and provide high performance since they are written in C.

The standalone MATLAB and the corresponding Python version have the same structure. Flowchart 7 shows the essential processes of these algorithms. The general PMR method is explained in more detail in section 2.3. The inputs to the algorithm are the volume fraction, the number of iterations, and the problem

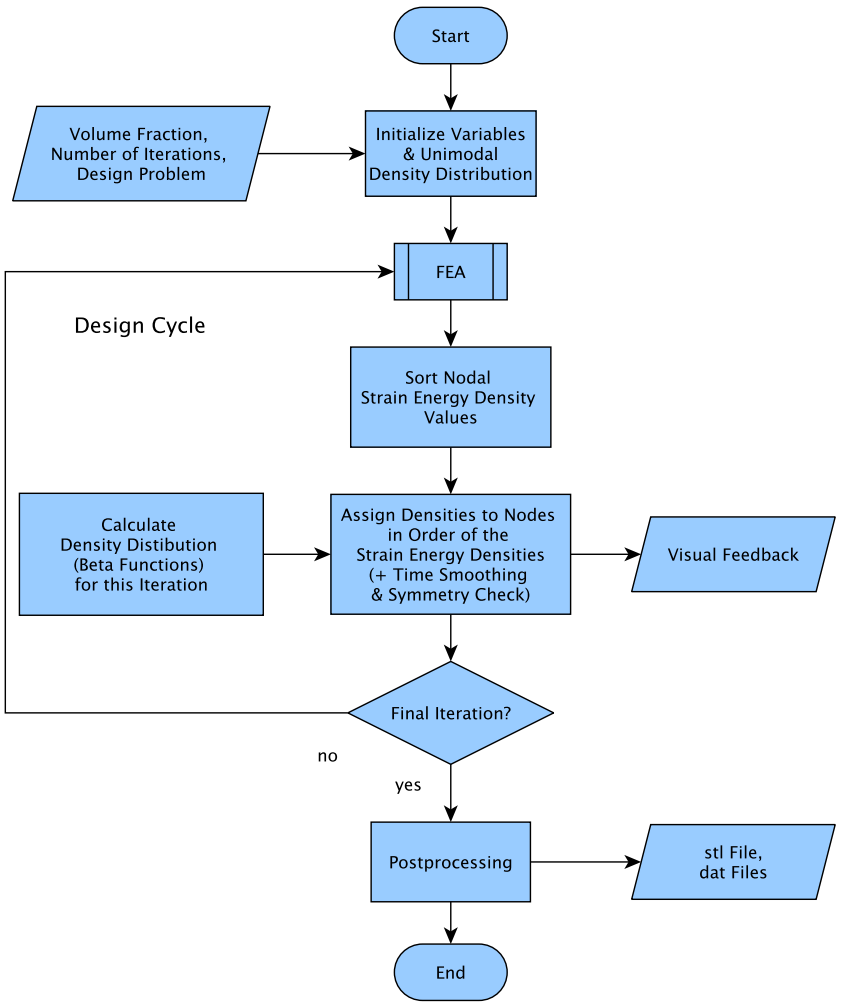


Figure 7: Structure of the Standalone PMR Versions

definition, including domain/mesh, loads, and boundary conditions. After initializing all variables and assigning each node the same partial density, the main design cycles start. One design cycle consists of an FE analysis, sorting the resulting nodal strain energy densities, calculating the beta distributions, and assigning densities to nodes based on the strain energies. The nodal densities are averaged for each element and the elemental stiffness is calculated with the linear material model. Additionally, each design cycle gives visual feedback and optional filtering and symmetry corrections are applied. The symmetry check, if applied, equates the

densities of nodes with exact same strain energy density values and, thus, corrects unsymmetrical density distributions caused by the computational discretization of the beta function. The time smoothing filtering combines the density distribution of the current design cycle/ iteration with the previous two distributions:

$$\rho'_t = \rho_{t-1} + \frac{(\rho_t - \rho_{t-2})}{2} \tag{12}$$

If the time smoothing is disabled, the algorithm was found to give erroneous results and the solutions start to oscillated at some point. Figure 8 illustrates the oscillations in two ways. The left illustration shows the density distribution at the 200th of 250 iterations for the Half Michell Arch Case (see section 4.3). The structure oscillates in multiple modes. One mode is the transverse wave in the trusses. Another oscillation is the radial emission of material chunks, which can be seen on the right side of the density plot. The right illustrations plots the strain energy of the system over the iteration history. The values start to oscillate at about 150 iterations. The time smoothing was enabled for all problems in the benchmark, after this effect was found multiple times.

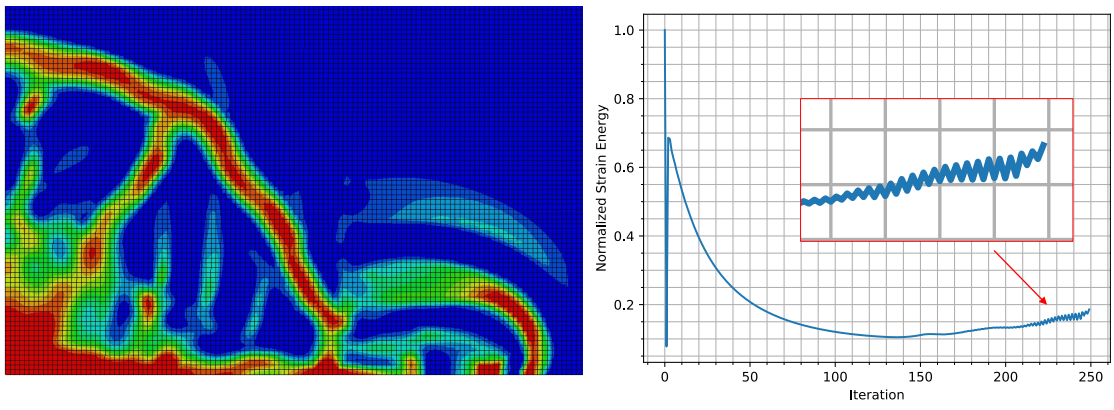


Figure 8: Oscillations when Time Smoothing is Disabled

The analysis of the PMR algorithm structure showed that the actual FE analysis can be easily decoupled from the other processes in the design cycle. This

decoupling was the starting point for the second step, which aimed to perform the computationally expensive FE analysis with a high-performance solver outside the implemented PMR code. Additionally, the Abaqus Python interface provides not only a more efficient FEA solver, but also an interactive graphical user interface (GUI) to potentially increase the ease of use in the pre- and post-processing steps.

### 3.2 Embedding the Commercial Finite Element Solver

The second step in the methodological approach was to combine the commercial FEA software Abaqus and the Python PMR scheme. Abaqus offers an application programming interface (API), "Abaqus Scripting", using Python as a programming language [49]. A Python script can then be run from within the interactive GUI Abaqus CAE [15]. However, there are limited options to call Abaqus from an external Python file. Additionally, Abaqus CAE offers comprehensive, visual, and interactive pre- and post-processing. For this reason, the new PMR algorithm was developed to be called from inside Abaqus CAE. The script reads, writes, and manipulates the Abaqus files in a way that pre- and post-processing can separately be done in Abaqus CAE.

Figure 9 shows a simplified flowchart of the newly developed algorithm structure. Processes performed in Python have a blue symbol and those that are using Abaqus are orange. By using Abaqus CAE, the pre- and post-processes became more clearly separated from the actual main part with the iterative design cycles. The interface between the Python script and the Abaqus parts was established by reading and writing files. The Python scripts writes and manipulates the INP files that are read by the FEA solver. The solver, subsequently, outputs an ODB file that is read by the Python script again. The script then writes the density distributions and the strain energy history into the ODB file.

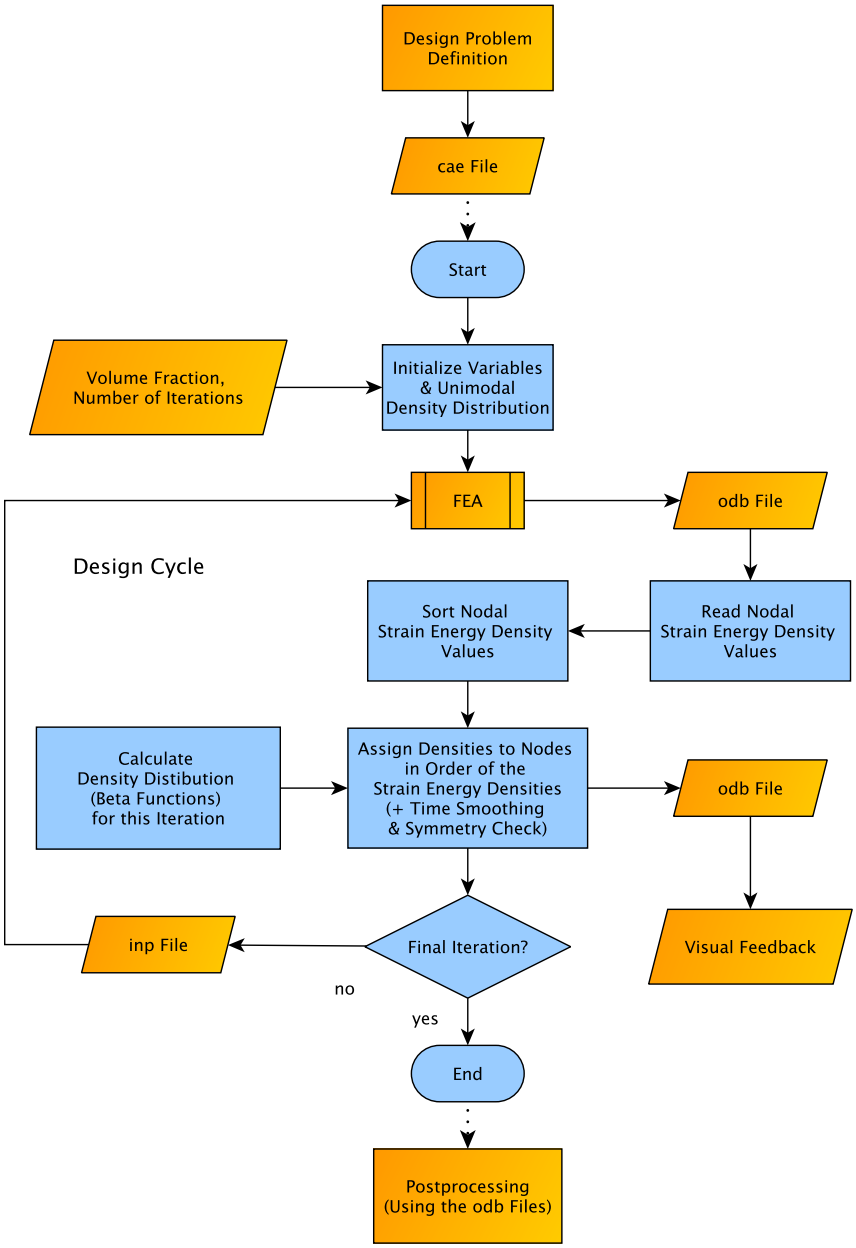


Figure 9: Structure of the Abaqus-Python PMR Version

The main challenge in the implementation of the method was the design of these interfaces mentioned previously. The particular problem was that Abaqus CAE does not provide options to define a local material in the way needed for the PMR scheme. The problem was overcome by manipulating the INP files instead of the model database. This manipulation is used in a similar manner by another

graduate student, who is currently using MATLAB to interface with Abaqus. A distribution table containing each individual elemental stiffness value is appended to the initial INP file. This INP file is then run as an Abaqus job.

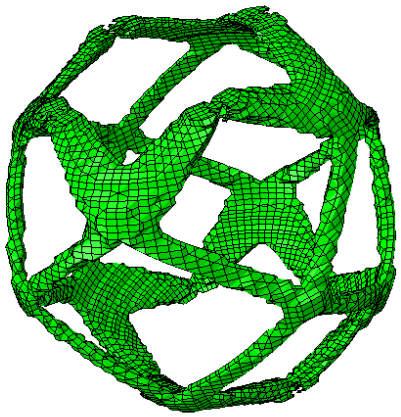

Figure 10: PMR Result for the 3D Loxodrome Case

The example cases used in the development process were two dimensional because these cases are less computationally expensive. Finally, it was verified that the exact same code was also applicable to 3D cases. Figure 10 shows a PMR result of the Loxodrome Torsion Case (see [3] and Figure A.58) with roughly 150,000 linear elements.

### 3.3 Performance and Ease of Use Improvements

The second methodological step did not only include the connection Abaqus and Python code, but also the realization of a potential increase in speed and ease of use. The question "How to make it faster?" affected primarily the iterative main part of the algorithm. The question "How to make it easier to use?", however, was focused on the pre- and post-processing steps.

The limitations of the standalone versions shown in section 3.1 defined the major requirements for an increase in ease of use. Hence, a greater variety of features and more design options are also evaluated as an increase in ease of use

in the following section. The first enhancement was achieved by using the interactive Abaqus GUI. The GUI automatically simplified and visualized the design process in the pre-processing stage. The GUI also facilitated executing the PMR algorithm. A pop-up window prompts the user to enter the two PMR parameters, volume fraction and number of iterations, while the other design information is automatically transferred from the model database to the Python code. Utilizing this Abaqus model database allowed for using arbitrarily-shaped design domains and a mesh that is independent from that domain. This mesh could contain a variety of new element formulations. That is, triangular and tetrahedral element shapes, mixed shape meshes, locally refined meshes, axisymmetric elements, linear and quadratic interpolation, as well as reduced and full integration could be handled by the algorithm. Different sized elements (e.g., in locally refined meshes) were made possible by using the Abaqus output variable "SENER" which measures elastic strain energy density per unit volume. The elemental volume in the standalone versions was always equal to one.

Additionally, more complex loading and boundary conditions, such as, pressure loading, could be defined more easily. All these new features were tested. However, there might also be more and potentially nonlinear design options within the general static step that could be applied to the design problem and were not tested, such as, multiple part interaction and friction. There are, nevertheless, limitations to the pre-processing options. Dynamic analysis as well as non-mechanical loading and boundary conditions (e.g., heat transfer) cannot be used since this formulation of the PMR scheme exclusively applies a static analysis in a minimum compliance problem.

The post-processing options of the standalone version were copied and enhanced in the integrated PMR version. That is, the visual feedback now uses

the Abaqus viewport to illustrate the density distribution in the design domain. Unlike the other schemes that converge, the PMR method has a fixed number of iterations. Thus, a duration that is still remaining to finish the scheme can be linearly estimated fairly precisely. This time prediction was additionally implemented in the visual feedback. This feature is possibly advantageous to designers because they can schedule large, complex problems better with the heuristic PMR approach than with any converging method. After the main part of the algorithm, the files get saved separately to prevent accidentally overwriting the data. All default mechanical field quantities, like displacement, strain, and stress can be found in the files. The density distribution history was also written into the ODB files. With this capability, the optimized structure can be cut out the design domain by using the "view cut" function and a certain intermediate density threshold (common practice see 2.6). This structure can then be exported as an OBJ file. This file format can either be 3D-printed directly or converted to a STL file first and then printed.

The main body of the algorithm is primarily responsible for the total duration of the scheme. Figure 11 illustrates the ratios of different functions' durations in a single design cycle of the most efficient algorithm version. The computer specifications can be found in chapter 4. The example case used was a Half Michell Arch problem (see section 4.3) with 40,000 four-node linear elements with 75 iterations in total. The total duration of one design cycle was about 20.4 seconds. Roughly 85% of the iteration time was spent with the execution of the FE analysis job. This duration could not be optimized within the Python code because the calculation is completely performed by Abaqus. The Abaqus LOG files were analyzed to determine the duration of the individual steps of an Abaqus/Standard job execution. The total measured job execution took about 17.5 seconds, of which 6 seconds

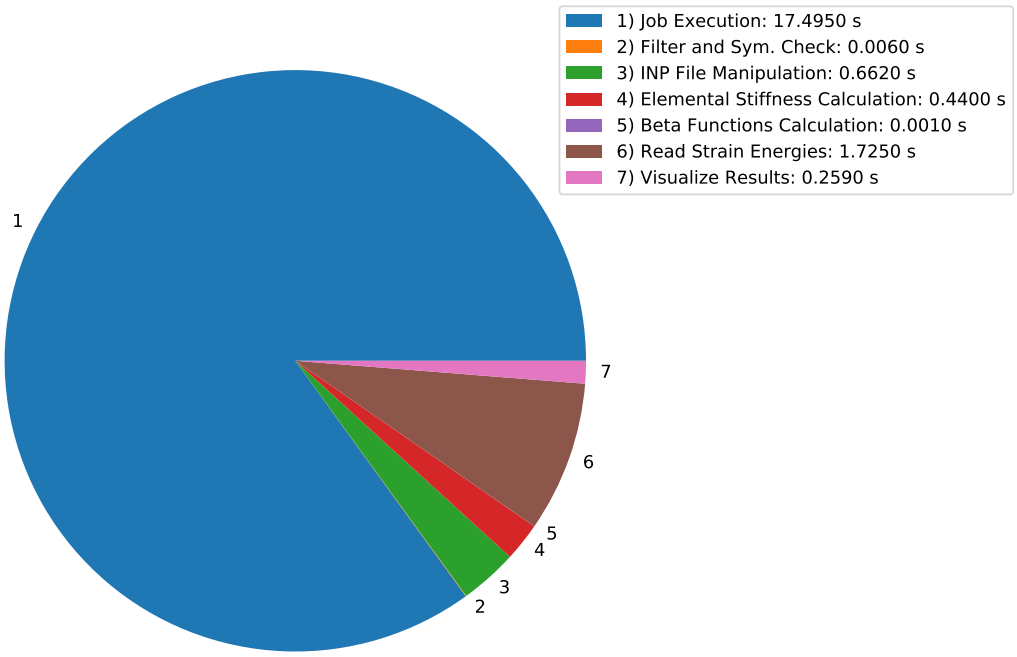


Figure 11: PMR Design Cycle Durations (Half Michell Arch, 40k elements)

were spent for the Input File Processor, 4 seconds for the actual Abaqus/Standard FE analysis, and 2 seconds to complete the job. The sums of these processes left 5.5 seconds for the license check, which was not reported in the LOG file. The comparison to the execution of Tosca jobs showed that these jobs did not seem to need the Input File Processor and the license check, which cuts down the duration significantly. Different approaches to make use of this advantage with the PMR algorithm were evaluated. However, there was no method found to execute the FEA in the PMR scheme without checking the license and pre-processing in every iteration using the Python interface.

Table 1: Comparison of the Old and New Symmetry Check Functions

Old Symmetry Check Algorithm

```
import numpy as np

def check_sym(U, x):
    # find nodes with non - unique strain energies
    U_siz = np.shape(U)
    isame = -1*np.ones((U_siz[0], 1))
    for i in range(0, U_siz[0]):
        for j in range(0, U_siz[0]):
            error = abs((U[i] - U[j]) / U[i])
            if (error < 1e-4) and (i != j):
                isame[j] = i

    for i in range(0, U_siz[0]):
        for j in range(0, U_siz[0]):
            if isame[j] == i:
                x[j] = x[i]
    return x
```

New Symmetry Check Algorithm

```
import numpy as np

def check_sym(U, x):
    # find nodes with non - unique strain energies
    values, index, count = np.unique(U, return_counts=True, return_index=True)
    index2 = index[count > 1]

    for i in index2:
        y = np.argwhere(U[:, 0] == U[i])
        x[y] = float(np.mean(x[y]))
    return x
```

The other functions were primarily limited in performance by Python, not by Abaqus. These limitations were most likely caused by Python's poor performance with nested loops [50], which were repeatedly used in the script. Thus, loop vectorization was applied, which also was one method for increased performance in the academic 88-line SIMP code [35] (see section 2.5). The two greatest improvements in speed were made in the function that reads the strain energy densities and in the symmetry check, which both scale significantly with the size of the arrays. Table 1 compares the old symmetry check function and the new vectorized form. Both algorithms compare each nodal strain energy density value to all the other values. If two nodes have the exact same strain energy, they are assigned the exact

same density. The new algorithm took 0.0156 seconds to run in an example with a randomized 10,000 by 1 strain energy vector, the old version needed 138.89 seconds. Hence, the newly developed scheme is four orders of magnitudes faster. The test was conducted multiple times and the same result or an even bigger difference could be observed every run. This performance boost made the duration of one of the initially slowest functions in the scheme negligibly short. The new version has an additional advantage because it conserves mass by taking the average density, while the old version did not. The difference between the old solution and a mass conserving solution is, however, negligible, because a new mass conserving density distribution, which is independent from the symmetry check, is imposed on the system in the beginning of every iteration.

### 3.4 Integration Point-based Density Distribution

A completely different approach to the PMR method is introduced in this section, which was not part of the planned methodological concept. The original idea for this new scheme was to eliminate a computationally expensive loop from the algorithm by imposing the density distributions on the integration points instead of the nodes. The integration points are the points at which Abaqus outputs the strain energy density values by default, while a nodal output requires one extra step. Additionally, there are more integration points in a mesh than nodes when using full integration. For example, four node linear elements have four integration points and four nodes, but the adjacent elements in a mesh share their nodes. The integration points, however, lay inside the elements. Thus, a potential increase in accuracy and resolution with a potentially increased performance was assumed for this approach.

The similarity to the initial approach simplified the implementation of this scheme. Hence, only minor changes had to be made to the algorithm. After

the implementation, the integration point-based PMR method was tested on 2D cases. The most important testing parameter was the dependency on the element formulation (linear vs. quadratic interpolation and reduced vs. full integration). The results can be found in section 5.1.2

# CHAPTER 4

# Benchmark

This chapter discusses the third methodological step, which is setting up the benchmark. The sections review what was measured (objectives), how it was measured (measurements), and in what framework the benchmark was conducted (cases and settings). The main benchmark included three methods: the new Abaqus Python PMR version, the SIMP, and the RAMP implementation in SIMULIA Tosca Structure. The following chapter concentrates on these three methods. However, additional cases were run to compare the standalone versions with the research MATLAB codes of the SIMP and level set schemes (see section 2.5).

## 4.1 Benchmark Objectives and Hypotheses

The benchmark was focused on an overall comparison of the PMR, SIMP, and RAMP schemes and on the effects of varying specific parameters. These parameters were the number of iterations, the volume fraction, the mesh density, the element formulation, and the mesh distortion. Bulman et al. [17] also investigated the effects of non-uniform initial density distributions (see section 2.4). These effects were not part of this benchmark, because the PMR scheme requires an unimodal initial density distribution.

### 4.1.1 Overall Evaluation

The first objective of this benchmark was to determine if the chosen measurements, example cases, and settings are reasonable. That is, the general sampling and measuring needed to be statistically valid and reliable. Second, the benchmark test cases had to be tested on typical TO issues. The most common problems of topology optimization were introduced in section 2.2.5. The two problems checker-

boarding and mesh dependency were particularly interesting to this benchmark. For these reasons, the following questions were posed:

- Do the test cases show checkerboarding patterns?
- Do the test cases show mesh dependency?

The primary objective of the overall evaluation was to compare the different methods and the sample cases. The goal of the comparison was to obtain relative advantages and disadvantages of the methods and sample cases. For these reasons, the following questions were formulated:

- What are the visible differences between the three algorithms (e.g., in the way they converge)?
- Is one method faster than the others?
- Does one method give better results than the others?
- Does one sample case give better results than the others?
- Is one sample case better suited for one particular method?

The following sections reviews more detailed benchmark hypotheses based on the variation of single parameters.

### 4.1.2 Number of Iterations

The number of iterations is explicitly specified before starting the PMR scheme. However, the total number of iterations needed in the SIMP and RAMP formulation is implicitly determined by the convergence criteria and differs for each problem. Hence, PMR and the other two methods required two different analyses. On the one hand, a predictive hypothesis was formulated for the effects of varying

the iteration number for the PMR scheme. On the other hand, a descriptive analysis was set up to investigate potential patterns in the required number of iterations when varying different other parameters of the SIMP and RAMP schemes.

The PMR algorithm calculates as many beta functions as there are iterations specified. Thus, the transition from the unimodal to the bimodal density distribution is smoother for a greater number of iterations. It was anticipated that this smoother transition would lead to a better result because the incremental changes in the density distribution are smaller. For this reason, the following hypothesis was formulated:

**Hypothesis 1** *A better result will be obtained with the PMR scheme if a higher number of iterations is used.*

This hypothesis was tested for 10, 25, 50, 75, 100, 150, and 250 iterations. One goal of this test was to determine a generally reasonable number of iterations that balances the overall duration with the obtained solution quality of the scheme. A designer could obtain a first practical result by using this number and refine the result by specifying more iterations. It needs to be mentioned that this hypothesis is directional and, thus, was not analyzed in the same wording as formulated. Instead, it was tried to reject the non-directional null hypothesis that all the mean evaluations are statistically identical. Other hypotheses in this chapter were analyzed analogously.

### 4.1.3 Volume Fraction

The volume fraction is a parameter that rather defines the problem than changes the settings of the algorithm. Hence, the effects of this parameter needed to be analyzed from two different perspectives. First, adding material makes a structure stiffer (see section 2.6). Thus, it was expected that higher volume fractions lead to stiffer results for all three methods:

**Hypothesis 2** *A stiffer result will be obtained if a higher volume fraction is used.*

Second, volume fraction and domain dimensions are correlated. Slender structures are obtained when using a very small volume fraction, and simple, uniform structures are achieved if the volume fraction is chosen very high. Thus, there is a range of visually optimal volume fraction for each individual problem definition. For this reason, the result quality of the different algorithms was tested for different volume fractions. The particularly interesting research question was whether the methods fail to give good results for very high and very low volume fractions. A non-directional hypothesis was formulated accordingly:

**Hypothesis 3** *There is no difference in the average solutions quality between the three methods when varying the volume fraction.*

Both Hypotheses 2 and 3 were aimed on the final results of the optimization runs, but not on the efficiency of the algorithms. However, the volume fraction is a constraint parameter that should not change the algorithm formulation explicitly. It was anticipated that there should be no significant difference in the duration of the schemes with a change of the parameter:

**Hypothesis 4** *There is no difference in performance within each scheme when varying the volume fraction.*

All hypotheses were tested with 5%, 10%, 15%, 20%, 25%, and 50% volume fraction.

### 4.1.4 Mesh Density

Mesh dependency as a common problem in TO was reviewed previously (see sections 2.2.5 and 4.1.1). One characteristic of the mesh is its density. Refining the mesh by increasing density is referred to as "h-method" in FEA (see section

2.1). All the benchmark schemes are based on FE calculations. For this reason, it was assumed that more accurate and better TO results can be achieved with the h-method:

**Hypothesis 5** *A better result will be obtained with all scheme if the mesh density is increased.*

The increase in mesh density requires the calculation of matrices with a greater number of elements. Thus, FEA with more elements is usually computationally more expensive. A similar effect was assumed for the three FE-based TO schemes:

**Hypothesis 6** *The computational costs of each scheme will increase if the mesh density is increased.*

The type of this correlation (linear, quadratic, etc.) and a comparison between the different method would be particularly interesting if the hypothesis had been proven. Thus, three regressions were fit to the mesh and performance data of each scheme. Subsequently the calculated regression parameters of the different schemes were compared. The effects of the mesh density were tested by approximately meshing the design domain with 5,000, 10,000, 20,000, 40,000, and 80,000 elements.

### 4.1.5 Element Formulation

The formulation of the elements is another characteristic of a FE mesh. First, increasing the polynomial order of the interpolation function of the elements (p-method) is a way of mesh refinement (see section 2.1). Thus, it was assumed that using linear and quadratic interpolation functions should give different results, equivalently to varying the mesh density (h-method) in the previous chapter. Second, generally more accurate FEM results should be obtained when a higher integration order is used (see section 2.1). Hence, it was anticipated that the use

of full and reduced integration should also give different results. Finally, the number of nodes per element changes with shape of the element. For this reason, it was also important to test if the results change when using triangular instead of quadrilateral elements. Two hypotheses were formulated by combining these three aspects:

**Hypothesis 7** *There is a difference in the solutions quality for each of the methods when varying the element formulation.*

and

**Hypothesis 8** *There is a difference in the performance for each of the methods when varying the element formulation.*

The aim of these hypotheses was to determine if some formulations are advantageous to apply because their calculation is faster and/or if they give better results. This advantage had to be examined for each method individually. The following element types were tested in plane stress conditions: four-node linear quadrilateral with reduced integration (CPS4R), four-node linear quadrilateral with full integration (CPS4), eight-node quadratic quadrilateral with reduced integration (CPS8R), eight-node quadratic quadrilateral with full integration, three-node linear triangle (CPS3), and six-node (modified) quadratic triangle (CPS6M).

### 4.1.6 Mesh Distortion

The degree of mesh distortion can be interpreted as a characteristic of a FE mesh. Highly distorted element shapes lead to poor results [11] and are usually undesired. Figure 12 shows a part of a mesh with extremely distorted elements.

A method for generating and measuring mesh distortion for FE meshes in TO has been developed by Bulman et al. [17]. The same technique was used in

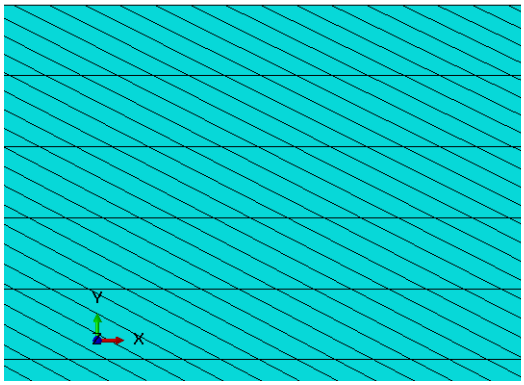


Figure 12: Detail of a Highly Distorted Mesh

this benchmark with different values. Three meshes for each sample case were compared. One of the test cases was the straight mesh, one case was distorted to the left, and one to the right. The distortion was achieved by seeding the top edge of the rectangular domain differently than the other three edges. The top edge of the left-distorted (25%) case had half of the elements evenly spaced within the first 25% of the edge length and the remaining elements evenly distributed over the last 75% of the edge. The right-distorted (75%) case is generated the exact other way around. This method is schematically shown in Figure 13, which compares the three meshes with about 40,000 elements each.

Different degrees of mesh distortion were achieved in this benchmark because the aspect rations of the rectangular design domains differed between the sample cases. Additionally, some sample loading conditions were symmetrical. Thus, it was assumed that these cases show a mirror image of the same effects when distorting the mesh to the left and to the right.

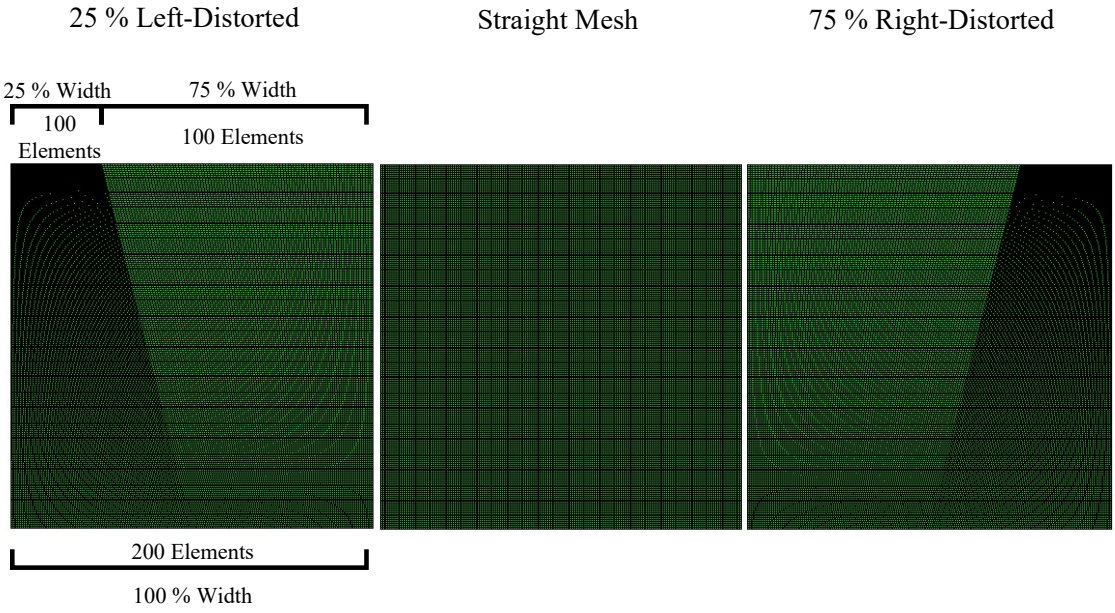


Figure 13: The Left-Distorted, the Right-Distorted and the Straight Mesh

### 4.2 Benchmark Measurements

The previously formulated hypotheses were statistically tested after the benchmark data was collected. The quantitative and qualitative measurements of the benchmark variables, such as solution quality or computational cost, had to be defined in advance. The solution quality, for example, was expressed both quantitatively (strain energy values) and qualitatively (visual assessment). This approach can also be found in other benchmark studies (e.g., in [17]). Additionally, the statistical testing methods had to be reviewed and the associated assumptions had to be checked.

The quantitative measurements are explicit outputs of the TO algorithms. The fundamental assumption of this benchmark was that these computed model results are identical, when the same case is run multiple times. However, the computational time was expected to vary around some constant average value. An example case was run several times to check these assumptions.

The computational costs of a problem were measured as the time that passed from the very start to the very end of the script. This duration is a useful information for any designer that needs to schedule TO analysis. However, it is dependent on the number of iterations, which is not known in advance for the two converging algorithms. Thus, the specific duration per iteration was used in most analyses to measure computational efficiency of the algorithms.

The total strain energy of the system was used as a quantitative measurement of the solution quality. Minimizing this quantity was the objective of all the tested problems. However, all the cases were set up with different dimensions, different volume fractions, and loading condition with arbitrary magnitudes of 1. Thus, the units of the total strain energy were not meaningful and a normalized quantity was used to unify the objective values for different cases. This approach is also used in other benchmark studies (e.g., normalized objective function in [22] or normalized strain energy in [17]).

Normalizing the total strain energy led to the problem of finding a reasonable denominator. Three ideas were tested. First, the total strain energy in each step was divided by the initial strain energy, which represents the strain energy of the uniform partially dense design domain. The advantage was that the results could be interpreted as a percentage of improvement. However, the different schemes penalize intermediate densities differently and, thus, initial strain energies could not be compared between the schemes. Second, the schemes do not penalize fully dense regions. Another idea was to divide each strain energy value by the strain energy of the fully dense design domain. This approach, however, did not unify the different cases. The best idea would be to divide the values by a known analytically solution. However, this solution was not known for all the cases and, thus, was

approximated by the lowest final strain energy value for each case. This approach showed the most unified results. The normalized strain energy is has no unit.

A visual assessment of the results was used as the qualitative measurement in the benchmark. The assessment was based on a symmetrical 5-point interval scale, where 1 was the lowest rating and 5 was the best rating. The rating combined two quality measures, the qualitative similarity to the exact analytical solution and the benefit for a designer using this tool. That is the question, "Can an engineer easily use the result without the need for major structural changes?" The five ratings had the following literal descriptions:

1. not close to the expected shape at all, pieces floating
2. some intermediate densities, can be made reasonable by adding some density in some regions (changing the threshold, see sections 2.6 & 3.2)
3. reasonable, physically functioning light weight structure, no symmetry
4. good results with minor flaws (e.g., small non-symmetries), some changes need to be made for perfect shape
5. perfect shape

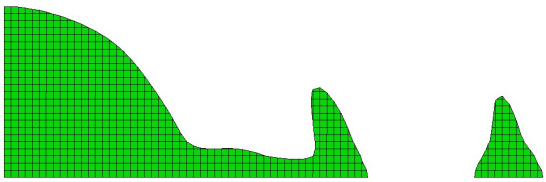

Figure 14: A Half Michell Arch Result with a Rating Score of 1

Figures 14 to 18 show five different solutions, one for each of the ratings. Visual assessments like this are generally rather subjective and potentially biased. For this reason, all test cases were assessed by the author and two independent mechanical

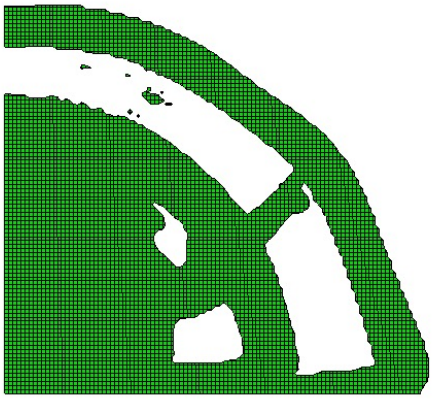

Figure 15: A Half Michell Arch Result with a Rating Score of 2

engineering students with a broad background in structural design and FEA. The arithmetic mean of the three ratings was then used as the qualitative solution quality measurement. The specific problem definitions and the TO method were not clearly disclosed to the other two participants to avoid a bias that promoted a particular scheme. The assessment varied between the three persons. However, only few test cases (3 out of 306 with a difference of 4 and 13 with a difference of 3) showed significant deviation in the rating scores. Most of these deviations could be attributed to unexpected and complex designs which differ from the analytical solution. The individual weighting of "similarity to the exact solution" and "benefit for the designer" caused the deviating ratings.

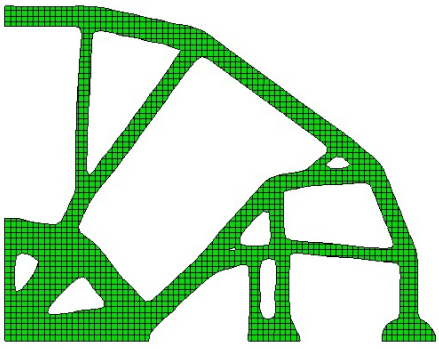

Figure 16: A Half Michell Arch Result with a Rating Score of 3

Using visual assessment has been criticized in the TO literature. Rozvany [10], for example, rated the visual comparison of TO results and the exact optimal truss topology as "very subjective". Bulman et al. [17] were not satisfied by sole visual similarity and demanded quantitative measurements in algorithm testing from an optimization perspective. However, PMR is a heuristic scheme intended as a tool for a designer and, thus, this benchmark was not exclusively focused on the optimization perspective but rather on the practical applicability.

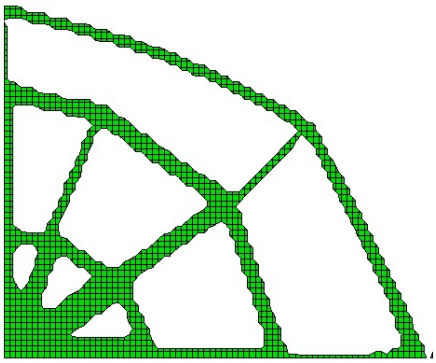

Figure 17: A Half Michell Arch Result with a Rating Score of 4

The previously formulated hypotheses were checked using three statistical tests: the linear least-squares regression, the one-way analysis of variances (ANOVA), and Tukey's honest significance test. The theoretical foundations of these tests can be found in text books such as [51] or [52]. Using these tests requires a certain quality of the data and the fulfillment of specific assumptions for each test. Validity and reliability were previously discussed in section 4.1.1 and as the fundamental assumption of the quantitative measurements. The objectivity of the data was reviewed along the visual assessment. One problem of this research was the size of the sampling. Some tests require a minimum number elements in a sample. This minimum size was roughly estimated to about 30 samples per group for this study, which was not met in all the tests. Additionally, the ANOVA and Tukey's range test require independence, normality, and homogeneity of variances

(homoscedasticity). The independence of the data was a reasonable assumption because the different test cases do not affect each other. All other premises were also assumed to be met. However, random Kolmogorov-Smirnov and Levenne tests showed that the assumptions of normally distributed data and homoscedasticity did not always hold in this benchmark. The small group size and the lack of compliance with the statistical assumptions are part of this research's limitations (see section 6.2).

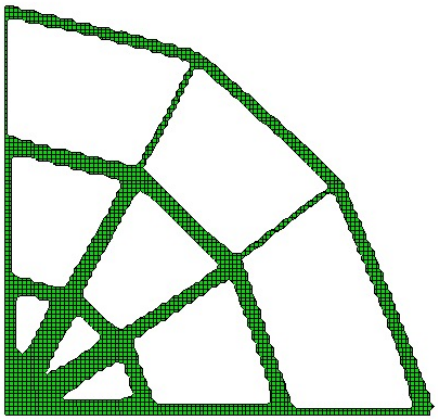

Figure 18: A Half Michell Arch Result with a Rating Score of 5

### 4.3 Benchmark Sample Cases

Six sample cases were chosen for this benchmark: the "Half Michell Arch", Cases 1 to 4, and the "MBB Case". These samples include some versions of the standard benchmark cases introduced in section 2.4. Cases 1, 2, 3, and 4 are identical to those used in the PMR development by Taggart et al. [3]. Most of the sample cases go back to the work of Michell [18]. The loading and boundary conditions and some analytically exact solutions or good TO solutions of the sample cases are illustrated in Figures 19 to 25.

The first case, which in this benchmark is called the Half Michell Arch (see Figure 19), is a version of the symmetric Michell Half Wheel problem illustrated

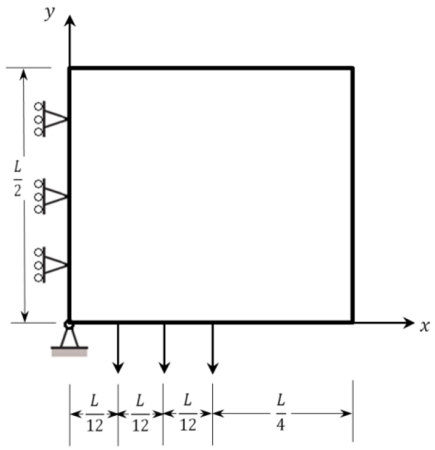


Figure 19: The Half Michell Arch Loading and Boundary Conditions

in Figure 20 (see [2]). The Half Michell Arch sample makes use of the symmetry by imposing a boundary condition on the symmetry axis and replaces the line load with three point loads. The exact analytical solution would have a fan of trusses and three arches connecting the point loads and the left face of the design domain.

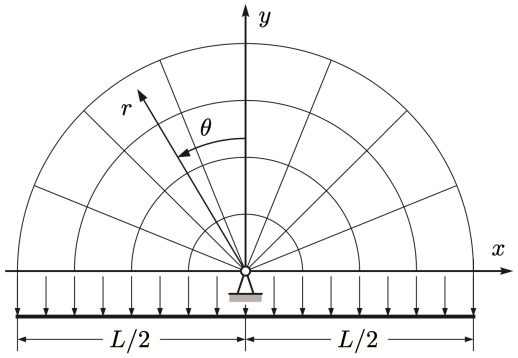


Figure 20: The Michell Half Wheel [2]

Case 1 is a center fan topology with a single central load. Case 2 and 3 are variations of that case (see Figure 21). Case 2 deviates to Case 1 because of a pin support instead of a roller on the right side. Case 3 uses the full design domain above and below the supports. Case 4 is a cantilever problem. Some cantilever problems clamp a complete face of the design domain (e.g., [10], [17]). The used sample only supports two points.

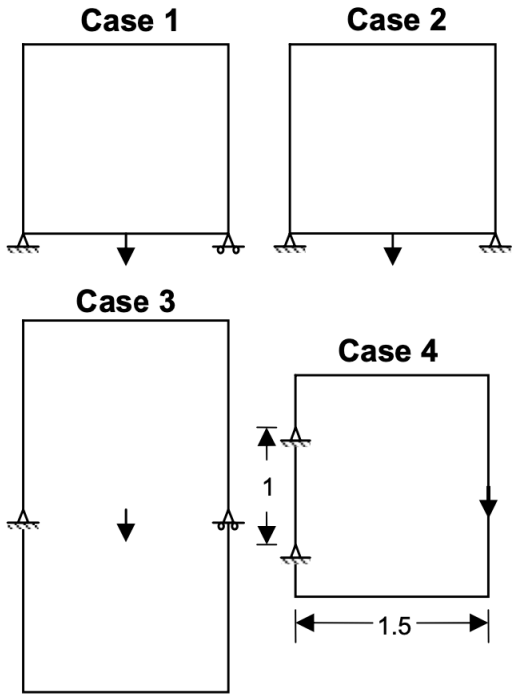


Figure 21: Loading and Boundary Conditions of Cases 1, 2, 3, and 4 [3]

Figure 22 shows visually good solutions obtained with the PMR scheme by Taggart et al. [3] and Figure 23 additionally shows the analytical solution for a cantilever problem.

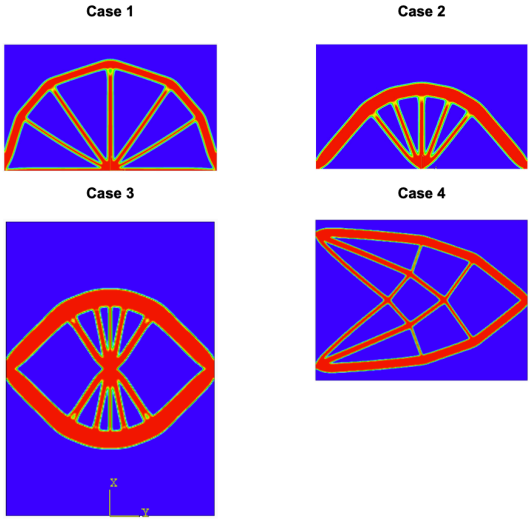


Figure 22: Good Results for the Cases 1, 2, 3, and 4 [3]

The Messerschmidt-Bülkow-Blohm (MBB) Beam (see Figure 24) is one of the standard sample cases that are widely used in TO literature. The beam is describing the loading conditions of the floor in the fuselage of an aircraft [17]. Figure 25 shows half of the analytical solution.

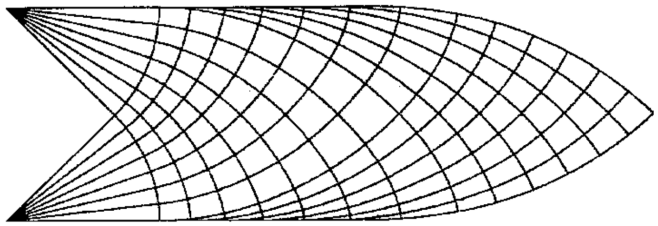

Figure 23: Exact Analytical Solution for a Cantilever Problem [4]

Figures of the loading and boundary conditions within the Abaqus model can be found in Appendix A.

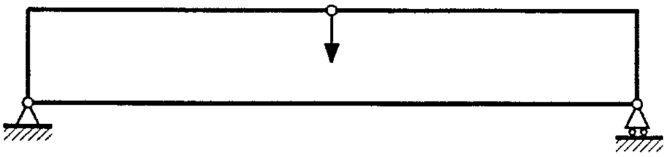

Figure 24: Loading and Boundary Conditions of the MBB Beam [4]

All these cases are two-dimensional. The benchmark was exclusively focused on two-dimensional problems because of the computational time associated with three-dimensional cases that have a high resolution. For example, a square-shaped, two-dimensional design domain with 200 elements along each edge has a total of 40,000 elements. This resolution gave good results, while only about 30 min were required for the calculations. A three-dimensional mesh with the same resolution would have a total of $200^3 = 8,000,000$ elements, which would exceed the computational capacity of the computer used. Only one three-dimensional case was run for validation purposes (see section 3.2).

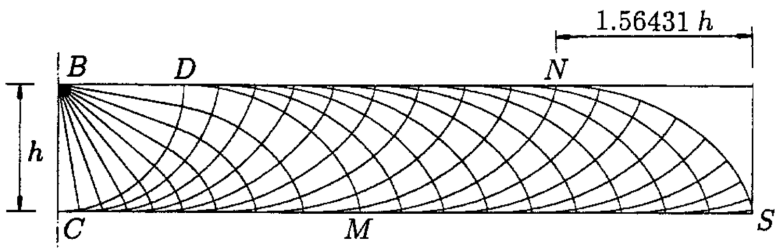


Figure 25: Half of the Analytical MBB Beam Solution [4]

### 4.4 Benchmark Settings

The same computer has been used for computing all test cases in this benchmark. The machine has the following specifications: eight core Intel i7-9700 CPU operating at 3 GHz, 16 GB of RAM, one SSD with 256 GB memory, Intel UHD Graphics 630 and AMD Radeon Pro WX 3100. The operating system was Microsoft Windows 10 Pro (64 bit).

Identical settings were applied for all test cases. The mechanical properties of steel were used (Young's Modulus: $210 * 10^9 \frac{N}{m^2}$ and Poisson's Ratio: 0.3). The default settings for the Tosca optimizations were accepted. That is, among others, the penalization factor $p = 3$, minimum density $\rho_{min} = 0.001$, the two convergence criteria "objective function delta" = 0.001 and "element density delta" = 0.005, and the setting "delete soft elements" disabled. The maximum number of iterations was set to 75 for all cases. All PMR results were run with time smoothing and symmetry check enabled.

Table 2: Default Volume Fraction and Dimensions of the Six Cases

| Case \Property | Default Volume Fraction [%] | Height [m] | Width [m] |
|---|---|---|---|
| Half Michell Arch | 5 | $150 * 10^{-3}$ | $150 * 10^{-3}$ |
| Case 1 | 10 | 100 | 100 |
| Case 2 | 15 | 100 | 100 |
| Case 3 | 10 | 200 | 100 |
| Case 4 | 20 | 150 | 150 |
| Case MBB | 50 | 10 | 100 |

Table 2 includes the default volume fractions and the dimensions of the sample cases. Each case's straight standard mesh had 40,000 four-node linear quadrilateral elements with reduced integration in plane stress conditions (CPS4R). The PMR problems were solved with 75 iterations by default. These standard settings were defined for each case in advance based on pre-benchmark tests. Some of these parameters were varied in the different benchmark runs. It needs to be mentioned that there was no intended physical meaning to the units and that load magnitudes of 1 were used. The meshing of the domains was automated in the benchmark by using global edge seeds. The seeds that were used to generate specific numbers of elements for each case can be found in Table C.4 in the appendix.

# CHAPTER 5

# Findings

Collecting and evaluating the benchmark data was the fourth methodological step of this research. One the one hand, this chapter reviews the findings on the newly developed PMR algorithm and evaluates the associated enhancement of the method. On the other hand, the comparative benchmark is discussed and the formulated hypotheses are tested. Some results fall, however, under both of these categories.

## 5.1 Findings about the Enhanced PMR Version

The first half of this study was focused on the potential increase in performance and ease of use, when the Python-Abaqus interface was used instead of the existing standalone MATLAB code. The performance of specific parts of the algorithm was already discussed in section 3.2.

### 5.1.1 Rerunning the Same Case

The reliability of the quantitative measurement of the PMR method was checked by rerunning the same case multiple times. Each run generated the exact same density distribution for each iteration step, as it was expected beforehand. Hence, the structure and the quantitative measures, such as the total strain energy of the system, were identical.

The algorithm's duration varied with each run. The values scattered around an average if each run was manually started. This variation was also expected beforehand. However, if the runs were automatically started in a loop of a Python script, a linear trend was observed. The experiment was conducted multiple times to rule out a coincidental trend in the data potentially caused by random effects.

Figure 26 shows the specific duration data of 90 PMR runs of the Half Michell Arch Case (5% volume fraction, 40,000 CPS4R elements, 10 iterations). The first 30 runs were executed in one loop, the next 30 in another loop, and each of the last 30 runs were started manually.

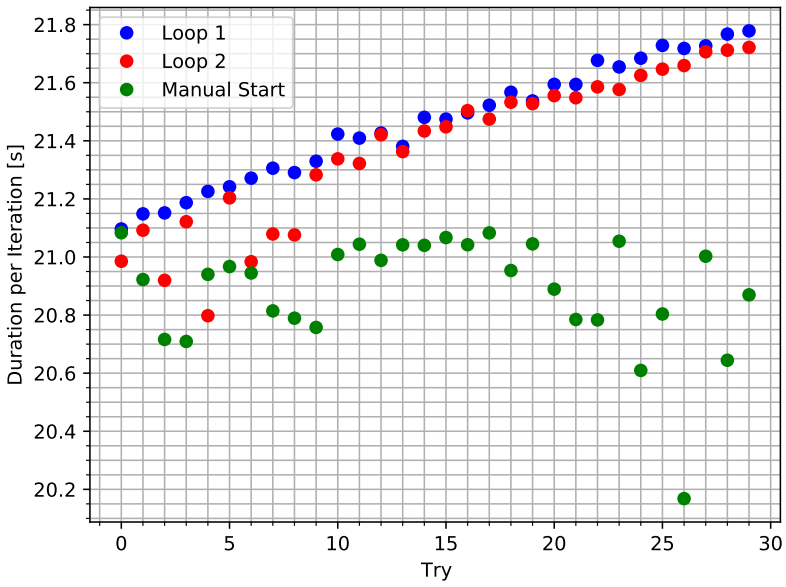


Figure 26: Specific Duration of the Same PMR Problem Run 90 Times

Linear regressions were fitted to the loop-executed data points. The linear functions are good approximations of the trend in the data (loop 1: $R^2 = 0.9868, p = 0.0000$ and loop 2: $R^2 = 0.8911, p = 0.0000$). Both cases had similar interception of about 21 seconds and similar slopes of 0.0231 seconds per try and 0.0282 seconds per try, respectively. The arithmetic mean of the manually started cases was also about 21 seconds.

Loops were also used to conduct the parametric study of the main benchmark, e.g., to loop over different mesh densities. The linear trend caused by the loop could distort a trend originally related to the change in parameter. However, it was assumed that this effect is negligible for two reasons. First, the slope measured

for this sample case was three magnitudes smaller than the average value. Hence, looping over five to seven different values for a parameter would not change the specific duration significantly. Second, the scattering of the specific duration when manually started was of the same magnitude as the linear trend. This linear trend related to looping over the execution of the PMR algorithm was not found for looping over the execution of the Tosca SIMP and RAMP problems.

### 5.1.2 Results of the Integration Point-based Approach

An alternative approach to the PMR method using densities at integration points instead of nodal densities as design variables was introduced in section 3.4. This new scheme was tested on quadrilateral elements with linear and quadratic interpolation functions as well as with full and reduced integration. Figures 27 and 28 display the resulting structures for the linear reduced integration element (CPS4R) and the quadratic full integration element (CPS8). The test problem was the default Half Michell Arch Case with 5% volume fraction, 40,000 elements, and 75 iterations. Results using CPS4 and CSP8R elements can be found in Appendix B.1.

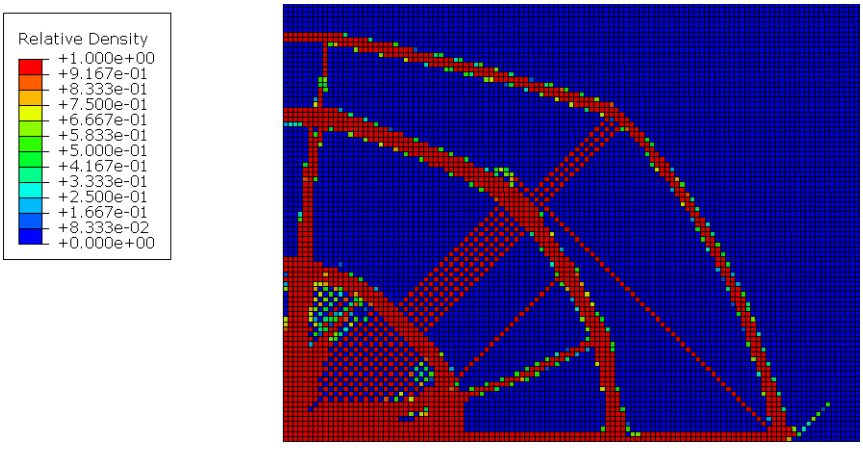


Figure 27: Integration Point-based PMR Half Michell Arch Result (CPS4R)

The CPS4R solution was the only result to show checkerboarding patterns. These elements have a single integration point in the element center. Hence, this case could be seen as equivalent to having an elemental design variables. The checkerboarding was assumed to be caused by this single value elemental allocation of density that does not require continuity between the elements. SIMP is also a element-based scheme and checkerboarding is one of SIMP's common issues (see section 2.2.5).

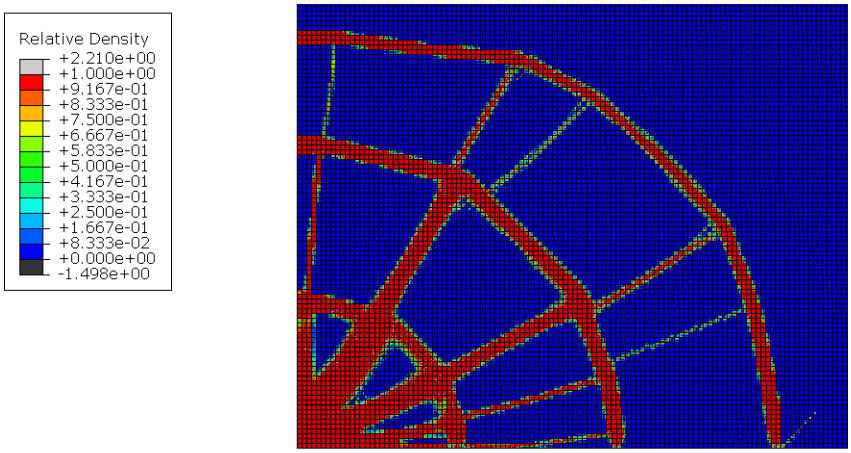


Figure 28: Integration Point-based PMR Half Michell Arch Result (CPS8)

The other three element types did not show checkerboarding. Checkerboarding is commonly prevented by using higher order elements (see section 2.2.5 and [33]). A similar effect might be seen here for the cases that have more than one integration point per element and partly higher interpolation function order. However, the solutions of these element types have a different disadvantage. It was discovered that relative density values $\rho(\mathbf{x}) > 1$ and $\rho(\mathbf{x}) < 0$ are artificially generated by the density interpolation within the element. These values, particularly negative relative densities, have no meaningful physical interpretation. The legend in Figure 28 includes the minimum (-1.498) and maximum densities (2.210). The color map of the contour plot, however, is limited to values between zero and one, and

nodal output averaging is disabled. The interpolation point-based PMR approach was not pursued any further due to the two discussed disadvantages.

### 5.1.3 Comparison of Different Versions

There are multiple SIMP and level set TO codes available online for academic purposes (see section 2.5). These algorithms were built in the similar way as the standalone PMR codes. Comparing such codes' performance and ease of use with a commercial software is not a meaningful exercise. Some of the codes are purposely not optimized for performance, but for readability [32], [44]. Hence, the goal of this evaluation was to compare these standalone schemes with each other and subsequently compare the results of this analysis to the results of the main benchmark.

Two SIMP implementations in MATLAB, the 99- and the 88-line code, and the Python version of the 88-line code were used in the quantitative study on performance. The performance was measured as the specific duration of each iteration for the same mesh densities that were tested in the mesh refinement study of the main benchmark (5,000, 10,000, 20,000, 40,000, and 80,000 four-node elements). The chosen case was the default Half Michell Arch (5% volume fraction and 75 PMR iterations). The SIMP codes were run using a penalization factor $p = 3$, a minimum radius $r_{min} = 1.2$ for the mesh-independency filter, and a density delta convergence criteria of 1%.

Figure 29 shows the results of the performance benchmark. The dashed lines are the PMR codes and the solid lines the SIMP codes, square markers are used for MATLAB codes and round markers for Python implementation. Both Python scripts had to be paused for 0.01 seconds each iteration for smooth visual feedback. It was determined that the Python versions of the same algorithm were always slower than the MATLAB versions. Additionally, the claimed performance boost

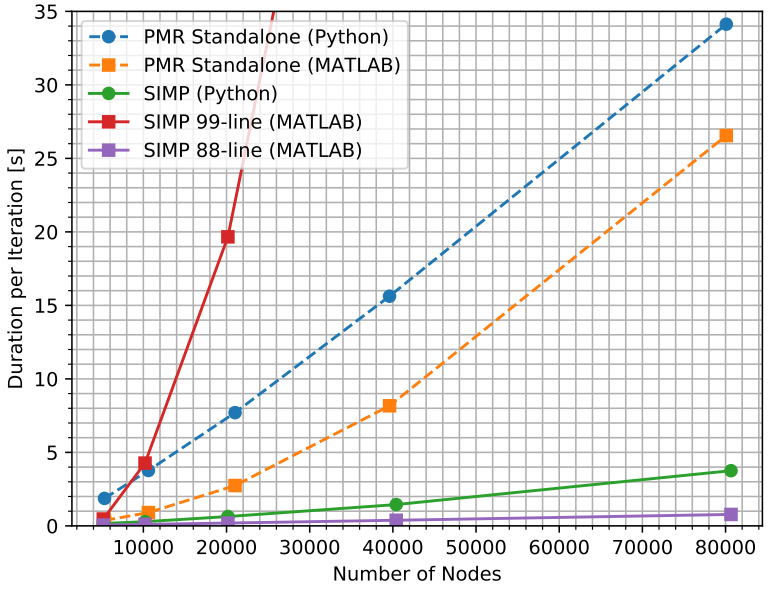


Figure 29: Efficiency of the Academic Codes for Different Fine Meshes

of two magnitudes between the 99- and the 88-line version was validated in this test (see section 2.5 and [35]). The improved codes generally appeared to lose less efficiency with a higher number of elements than the original codes. The original PMR version was faster for bigger problems than the original 99-line SIMP version. For this reason, it could be assumed that a properly improved PMR MATLAB code could achieve performance similar to the 88-line code.

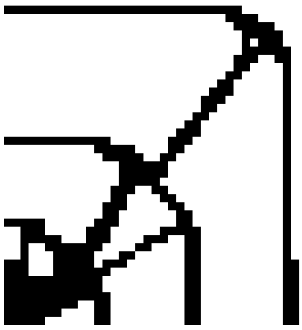

Figure 30: Half Michell Arch Problem Solved with 129-line Level Set Code

The level set methods were exclusively tested qualitatively. Figures 30 and 31 show structures generated with the level set algorithm by Challis [32] and by the TopOpt group [46], respectively. Both algorithms found the essential structures for the Half Michell Arch Cases. However, finding a visually satisfactory structure was highly dependent on the right choice of numerous parameters. The input parameter for the solutions generated with level set schemes can be found in Table C.5 in the appendix.

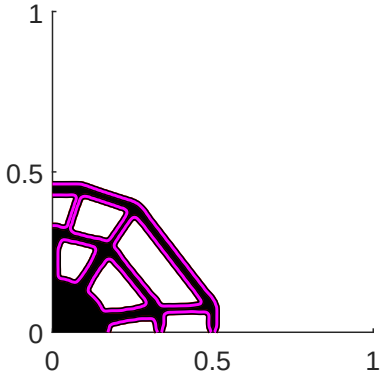


Figure 31: Half Michell Arch Problem Solved with TopOpt Level Set Code

The problem definition in the SIMP and in the PMR scheme are similar. It was generally easier to define the Half Michell Arch problem and to apply the TO method in the SIMP codes than in the level set schemes. However, the SIMP results showed checkerboarding if a minimum radius $r_{min} \leq 1$ was used (see Figure 5). The TopOpt level set scheme uses the MATLAB-specific struct object. Using this data structure type similarly in the other MATLAB scripts would improve their ease of use.

## 5.2 Findings of the Benchmark

This section reviews the findings of the main benchmark. The sampling of this benchmark comprised 306 individually assessed test cases, which did not include

the mesh distortion cases, the previously discussed academic test cases, and other auxiliary test cases. Only considering the assessed test cases, 126 used the PMR scheme, 90 the SIMP, and 90 the RAMP formulation in SIMULIA Tosca Structure. A significance level of 5% was used. The values of the statistical test results can be found in detail in Appendix C.

### 5.2.1 Number of Iterations

The number of iterations of the PMR scheme was varied to analyze its effects on the solution, both quantitatively and qualitatively. The goal was to find a reasonable iteration number balancing duration and result quality. Hence, Hypothesis 1 was formulated accordingly. The quantitative analysis of the normalized strain energy of the final structure values can be found in Figure 32. There was no clear trend found in the data. That is, more iterations, and thus a smoother transition, did not necessary generate a stiffer result. However, solutions with only 10 iterations were either significantly stiffer or more compliant than the other solutions, which were similarly stiff. The 10 iteration results also did not generate visually good structures. Hence, a minimum number of iterations greater than 10 should be chosen.

The qualitative analysis tested the resemblance of the visual assessment distributions of the seven groups with 10, 20, 50, 75, 10, 150, and 250 iterations (see Appendix B.2). It was discovered that the distributions did not have the same population mean ($F = 15.44, p = 0.0000$). The mean visual assessment was higher for cases with more iterations. However, the pairwise distribution comparison could only significantly prove that solutions with 50 or more iterations were better than solutions with 10 iterations and results with 75 or more iterations were better than those with 20 iterations. It should be mentioned again that each group only consisted of 6 sample cases. The appropriate group size for an ANOVA and the

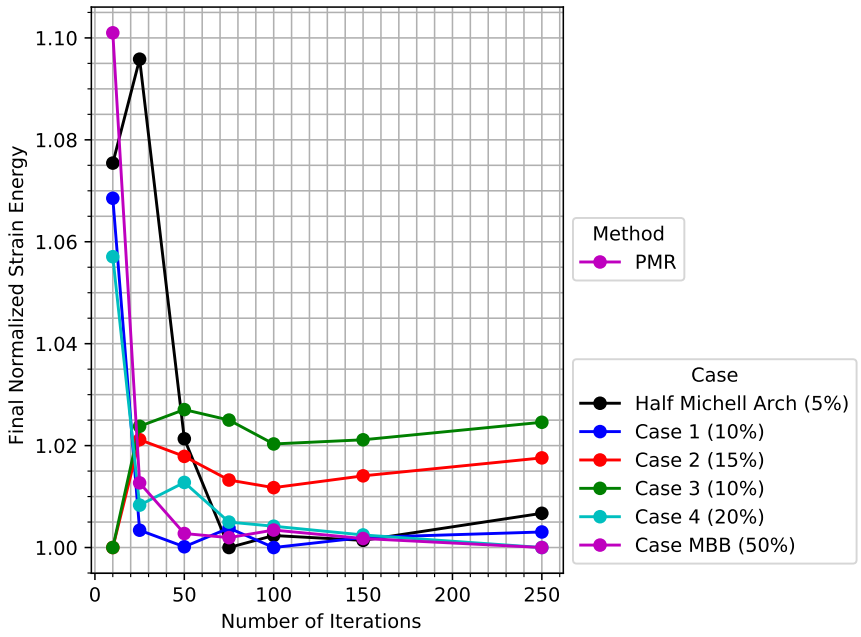


Figure 32: Final Normalized Strain Energy for Different Iteration Numbers

Tukey test was, however, estimated to about 30 or more elements (see section 4.2). The results led to the conclusion that a first good result could be achieved with 50 to 75 iterations (assessment means: 3.17 and 3.56).

The duration of the variable initialization at the beginning of the script was assumed to be independent of the total number of iterations and, thus, the specific duration should decrease with more iterations in total. It was discovered that the duration of an iteration in 4 of 6 sample cases was considerably higher if the scheme used 10 or 20 iterations in total. These cases had similar specific durations for 50 and more total iterations (see Appendix B.2).

Based on the three reviewed aspects, it was concluded that PMR problems with 75 iterations give good results and have a reasonable duration. Hence, all other PMR benchmark cases were conducted with 75 iterations by default. Calculating more iterations will take longer, but better results might also be generated. Additionally, 75 iterations were set as the iteration maximum for the RAMP and

SIMP formulation. The number of iterations these schemes needed to converge was analyzed separately. These iteration numbers did not noticeably relate to a change in volume fraction, element formulation, or mesh density in the benchmark tests. Thus, it could not be stated if and how a change of these parameters would affect the iterations needed to converge. These iteration numbers were also analyzed from an overall perspective (see section 5.2.4).

### 5.2.2 Volume Fraction, Mesh Density, and Element Formulation

This section discusses the effects of varying the volume fraction, the mesh density, and the element formulation. The duration (efficiency), the final normalized strain energies (quantitative solution quality), and the assessment (qualitative solution quality) are reviewed.

The total duration of all methods is dependent on the number of iterations. It was previously discussed that the number of iterations needed in the converging schemes (SIMP and RAMP) could not be explained by effects of varying volume fraction, mesh density, and element formulation. For this reason, only the specific duration per iteration was used to compare the methods in this section.

Changing the volume fraction was assumed to have no effect on the efficiency of the schemes (Hypothesis 4). The benchmark results verified this assumption qualitatively. Figure 33 plots the duration per iteration over the volume fraction. It was discovered that the duration per iteration was varying around a constant value as expected. SIMP and RAMP formulated problems were faster than PMR cases. Some samples ran faster with the RAMP scheme and some with SIMP. The efficiency of SIMP and RAMP was very similar and the different sample cases also had similar computational costs.

A finer mesh was expected to increase the computational costs of all the schemes (Hypothesis 6). This effect was indeed found in the benchmark data.

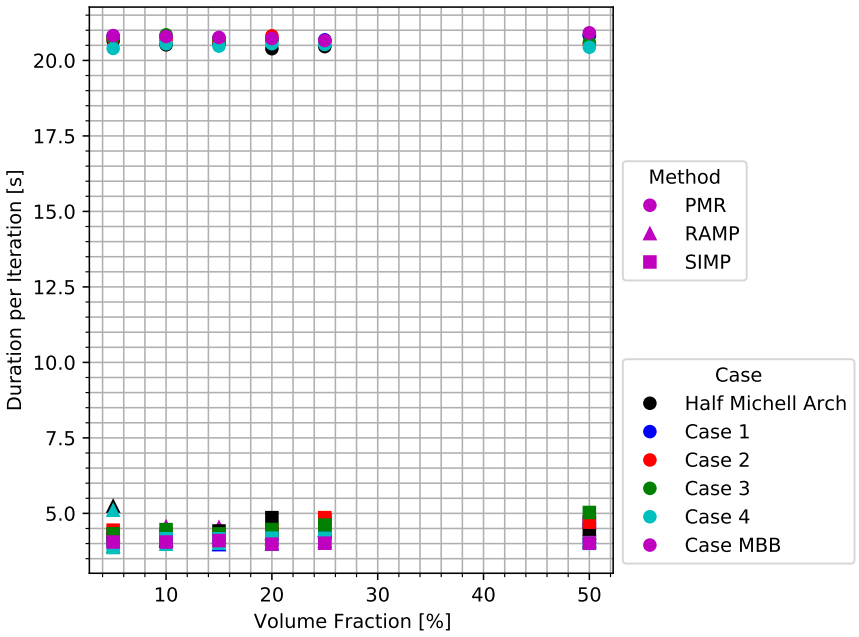


Figure 33: Duration per Iteration for Different Volume Fractions

A linear trend was determined between the duration per iteration and the total number of nodes in the mesh for each of the schemes. Figure 34 shows the collected duration data and three linear regressions (PMR: orange, RAMP: grey, SIMP: lime). The linear functions approximated the data points well (PMR: $R^2 = 0.99, p = 0.0000$, RAMP: $R^2 = 0.96, p = 0.0000$, SIMP: $R^2 = 0.99, p = 0.0000$). SIMP and RAMP had similar intercepts (RAMP: 2.01 seconds, SIMP: 1.91 seconds) and slopes (RAMP: 0.000053 seconds per node, SIMP: 0.000057 seconds per node). That is, their computational efficiency was almost identical.

The PMR code was generally slower (intercept: 13.27 seconds) and about three times more sensitive to a change in mesh density (slope: 0.00017 seconds per node). However, the duration of an iteration in the PMR scheme was limited by the time of the job execution. That is, the specific duration could not be shorter than Abaqus needed to run the job, including the license check ($\approx$ 5.5 seconds) and pre-processing ($\approx$ 6 seconds). These two processes took a significant amount of the

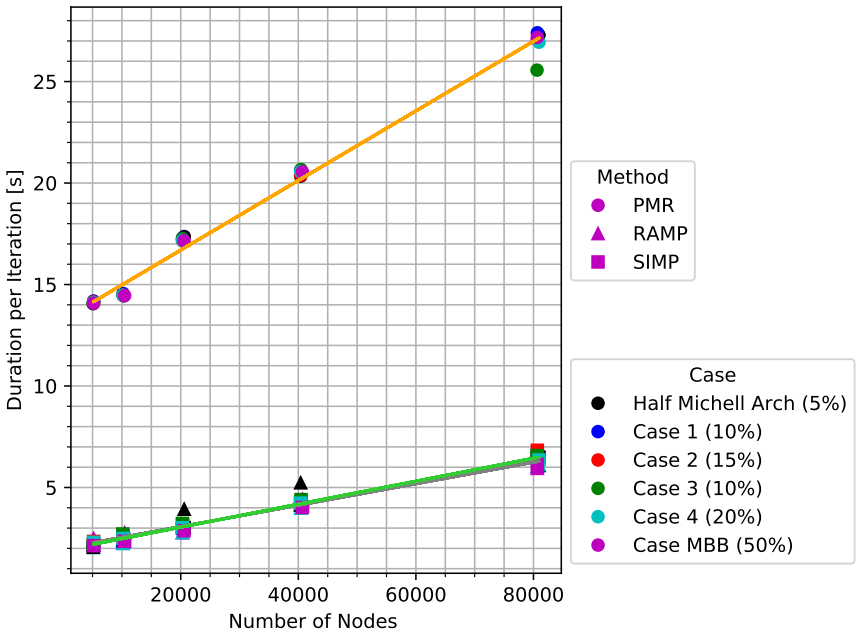


Figure 34: Duration per Iteration for Different Mesh Densities

total iteration duration in the tested cases (about 11.5 of 17.5 seconds, see section 3.2). If the PMR code could be implemented into Abaqus like the Tosca SIMP and RAMP algorithms, it could be assumed that these processing times would be eliminated. Hence, these times could be subtracted from the intercept, which than would be very similar to the intercept of the RAMP and SIMP regressions. Moreover, the higher slope was most likely caused by not fully optimized code and the use of time intensive loops, e.g., for reading the strain energy density values. Considering these two effects, it could be assumed that a commercial PMR code would be competitive. The results of this analysis were very similar to the benchmark of the academic software. The academic codes were faster than the Abaqus schemes for the tested meshes. The only exception is the PMR Abaqus code, which outperformed the standalone algorithm when using a 80,000 element mesh and presumably when using even finer meshes.

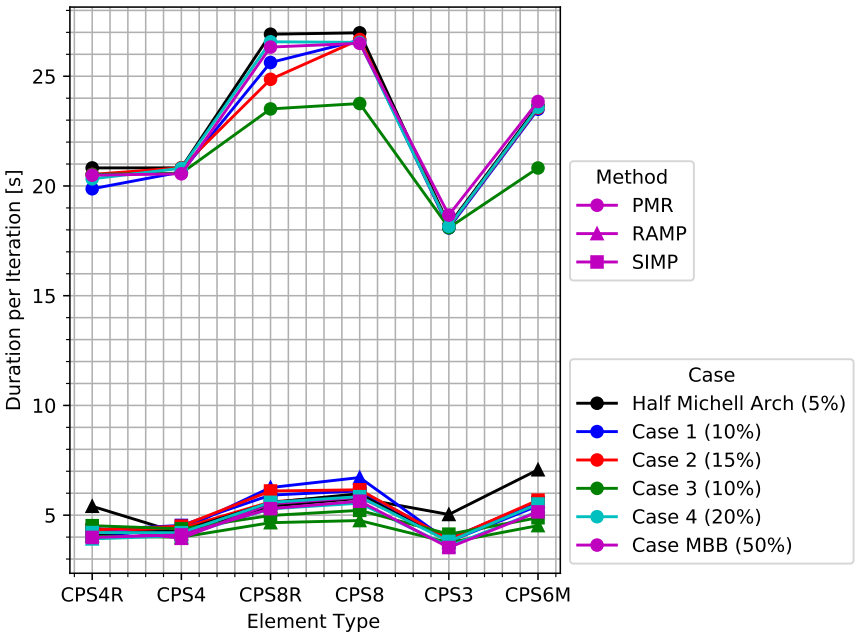


Figure 35: Duration per Iteration for Different Element Types

The element formulation was also expected to have an effect on the performance of the schemes (Hypothesis 8). In fact, rather the sum of two effects than one single effect was discovered, the integration order effect and the interpolation order effect. The latter was more important and is similar to the effect discussed in the previous paragraphs. All test cases in this analysis were meshed with 40,000 elements. That is, four-node elements generated about 40,000 total nodes, eight-node elements 120,000, the three-node triangle 20,000, and the six-node triangle 80,000. A linear trend similar to the previous ones was noticed in the duration per iteration and number of nodes data. The far smaller effect was caused by the change of the integration order. Full integration was slightly slower than reduced integration in most of the test cases. That is, when both effects were combined, the linear triangle meshes were the fastest and the eight-node quadrilaterals were the slowest. Figure 35 shows the cumulative effect of the element formulation by plotting the specific duration for the different tested element types. The continu-

ous line connecting the data points in the discrete groups is only drawn to visually assist determining which values are smaller or bigger, but has no physical meaning.

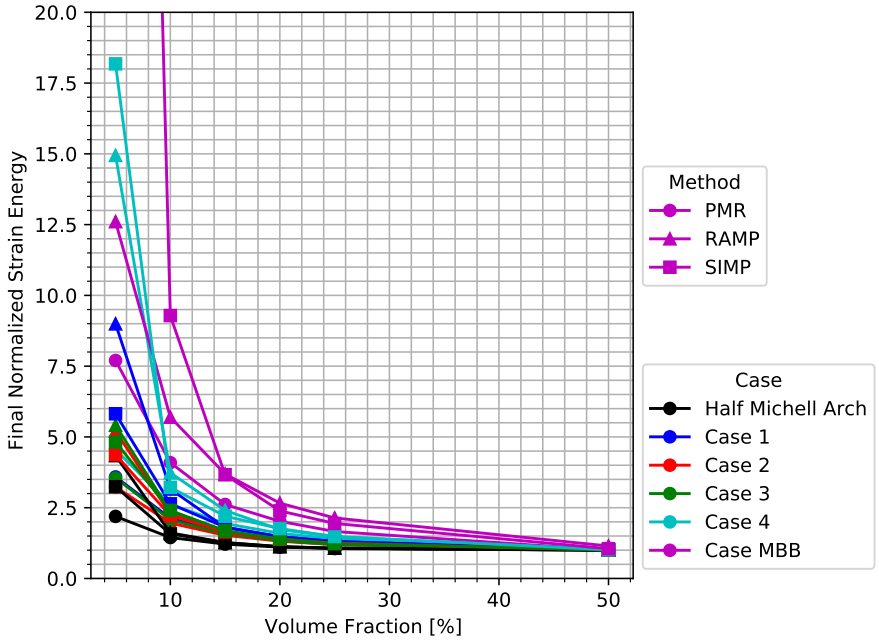


Figure 36: Final Normalized Strain Energy for Different Volume Fractions

The normalized total strain energy of the final structure was used as a stiffness measurement and, thus, as a quantitative solution quality measurement. It was assumed that the solution of an identical problem should generate a stiffer structure with a higher volume fraction (Hypothesis 2). Hence, the final normalized strain energy value was expected to be lower for higher volume fractions. Figure 36 plots the final normalized strain energy values over the tested volume fractions for each sample case and method. The stiffest structures were obtained, when the highest volume fraction was defined, as expected. The decreasing trend in the normalized strain energy data was over-proportional, presumably exponentially. Furthermore, it was found that the scattering between the different sample cases and methods decreased with the volume fraction. The reason for the resemblance of high volume fraction solutions could be that these structures were simple, undefined, and

looked similar for different sample cases and methods. The low volume fraction structures, however, had more slender trusses, more complex designs, and, thus, the transmission of loads was very different for each structure. Hence, the stiffness of these structures scattered more. Moreover, it was found that for each sample case and volume fraction the PMR scheme generated the stiffest structures.

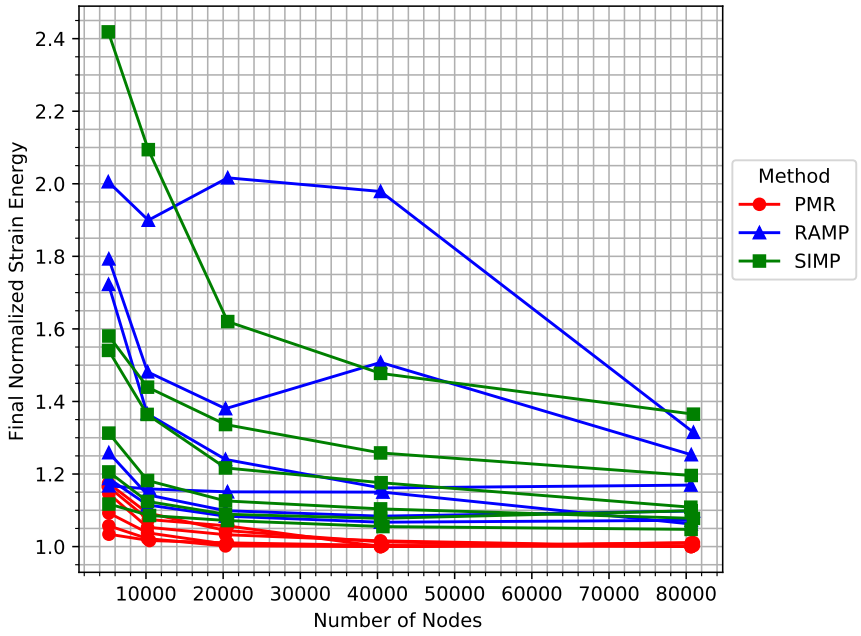


Figure 37: Final Normalized Strain Energy for Different Mesh Densities

The objective of the minimum compliance problem is to generate structures that are as stiff as possible. Thus, the stiffest solution is the quantitatively best solution. Better results were expected when the mesh density was increased (Hypothesis 5). This effect was found in the benchmark data (see Figure 37). The normalized strain energy of the final structures was over-proportionally decreasing with an increase in mesh density in most cases. However, this effect was stronger and clearer for rather coarse meshes, the finer meshes (20,000, 40,000, and 80,000) had similar strain energy values. One possible reason for the decreasing final strain energies could be that point loads generate stress concentrations,

which finer meshes can handle better. However, a pressure load PMR case showed similar trends for the final strain energy. The finer mesh described the problem in every iteration better. The difference between fine and coarse mesh results, however, grew with the iterations. That effect caused the normalized strain energy curves to split over the iteration history (see Figure 38). The boundary and loading conditions for this case can be found in the appendix A.2. Moreover, another effect, which was already found in the previous analysis, was also discovered with varying mesh density. The PMR code generated stiffer final results than the SIMP and RAMP equivalents. This observation is illustrated by the color code in Figure 37. The PMR solutions are red, RAMP results blue, and SIMP results green.

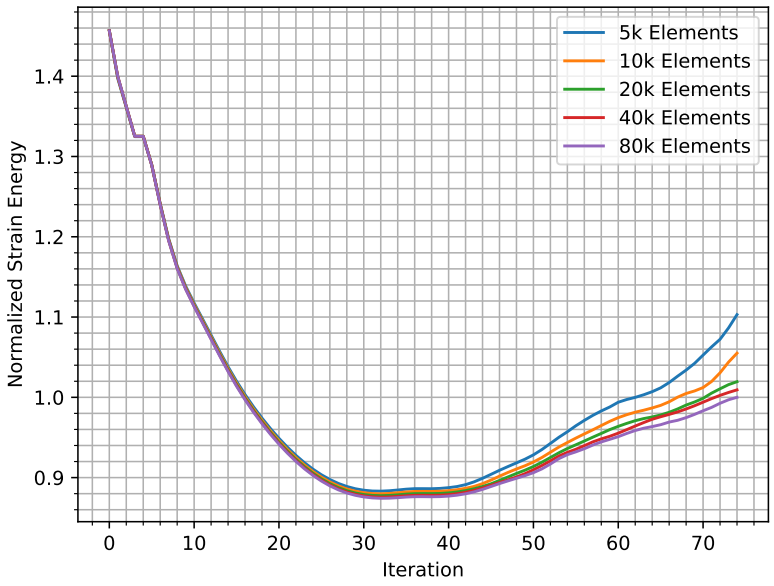


Figure 38: Normalized Strain Energy History of a Pressure Load Problem

Hypothesis 7 assumed a difference in solution quality for different element types. Figure 39 shows the final normalized strain energies resulting from the sample cases, when different elements types were tested. Multiple patterns were noticed. The four-node quadrilateral with reduced integration, which has only a

single integration point, gave the most compliant final structure in most of the cases. The stiffest structure was generated by PMR scheme with three-node triangles in 5 of 6 sample cases. Full integration results were mostly stiffer than reduced integration solutions. This effect was more distinct for the four-node than for the eight-node quadrilateral elements. The eight-node problems gave in many cases more compliant solutions than the fully integrated four-node element. The six-node triangle mesh was almost always more compliant than the three-node triangle mesh. A more compliant or a stiffer final result does not necessarily mean that the structure is physically very different, but that the strain energy quantities calculated by FEA are higher or lower. To give this strain energy patterns a meaningful interpretation, the next step was to analyze the benchmark cases qualitatively.

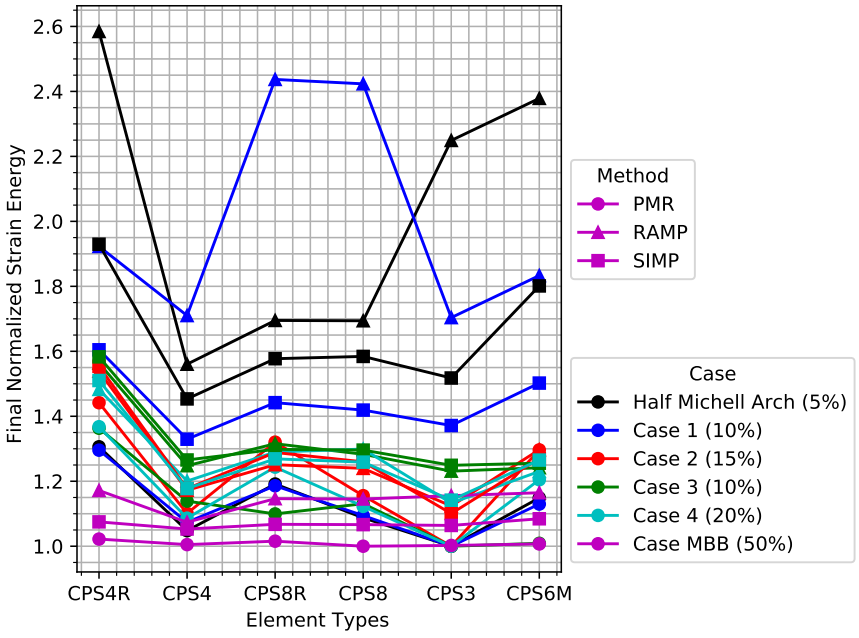


Figure 39: Final Normalized Strain Energy for Different Element Types

The visual assessment rating was used as a qualitative solution measurement. The assessment of the test cases was assumed to be similar for all three methods

if different volume fractions were used (Hypothesis 3). That is, it was expected all schemes generate the same solution quality on average. The average values could indicate if good solutions are found by all schemes even if very high or very low volume fractions were imposed. However, the PMR, RAMP, and SIMP scheme did not achieve the same average quality in the volume fraction benchmark ($F = 8.55, p = 0.0004$). The pairwise comparison showed that the PMR method on average generated better results than RAMP ($p = 0.0027$) and SIMP generated better results than RAMP ($p = 0.0010$). The RAMP method had the most cases in the worst rating category and no cases in the best category as illustrated in Figure 40. The histogram and this analysis contained 108 cases in total (6 sample case * 6 different volume fractions * 3 methods).

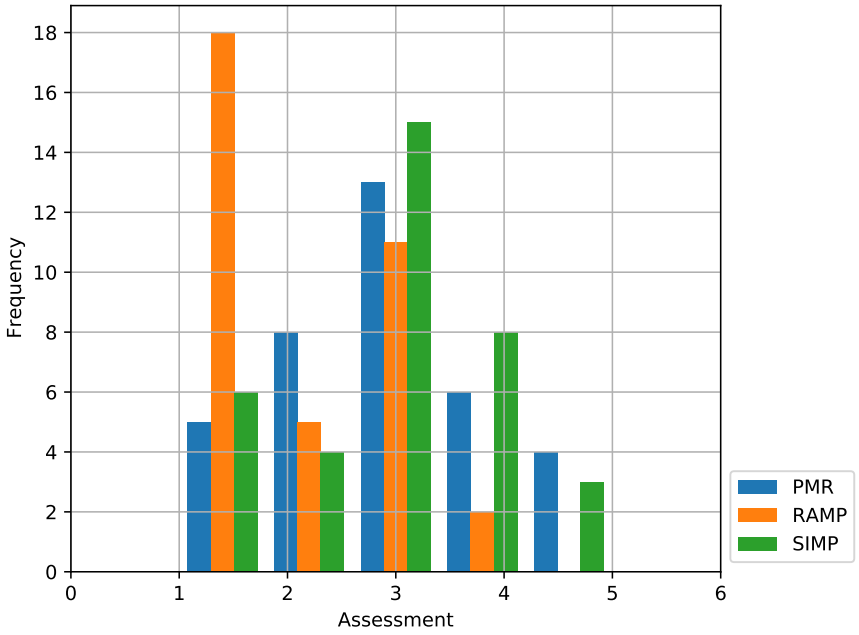


Figure 40: Histogram of the Assessments in the Volume Fraction Test

A better solution quality was expected when mesh density was increased (Hypothesis 5). The benchmark data indicated that the different chosen mesh density groups did not have the same solution quality average ($F = 2.66, p = 0.0383$).

The mean assessment values were bigger the more elements were used, except for the 40,000-elements mesh, which had a worse mean assessment than the 20,000-elements mesh. However, the pairwise comparison exclusively implied that better results are achieved with 80,000 elements than with 5,000 elements. These qualitative results were similar to the quantitative solution quality results discussed previously. It was concluded that first good solutions can be achieved when using 20,000 to 40,000 elements (assessment means: 3.39 and 3.30). The default value in all other analyses was 40,000 elements.

The visual solution quality was not significantly different for the different element types ($F = 0.85, p = 0.5146$). Thus, Hypothesis 7 could not be supported. This similarity of averages does not implicate a visual resemblance of the final structures. The final structures slightly changed visually if different element formulations were used. The similarity should rather indicate that all different element types can be used without a loss in solution quality. Also mixed-shape meshes should work well (see section 5.2.3). Stiffer results with a similar visual quality and similar computational costs might be achieved when using CPS4 instead of default CPS4R. The advantages of using a higher order interpolation function (p-method) or a finer mesh (h-method) should be evaluated individually because no significant results were found to promote one over the other. However, using eight-node elements in a 40,000-elements mesh resulted in about 120,000 total nodes, which increased the computational costs. These cases had an average assessment of 3.26. If an 80,000 CPS4R elements mesh was used, the mean assessment was 3.67. Hence, the designer could start to search for a better solution with refining mesh density, but should ideally try both methods of mesh refinement.

### 5.2.3 Mesh Distortion

The effects of mesh distortion were tested on three different meshes for each of the six sample cases as described in section 4.1.6. All three meshes had about 40,000 CPS4R elements with default settings. The degree of distortion differed between the sample cases due to different domain dimensions. The elements of the MBB Beam Case were the most distorted (see Figure 12).

The main focus of this analysis was on the visual appearance of the structures. The overall solution quality was similar between the three meshes. However, the solutions were dependent on the mesh distortion. That is, the left-distorted mesh, the straight mesh, and the right-distorted mesh always generated slightly different results. These differences were, however, negligible for the most test cases. Typical variations were slightly skewed structures and parts of structures, new holes or skewed holes in the structures, or other new features, like e.g., new trusses. Some distorted meshes gave better results than the straight mesh. The most striking example is illustrated in Figure 41. The RAMP formulation was used for this example. The MBB Beam mesh, which is significantly more distorted than the other sample cases' meshes, did not show significantly more distortion dependency. Furthermore, all the schemes showed similar distortion dependency and some symmetric test cases generated the expected mirror images for the left- and right-distorted meshes.

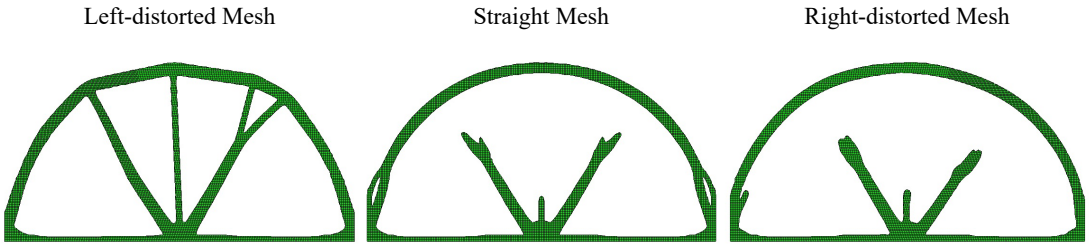


Figure 41: Distortion of Case 1 (Default Settings, RAMP Formulation)

The great resemblance of the distorted and straight mesh solutions might be caused by the high number of elements used. The high resolution might compensate the effects of mesh distortion. Nevertheless, it could be concluded that mesh distortion only has an insignificant effect on the TO solution. Mesh distortion frequently occurred if complex meshes were used. Because the effects of mesh distortion were assumed to be small, no decrease in solution quality was expected for more complicated meshes with different element shapes. For that reason, the PMR code was tested on locally refined meshes. That is, a fine mesh was used in regions of the design domain, where the final structure was assumed, and a coarse mesh elsewhere. More nodes, hence more design variables, describes the structure in a higher resolution and, thus, a higher volume fraction was required for similar results. Figure 42 shows a locally refined Half Michell Arch with about 40,000 mixed-shape elements. This approach required an involving meshing strategy and multiple tries for choosing reasonable parameter. However, good results can be achieved with locally refined meshes if used in a second, downstream design phase.

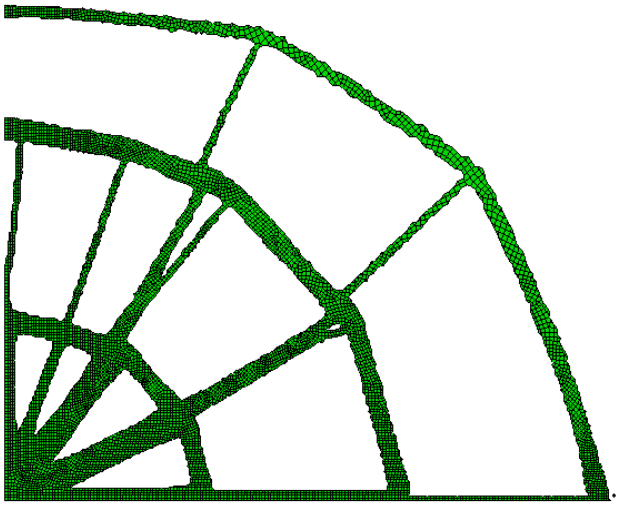

Figure 42: Half Michell Arch PMR Result with a Locally Refined Mesh

### 5.2.4 Overall Evaluation

The overall evaluation of this benchmark analyzed all 306 test cases in total, no matter which parameter was varied. The main goal of this analysis was to answer the previously formulated research questions on an ideally representative and broad group of samples.

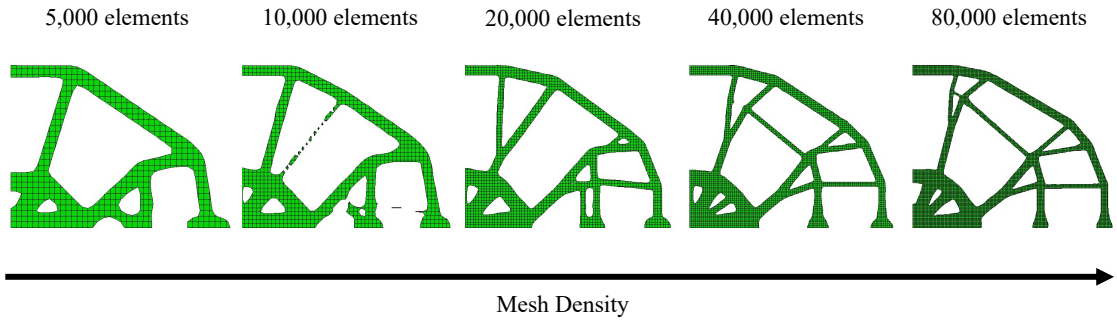


Figure 43: Half Michell Arch SIMP Results for Different Mesh Densities

First, the benchmark test cases were checked for the common problems, checkerboarding and mesh dependency. Checkerboarding patterns were not found in any of the main benchmark cases. The issue of mesh dependency was already discussed previously. Mesh distortion, varying the element type, and changing the mesh density altered the visual outcome of the TO problem for all schemes. However, most of these changes were relatively small and did not change the topology greatly. Mesh distortion and a variation of element type usually generated results that rather looked different than better or worse, based on subjective visual observations. Refining mesh density, however, was perceived to enhance the visual solution quality by creating more detailed topological structures with higher resolution (see exemplary Figure 43). This trend was, however, not completely statistically supported (see 5.2.2).

The next analysis focused on the performance difference of the three schemes. The computational efficiency of the PMR scheme was lower than in the other two

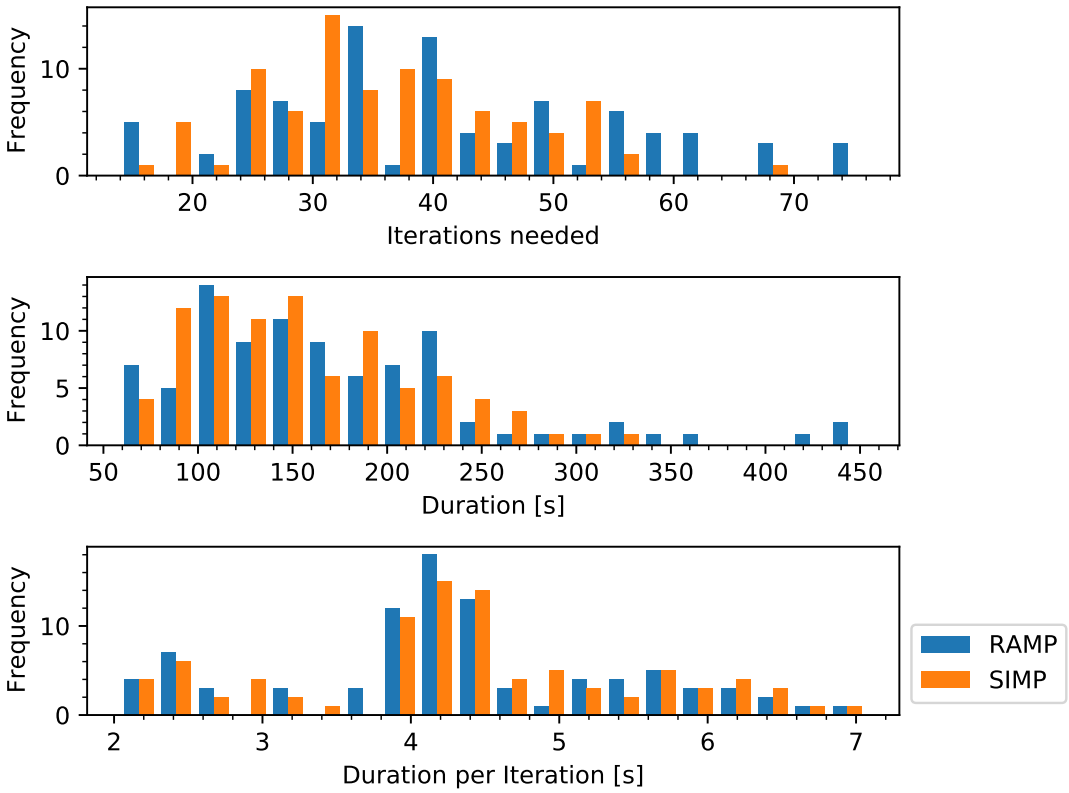


Figure 44: Performance Analysis of the SIMP and RAMP Schemes

methods in all test cases. This disadvantage was true for the total as well as the specific duration. It was discussed previously that a commercial PMR code might be competitively efficient and that the predictable duration due to the heuristic approach might be advantageous in scheduling big and intensive design problems (see sections 5.2.2 and 3.2). The similarity of the other two method's performance was tested separately. This analysis checked the distributions of total duration, specific duration, and iterations needed to converge. The resulting histograms can be found in Figure 44. The SIMP formulation needed fewer iterations to converge than the RAMP formulation ($F = 5.49, p = 0.0202$). The SIMP formulation needed 36.22 iterations on average, the RAMP scheme 40.70 iterations. However, the total durations and the specific durations of these two schemes were not significantly different ($F = 2.88, p = 0.0915$ and $F = 0.07, p = 0.7957$). Two test cases

using the RAMP formulation reached the maximum number of iterations. All other SIMP and RAMP cases converged before this limit. The average iterations needed for both schemes differed between the sample cases ($F = 8.30, p = 0.0000$), so did the total durations ($F = 3.71, p = 0.00320$), although the specific durations were similar ($F = 0.37, p = 0.8701$). Case 2 needed the fewest iterations on average (mean: 31.40) and Case 4 the most (mean: 47.20). The variations might indicate that the different sample problems were differently hard to solve.

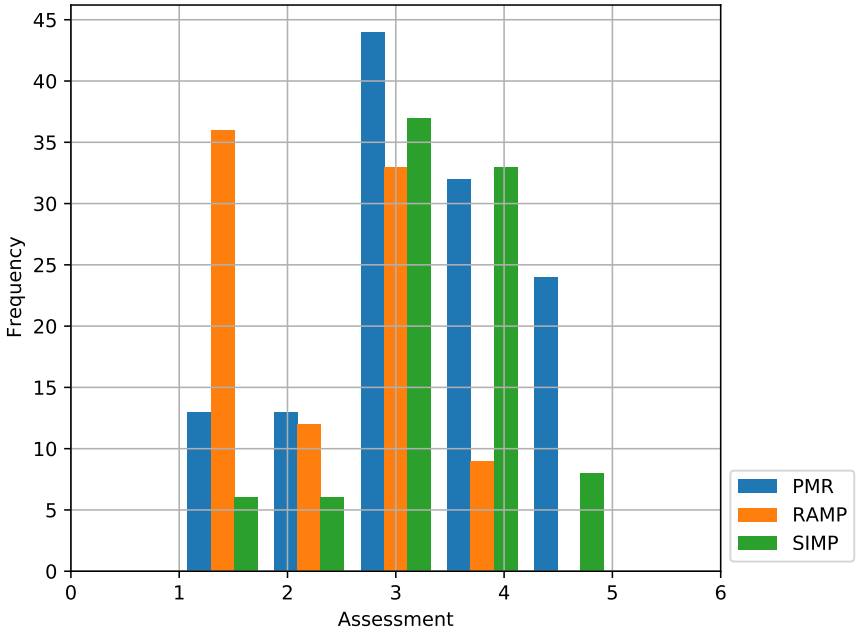


Figure 45: Assessment Histogram for the Three Different Methods

Another analysis addressed the obtained solution quality. The difference in quality for different method, for different sample cases, and for different methods for each sample case was particularly interesting. The overall visual assessments had an arithmetic mean of 3.01 (standard deviation: 1.00, median: 3.33). This distribution parameters support the assumption that the chosen quality measurement was statistically valid and the scale was symmetric. The different methods deviated in their solution quality ($F = 33.67, p = 0.0000$). The pairwise com-

parison showed that PMR and SIMP generated better results than the RAMP method ($p = 0.0010$ and $p = 0.0010$). The mean assessment of the PMR solutions was 3.29, of the SIMP solutions 3.28, and of the RAMP solutions 2.35. Figure 45 shows these assessment distributions grouped by method. The next test analyzed the assessments grouped by sample cases (see Figure 46). The sample cases differed in their average solution quality ($F = 7.90, p = 0.0000$). Case 1 had a significantly higher solution quality. The Half Michell Arch Case had the lowest mean assessment. This value might indicate this case was harder to solve because of the problem definition (e.g., three loads instead of one and low optimal volume fraction). The order of these solution quality means did not match the order of the average iterations and durations needed for the different cases (see Table 3). However, there was a slight overall trend that more iterations were needed, when better solutions were generated (see Figure 47).

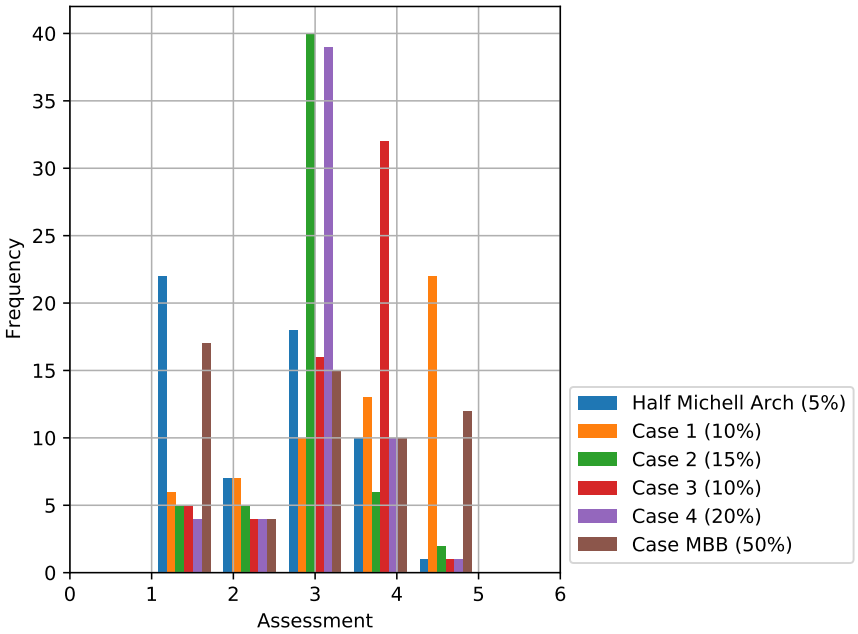


Figure 46: Assessment Histogram for the Six Different Cases

The next step was to check if one sample case was better suited for one particular method. No difference in the solution quality for the three methods was found in the cases 2 and 3 ($F = 2.71, p = 0.0754$ and $F = 0.41, p = 0.6686$). The solution quality in the other cases differed for the different methods. The pairwise comparisons indicated that the PMR and SIMP schemes gave better results than the RAMP scheme in the Half Michell Arch Case, in Case 1, and in the MBB Case. These are the same findings as in the overall analysis. The solutions of SIMP were better than of RAMP in Case 4. However, PMR and RAMP as well as PMR and SIMP did not differ significantly in that case. In summary, it was concluded the sample cases were chosen appropriately for this benchmark's purpose, despite the observed differences. It could, however, not be stated whether a specific scheme should be used for a specific type of design problem.

Table 3: Differences Between the Sample Cases

| Case | ∅ Iterations Needed | ∅ Total Duration [s] | ∅ Visual Assessment |
|---|---|---|---|
| Half Michell Arch | 35.50 | 158.55 | 2.48 |
| Case 1 | 39.33 | 165.27 | 3.52 |
| Case 2 | 31.40 | 137.59 | 2.98 |
| Case 3 | 33.30 | 139.99 | 3.29 |
| Case 4 | 47.20 | 196.57 | 3.03 |
| Case MBB | 44.03 | 189.03 | 2.99 |

Finally, the time history of the schemes was analyzed. That is, the process of density redistribution and the course of the objective value over the iterations was examined. Case 2 with default settings was chosen as an example, because its solution quality for all methods was similarly rated with about 3, which was an average assessment. Other test cases showed comparable results.

Figure 48 plots the normalized strain energy over the iteration history. Typical characteristics of the three schemes were determined. The PMR scheme is a heuristic scheme and does not converge to the minimum strain energy value. in fact, it was discovered that the PMR scheme created intermediate structures with

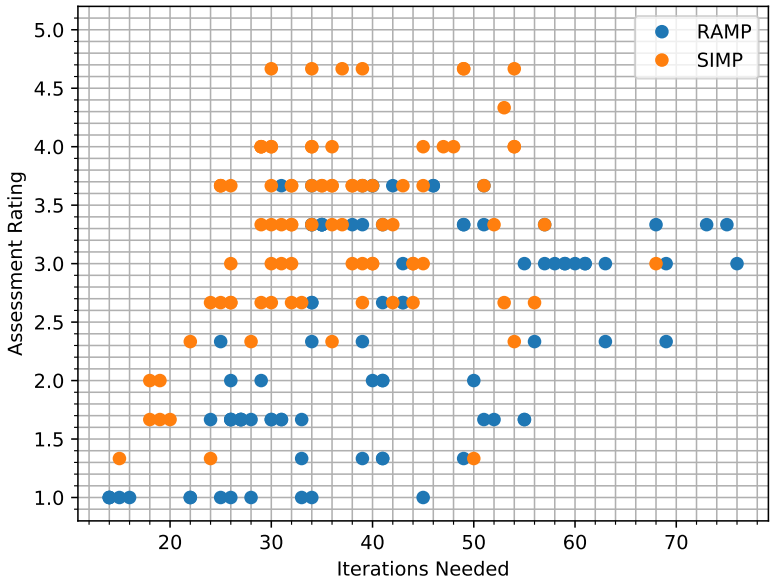


Figure 47: Iterations Needed to Converge vs. Visual Assessment

minimal strain energy, before the strain energy values increased towards the end again. These intermediate structures, however, are not manufacturable and the strain energy values are artificially computed by the linear intermediate material law. Another characteristic was the unchanging strain energy value from iteration 3 to 4, which is caused by the start of the time smoothing function (see section 3.1). The RAMP strain energy was discovered to oscillated in and out of local minima for some of the test cases. In this example the strain energy curve only had a small "dent" at about 10 iterations.

The PMR scheme had the smoothest transition in the density distribution. That is why the PMR strain energy curves were initially less steep. The SIMP and the RAMP formulation started more compliant, because their penalization functions penalize the initially uniform density distribution. However, the strain energy values dropped fast, and the structure emerged more quickly than with the PMR scheme. Sigmund and Maute [22] also stated that the SIMP algorithm

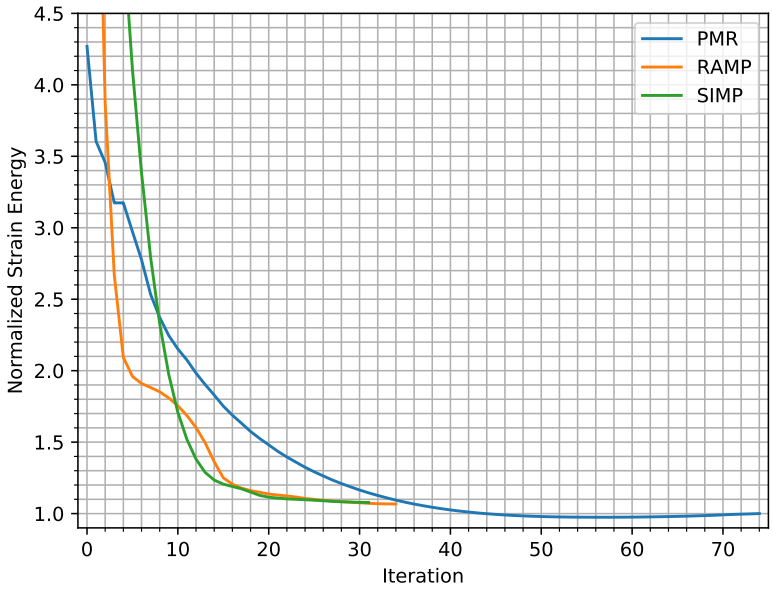


Figure 48: Normalized Strain Energy History for an Example Case

convergences fast from intermediate densities state to a solid-void design. It is, however, slow in making changes to that structure. Both effects are due to the penalization law for the intermediate densities. The PMR method could have an advantage in changing the structures because it does not penalize the intermediate material. However, the heuristic nature of the scheme would also be a disadvantage because the PMR cannot start with non-uniform initial guesses.

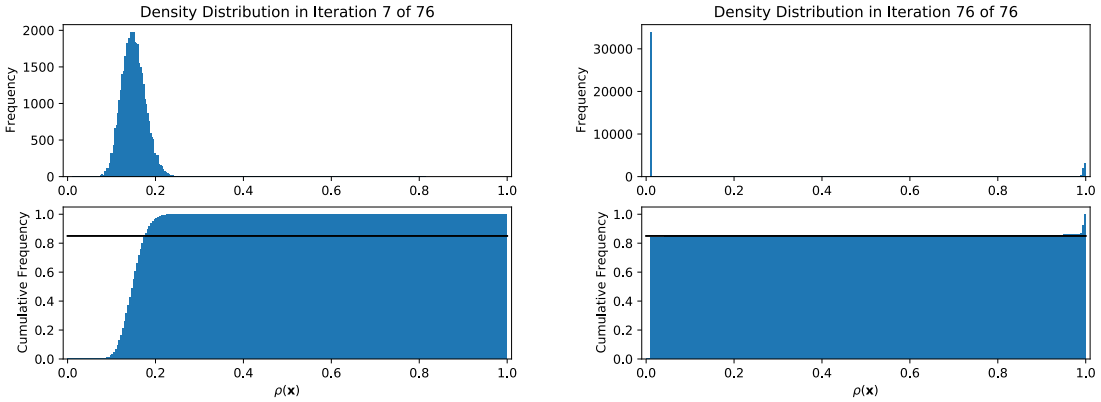


Figure 49: Density Histogram for an Example PMR Case

These described characteristics of the schemes can also be seen in Figures 49 to 51. The figures show the density distribution and the cumulative density distribution for each of the methods at about 10% of the iterations and at the last iteration of the example test case. The displayed densities of SIMP and RAMP are elemental quantities, while the PMR density is a nodal quantity. The optimal final distribution of material would look like two peaks at $\rho = 0$ and $\rho = 1$ in the top histogram and a function with two steps in the bottom cumulative histogram (see $t = 1$ in Figure 6). The amount of unwanted intermediate density in the final solution can visually be determined as the area over and under the black line in the bottom cumulative histograms. The PMR scheme came closest to this final distribution while the RAMP and SIMP formulations generated slightly more intermediate densities in the final solution. A possible reason could be that the density distribution is imposed in the PMR scheme and cannot deviate.

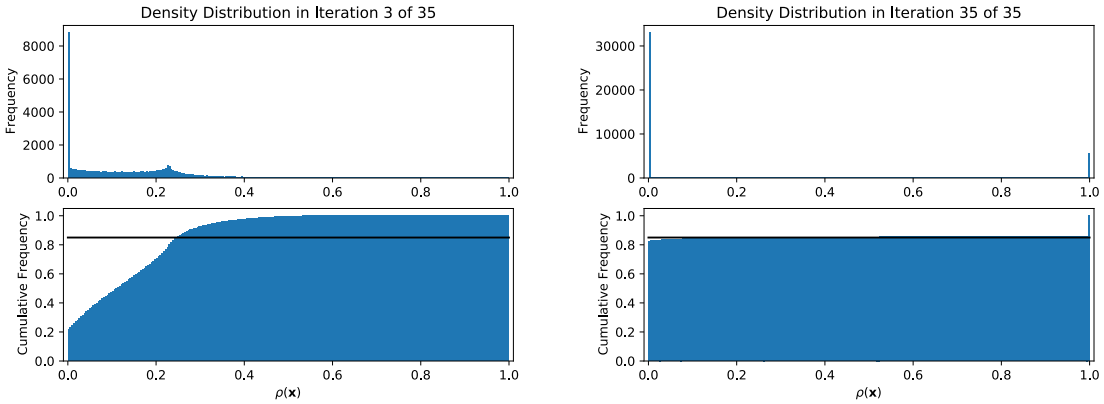


Figure 50: Density Histogram for an Example RAMP Case

These intermediate densities could also be the reason why the final PMR topologies were stiffer than the SIMP and RAMP results. The PMR scheme has fewer intermediate densities in the final structure and a linear material model, which does not "penalize" these densities. Additionally, the minimum density was chosen to be $\rho_{min} = 0.01$ for the PMR algorithm and the default Tosca SIMP value

was $\rho_{min} = 0.001$. That is, the minimum stiffness was $10^{-2}$ times smaller than the original material property for the PMR method and $10^{-9}$ times smaller for SIMP. Hence, the "void" regions of the design domain are computed multiple orders of magnitudes stiffer in the PMR algorithm.

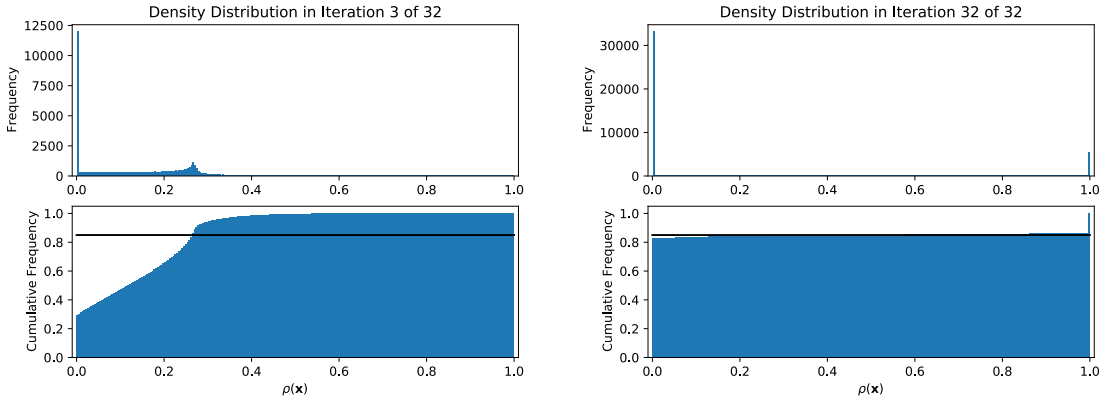


Figure 51: Density Histogram for an Example SIMP Case

Some discoveries on the three methods were made by looking at the final structure and were not qualitatively or quantitatively measured. First, the PMR code generated results that were visually the closest to the exact analytical solutions in most of the sample cases. That is, for example, the number of trusses in a fan structure was higher. However, the solution of the other methods also included the basic features of the analytical solutions. Second, the RAMP scheme was the most sensitive to a change in parameter. For example, only the finest meshed RAMP problem in the default Half Michell Arch Case generated a reasonable result. The resulting structures with coarser meshes were unusable. Finally, the SIMP algorithm was noticeable in that very complex but physically possible structures were generated in challenging setting, like e.g., very low volume fractions. The other schemes, particularly RAMP, achieved "unfinished" analytical designs, while the SIMP formulation rather found a completely different alternative structure. All

these characteristics were specifically found in this benchmark. It can, however, not be clarified if all the effects are generally valid.

# CHAPTER 6

## Conclusions and Future Research

In this chapter, the results of the research are concluded and the research questions are answered. Subsequently, the limitations of this work are discussed. The demand for further future research is then derived from the conclusions and limitations.

### 6.1 Conclusions

The fifth and final methodological step was to interpret and discuss the results of this study and, thus, to answer the posted research questions. This section follows the general structure of the research, first the enhancement findings and then the benchmark results are discussed.

The goal of translating and enhancing the original PMR code was to increase the performance and the ease of use. The computational efficiency boost was not entirely achieved. The newly developed scheme needed more time per iteration than the original standalone MATLAB version. However, the new code's performance was less sensitive to an increase in the number of nodes in the mesh. Moreover, the new scheme had fixed time costs due to the Abaqus license check and pre-processing functions. It was assumed that these Abaqus specific processes could potentially be removed or reduced in a further optimized commercial PMR implementation. Additionally, the comparison with the SIMP formulation showed that a good optimized academic MATLAB code was also faster than the Tosca Structure equivalent. Besides, the efficiency of individual PMR functions was already improved by multiple orders of magnitudes during the enhancement process. For this reason, it was argued that a commercial PMR algorithm could potentially be competitive to other methods in computational efficiency (see section 5.2.2).

The ease of use of the PMR code was noticeably enhanced. A greater variety of features and more design options were enabled through the use of the Abaqus FE solver. For example, TO problems with different element types and non-rectangular design domains can now be generated. Moreover, the direct ease of use when defining the problem was improved by utilizing the interactive GUI instead of specifying boundary and load vector explicitly. Every designer who can set up an elementary static FE model in Abaqus can use the tool and solely needs to specify two parameters, the volume fraction and the number of iterations.

Previously, it was discussed that the TO research community has developed some requirements for new schemes (see section 2.6). For example, Deaton and Grandhi [19] demanded a shift of focus from the simple minimum compliance problems to more realistic problems. Sigmund and Maute [22] required the concentration on more efficient and generally applicable methods and formulated nine particular challenges for TO schemes. The PMR code was checked on these challenges.

- Regarding efficiency, the current lack of efficiency was previously discussed. However, it was assumed that a commercial approach might be faster and that the predictability of the approach might be advantageous for scheduling large scale 3D problems.

- Regarding general applicability, the current version of the PMR method is only applicable to minimum compliance problems with a single volume constraint. However, the applicability of this heuristic approach on multi-physics problems has not been investigated and could be interesting to future research.

- Regarding multiple constraints, the feasibility of imposing multiple constraints has also not been investigated yet. Additional geometric constraints,

however, could potentially be implemented in the current PMR version, e.g., minimum length constraints (see section 2.3).

- Regarding complex boundary conditions, the present PMR version only supports static mechanical loading conditions.

- Regarding independence on starting guess, the PMR method generally imposes the uniform initial density distribution.

- Regarding few tuning parameters, one advantage of the PMR scheme is that only the number of iterations need to be chosen as a tuning parameter. However, the method is not generally applicable (see challenge 2) and, thus, no statement about problem depending parameter settings could be made.

- Regarding mesh-independent convergence, another advantage of this heuristic approach is that PMR has a fixed number of iterations and does not converge.

- Regarding ease of use, PMR requires a similar experience with FEA tools as other methods. With PMR, the use and the understandability are facilitated by the simplicity of the approach. That is, no understanding of optimization theory and convergence, of level set theory, or of artificial material models is required to understand the prescribed redistribution of material and obtain a topology. Hence, PMR could potentially be applicable in engineering education and research.

- Regarding alternatives to finite element analysis, the current PMR versions rely on FEA. Other tools for computing the strain energy fields have not been investigated.

It can be summarized that the current PMR version cannot comply with the nine challenges on general applicable and sophisticated TO software. However, the heuristic approach could potentially perform well in the minimum compliance niche. Further investigations are needed on PMR's general applicability.

The second part of this research was to conduct the benchmark of the PMR, RAMP and SIMP method. The efficiency as well as the quantitative and qualitative measures of the solution quality were examined. The PMR was found to be slower than the other method as discussed previously. The Tosca Structure's SIMP and RAMP formulation were similar in efficiency, which was expected because the formulations solely deviate in their material penalization model. These two methods also showed similarly stiff final results. The final PMR structures, however, where constantly stiffer, although the scheme does not include a convergence criteria. Two explanations, why these solutions may be stiffer than the SIMP and RAMP results were introduced in section 5.2.4. The visual solution quality analysis showed deviating results. The RAMP formulation generated worse results than the SIMP and PMR methods. The solution of the SIMP and PMR approaches were very similar. The PMR results were visually often closer to the expected analytical solution of the problems. The SIMP solution tended to find good practical and alternative structures for complicated problems. Moreover, SIMP converged quickly to a solid-void state, while the PMR imposed a smoother density distribution transition.

The conclusion of the benchmark was very similar to the previous discussed nine challenges and some results were already stated previously. SIMP is generally well-researched and applicable for many settings. The formulation is also part of the optimized and dedicated TO software. PMR, however, is a good heuristic scheme in the "minimum compliance under a single volume constraint" niche if

further optimized for more efficiency. Some advantages of PMR are resemblance of the results to the analytical solution and its duration predictability, which makes it easier to schedule big problems. Moreover, the PMR scheme is quite simple. That is, the PMR method is most likely easier to understand than sophisticated optimization approaches and needs few parameters. Hence, the PMR algorithm could be used for engineering education purposes. However, the current version is slower than the other software tools. The presumably biggest disadvantage of the PMR method is the focus on the compliance minimization problem. In this field, the heuristic approach was able compete with the solution quality of the dedicated optimization methods. TO generally finds first practical solutions, which need further analysis, such as buckling analysis, to obtain production-ready designs. Hence, the heuristic approach of PMR, potentially in a hybrid software tool, similar to the hard-kill methods (see section 2.2), could give sufficient results for many practical situations.

The parametric benchmark study also provided information on reasonable settings and characteristics of the three tested methods. The reasonable number of iterations for the PMR method was chosen to be about 75. In some cases, better results might be obtained with more iterations. A finer mesh might give better results for all methods, but the calculations will take more time. These meshes also generated stiffer results. This result is consistent with the findings of Bulman et al. [17]. A default mesh with 40,000 elements was chosen based on the findings (see section 5.2.2), which is a finer resolution than the minimum mesh proposal by Valdez et al. with 10,000 elements [38]. Different element types can be used in the meshes. Higher order elements might, however, increase the computational costs. Moreover, no checkerboarding was found for the tested TO schemes and mesh distortion did not have great impact on the visual solution quality. Hence,

mixed-shape, locally refined meshes could be used. The mesh distortion results were similar but slightly less distinct than in the study by Bulman et al. [17].

## 6.2 Limitations

Some assumptions formulated and accepted in this research limit the validity of the results. It was assumed that the data complied with the premises for all conducted statistical tests (see section 4.2). The group size of the sampling in particular was, however, smaller in some tests than typically required. Additionally, normality and homoscedasticity were not proven. However, most of the significant test results statistically supported trends that were expected beforehand and/or could visually be determined from the data plots. The statistical tests were used as a tool to potentially quantify these effects. It could be assumed that similar effects would be discovered with more data and satisfied assumptions.

Another critical assumption was the validity of the visual quality assessment. The measurement might have been biased towards a specific method. However, this issue was attempted to be ruled out by averaging three persons' ratings. The two other engineering students rated the test cases and were not aware of the TO method and other settings during the process (see section 4.2).

A more quantitative effect that might of have distorted the data was the effect of decreasing performance in a loop execution (see section 5.1.1). This effect was assumed to be negligible and this assumption seemed to hold. Comparing the order of parameters in the loop with the discovered trends in the duration data showed that the influences of parameter change were much greater and/or the loop effect was absence. For example, the mesh was tested in descending order, but the observed linear trend was the other way around.

Some results were not consistent with literature. Valdez et al. [38] state that in their study adapting the convergence criteria to the individual cases was

difficult and, thus, the benchmarked original SIMP algorithm commonly reached the maximum number of iterations, which was defined to be 150. The threshold in this benchmark was 75 iterations and was only reached two times by RAMP cases. The average SIMP algorithm case needed 36.22 iterations. Although a similar convergence criterion is used, it is difficult to argue why the SIMP algorithm this study used deviated from the SIMP algorithm the other study used.

Different SIMP algorithms use different solvers and other deviating settings to solve an optimization problem that includes the SIMP specific material model (see sections 2.2.1 and 2.4). That is, there is no single "SIMP algorithm", but multiple algorithms solving similar versions of the SIMP problem. For example, one benchmark study by Rojas-Labanda and Stolpe [39] focused specifically on analyzing different solvers. The results of this benchmark, however, particularly compare the Tosca Structure's RAMP and SIMP implementation using the MMA to the heuristic PMR scheme, because all three schemes use the same FE solver, Abaqus. However, it is arguable whether all findings have general validity for any RAMP and SIMP implementation.

## 6.3 Future Research

In the future, further research could be conducted on both parts of this study, on the enhancement and on the benchmark. The research on the benchmark could rule out some of the limitations. First, more test cases could be run to obtain a better statistical foundation by making the sampling sizes bigger. Additionally, the resolution of certain effects would be greater if more values for each parameter were tried, e.g., volume fractions between 25% and 50%. Potentially, more and other effects could be evaluated with this data. Furthermore, more people could visually assess the resulting structures to reduce a potential bias. These people could maybe have different theoretical backgrounds. The benchmark should also

be conducted with different settings and different convergence criteria to determine associated effects. Another important research focus would be to replicate this benchmark for other software and algorithms to check the general validity of the findings. Future research could also conduct a similar benchmark with the boundary variation schemes, because they are fairly new and are recently getting more popular (see section 2.2.4).

Further research on the PMR scheme could improve the method and bring it closer to commercial viability. For this, the feasibility of certain changes to the scheme and the effects of these new features need to be investigated. For example, it should be checked whether a minimization of volume with a stress constraint analogous to a study by Valdez et al. [38] is feasible. Another test could analyze the potential decoupling of design variable and the FE mesh, introduced shortly in section 2.2. One topology optimization element does not necessarily have to be one element in the FE mesh. Using this approach for the PMR scheme could potentially optimize efficiency or solution resolution. Moreover, the effects of a different material model for the PMR code should be tested. It would be interesting to see whether a penalization law similar to the SIMP formulation would give comparable results with fewer iterations. Furthermore, two ideas that are also found in other articles (e.g., [6] and [53]) could be tested in a similar manner, adaptive remeshing and removing void elements. The former is the fundamental idea of hybrid methods (see section 2.2). The foundations for adaptive remeshing have already been set by analyzing locally refined meshes. Hence, both ideas have a potential to be feasible extensions to the current PMR code. The initial PMR version has an option to predefine certain nodes to be fully dense or fully void and, thus, purposely exclude them from the design domain. This feature was not

implemented in current Abaqus PMR version. However, future research could realize a similar functionality for the new algorithm if needed.

Two of the presumably most discussed and important characteristics of TO schemes are general applicability and production-readiness. First, the lack of general applicability of the PMR was discussed previously (see section 6.1). Bendsøe and Sigmund [21] argued that the fundamental idea of homogenization methods is applicable to other physical properties than elasticity as well. PMR could be categorized as a homogenization method. Future research could investigate whether the PMR-specific heuristic transition of density with beta distributions is applicable to multi-physics problems for which sensitives might not be strictly negative (see section 2.6). Second, some authors demand the ability to impose (additive) manufacturing constraints on the TO algorithm to obtain designs with a higher production-readiness (e.g., [5]). For example, a minimum length constraint, similar to the constraint examined in a study by Guest et al. [34], could potentially be feasible and relatively straightforward to implement into the current PMR code. After further improvement, the commercial viability of the PMR scheme as a heuristic approach to TO optimization should be reevaluated.

## LIST OF REFERENCES


[1] D. Taggart and P. Dewhurst, "Development and validation of a numerical topology optimization scheme for two and three dimensional structures," *Advances in Engineering Software*, vol. 41, no. 7-8, pp. 910–915, 2010.

[2] A. V. Pichugin, A. Tyas, and M. Gilbert, "Michell structure for a uniform load over multiple spans," in *9th World Congress on Structural & Multidisciplinary Optimization*. Sheffield, 2011.

[3] D. G. Taggart, P. Dewhurst, L. Dobrot, and D. D. Gill, "Development of a beta function based topology optimization procedure," in *2008 Abaqus Users Conference, Newport, RI*, 2008.

[4] G. Rozvany, "Exact analytical solutions for some popular benchmark problems in topology optimization," *Structural optimization*, vol. 15, no. 1, pp. 42–48, 1998.

[5] S. N. Reddy K, I. Ferguson, M. Frecker, T. W. Simpson, and C. J. Dickman, "Topology optimization software for additive manufacturing: A review of current capabilities and a real-world example," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2016.

[6] D. Brackett, I. Ashcroft, and R. Hague, "Topology optimization for additive manufacturing," in *Proceedings of the solid freeform fabrication symposium, Austin, TX*, vol. 1, 2011, pp. 348–362.

[7] T. Zegard and G. H. Paulino, "Bridging topology optimization and additive manufacturing," *Structural and Multidisciplinary Optimization*, vol. 53, no. 1, pp. 175–192, 2016.

[8] A. Chandrasekhar and K. Suresh, "Tounn: Topology optimization using neural networks," *Structural and Multidisciplinary Optimization*, vol. 63, no. 3, pp. 1135–1149, 2021.

[9] Y. Xu, J. Zhu, Z. Wu, Y. Cao, Y. Zhao, and W. Zhang, "A review on the design of laminated composite structures: constant and variable stiffness design and topology optimization," *Advanced Composites and Hybrid Materials*, vol. 1, no. 3, pp. 460–477, 2018.

[10] G. I. Rozvany, "A critical review of established methods of structural topology optimization," *Structural and multidisciplinary optimization*, vol. 37, no. 3, pp. 217–237, 2009.

[11] D. L. Logan, *A first course in the finite element method*. Cengage Learning, 2016.

[12] K.-J. Bathe, *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[13] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method: its basis and fundamentals*. Elsevier, 2005.

[14] T. J. Hughes, *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[15] SIMULIA by Dassault Systems. "Abaqus CAE." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.3ds.com/products-services/simulia/products/abaqus/abaquscae/

[16] SIMULIA by Dassault Systems. "Abaqus documentation." Accessed on 07/08/2021. 2016. [Online]. Available: http://130.149.89.49:2080/v2016/index.html

[17] S. Bulman, J. Sienz, and E. Hinton, "Comparisons between algorithms for structural topology optimization using a series of benchmark studies," *Computers & structures*, vol. 79, no. 12, pp. 1203–1218, 2001.

[18] A. G. M. Michell, "Lviii. the limits of economy of material in frame-structures," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 47, pp. 589–597, 1904.

[19] J. D. Deaton and R. V. Grandhi, "A survey of structural and multidisciplinary continuum topology optimization: post 2000," *Structural and Multidisciplinary Optimization*, vol. 49, no. 1, pp. 1–38, 2014.

[20] X. Guo and G.-D. Cheng, "Recent development in structural design and optimization," *Acta Mechanica Sinica*, vol. 26, no. 6, pp. 807–823, 2010.

[21] M. P. Bendsøe and O. Sigmund, "Material interpolation schemes in topology optimization," *Archive of applied mechanics*, vol. 69, no. 9, pp. 635–654, 1999.

[22] O. Sigmund and K. Maute, "Topology optimization approaches," *Structural and Multidisciplinary Optimization*, vol. 48, no. 6, pp. 1031–1055, 2013.

[23] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, no. 4, pp. 193–202, 1989.

[24] G. I. Rozvany, M. Zhou, and T. Birker, "Generalized shape optimization without homogenization," *Structural optimization*, vol. 4, no. 3-4, pp. 250–252, 1992.

[25] M. Stolpe and K. Svanberg, "An alternative interpolation scheme for minimum compliance topology optimization," *Structural and Multidisciplinary Optimization*, vol. 22, no. 2, pp. 116–124, 2001.

[26] K. Svanberg, "The method of moving asymptotes—a new method for structural optimization," *International journal for numerical methods in engineering*, vol. 24, no. 2, pp. 359–373, 1987.

[27] T. Bruns, "A reevaluation of the simp method with filtering and an alternative formulation for solid–void topology optimization," *Structural and Multidisciplinary Optimization*, vol. 30, no. 6, pp. 428–436, 2005.

[28] Y. M. Xie and G. P. Steven, "A simple evolutionary procedure for structural optimization," *Computers & structures*, vol. 49, no. 5, pp. 885–896, 1993.

[29] O. M. Querin, G. P. Steven, and Y. M. Xie, "Evolutionary structural optimisation (eso) using a bidirectional algorithm," *Engineering computations*, 1998.

[30] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.

[31] M. Y. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Computer methods in applied mechanics and engineering*, vol. 192, no. 1-2, pp. 227–246, 2003.

[32] V. J. Challis, "A discrete level-set topology optimization code written in matlab," *Structural and multidisciplinary optimization*, vol. 41, no. 3, pp. 453–464, 2010.

[33] A. Shukla, A. Misra, and S. Kumar, "Checkerboard problem in finite element based topology optimization," *International Journal of Advances in Engineering & Technology*, vol. 6, no. 4, p. 1769, 2013.

[34] J. K. Guest, J. H. Prévost, and T. Belytschko, "Achieving minimum length scale in topology optimization using nodal design variables and projection functions," *International journal for numerical methods in engineering*, vol. 61, no. 2, pp. 238–254, 2004.

[35] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, "Efficient topology optimization in matlab using 88 lines of code," *Structural and Multidisciplinary Optimization*, vol. 43, no. 1, pp. 1–16, 2011.

[36] A. Diaz and O. Sigmund, "Checkerboard patterns in layout optimization," *Structural optimization*, vol. 10, no. 1, pp. 40–45, 1995.

[37] D. G. Taggart, P. Dewhurst, and A. U. Nair, "Systems and methods for finite element based topology optimization," Dec. 18 2012, US Patent 8,335,668.

[38] S. I. Valdez, S. Botello, M. A. Ochoa, J. L. Marroquín, and V. Cardoso, "Topology optimization benchmarks in 2d: Results for minimum compliance and minimum volume in planar stress problems," *Archives of Computational Methods in Engineering*, vol. 24, no. 4, pp. 803–839, 2017.

[39] S. Rojas-Labanda and M. Stolpe, "Benchmarking optimization solvers for structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 52, no. 3, pp. 527–547, 2015.

[40] T. Lewiński and G. Rozvany, "Exact analytical solutions for some popular benchmark problems in topology optimization ii: three-sided polygonal supports," *Structural and Multidisciplinary Optimization*, vol. 33, no. 4, pp. 337–349, 2007.

[41] T. Lewiński and G. Rozvany, "Exact analytical solutions for some popular benchmark problems in topology optimization iii: L-shaped domains," *Structural and Multidisciplinary Optimization*, vol. 35, no. 2, pp. 165–174, 2008.

[42] T. Lewiński, G. Rozvany, T. Sokół, and K. Bołbotowski, "Exact analytical solutions for some popular benchmark problems in topology optimization iii: L-shaped domains revisited," *Structural and Multidisciplinary Optimization*, vol. 47, no. 6, pp. 937–942, 2013.

[43] SIMULIA by Dassault Systems. "Abaqus documentation - the optimization task." Accessed on 07/08/2021. 2016. [Online]. Available: https://abaqus-docs.mit.edu/2017/English/TsoUserMap/tso-c-usr-sizing-optTask.htm

[44] O. Sigmund, "A 99 line topology optimization code written in matlab," *Structural and multidisciplinary optimization*, vol. 21, no. 2, pp. 120–127, 2001.

[45] TopOpt. "Apps/software - topopt." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.topopt.mek.dtu.dk/apps-and-software

[46] C. S. Andreasen, M. O. Elingaard, and N. Aage, "Level set topology and shape optimization by density methods using cut elements with length scale control," *Structural and Multidisciplinary Optimization*, pp. 1–23, 2020.

[47] The SciPy community. "Numpy documentation." Accessed on 07/08/2021. 2021. [Online]. Available: https://numpy.org/doc/stable/

[48] SciPy developers. "Scipy documentation." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.scipy.org/docs.html

[49] SIMULIA by Dassault Systems. "Abaqus scripting user's guide." Accessed on 07/08/2021. 2021. [Online]. Available: http://130.149.89.49:2080/v6.14/books/cmd/default.htm

[50] X. Cai, H. P. Langtangen, and H. Moe, "On the performance of the python programming language for serial and parallel scientific computations," *Scientific Programming*, vol. 13, no. 1, pp. 31–56, 2005.

[51] A. Dean, D. Voss, D. Draguljić, *et al.*, *Design and analysis of experiments*, 2nd ed. Springer, 2007.

[52] T. Haslwanter, *An Introduction to Statistics with Python*. Springer, 2016.

[53] A. Aremu, I. Ashcroft, R. Hague, R. Wildman, and C. Tuck, "Suitability of simp and beso topology optimization algorithms for additive manufacture," in *21st annual international solid freeform fabrication symposium (SFF)–An additive manufacturing conference*, 2010, pp. 679–692.

# APPENDIX A

## Boundary Conditions of the Sample Cases

### A.1 Benchmark Sample Cases

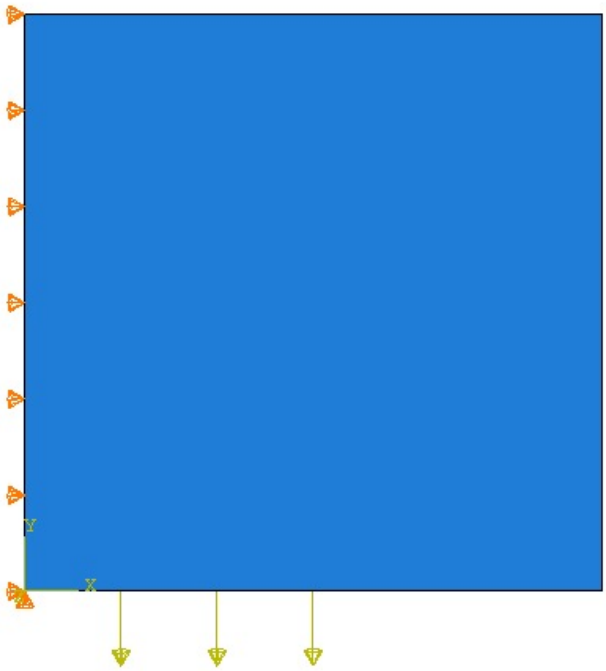


Figure A.52: Half Michell Arch: Loading and Boundary Conditions

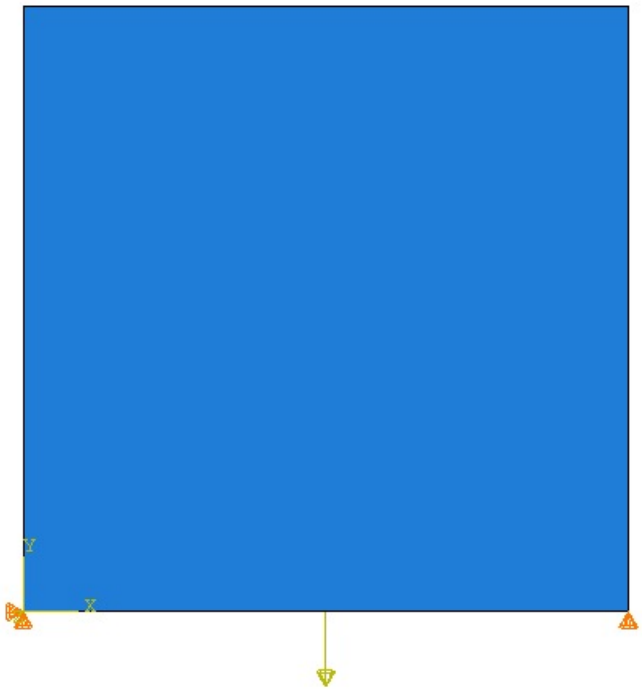

Figure A.53: Case 1: Loading and Boundary Conditions

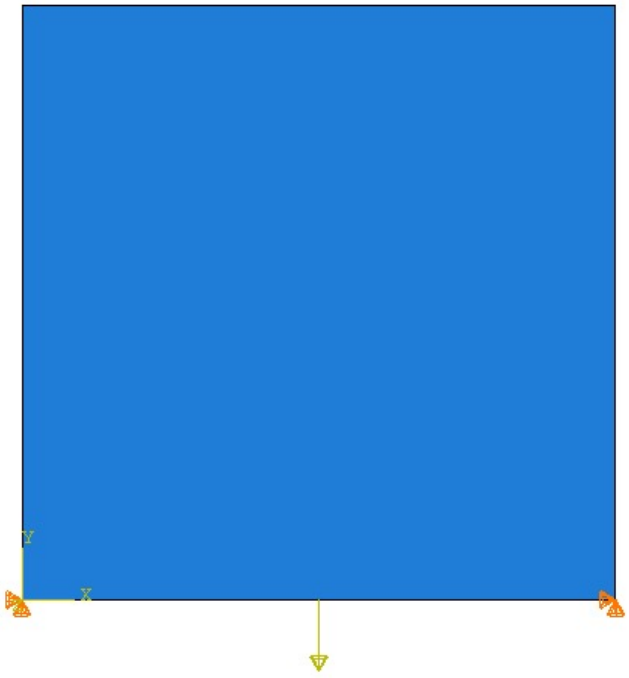

Figure A.54: Case 2: Loading and Boundary Conditions

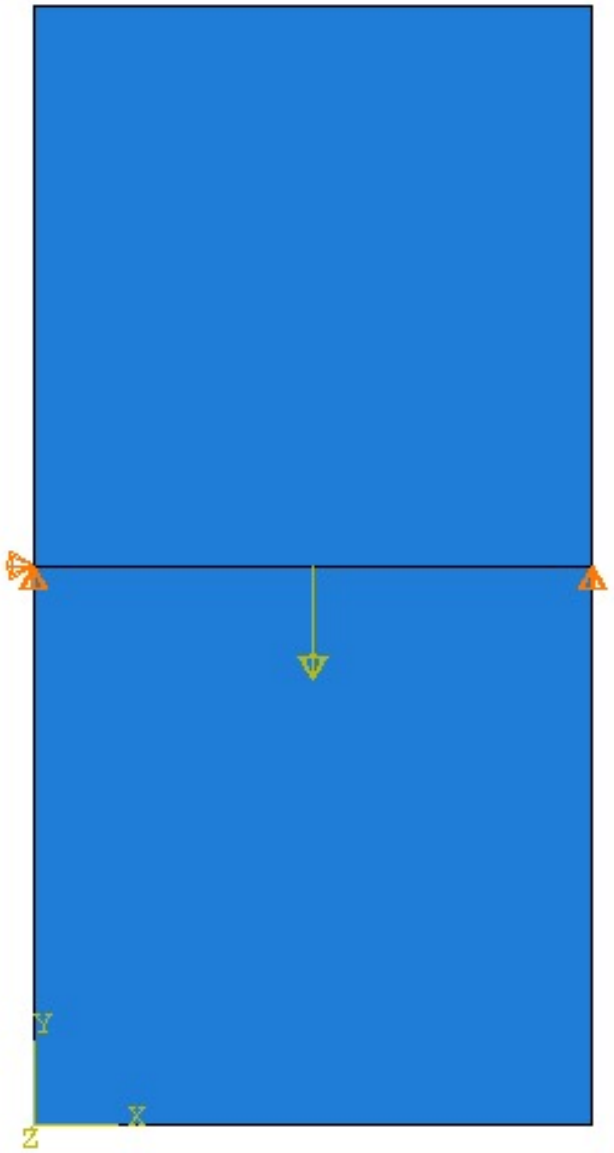


Figure A.55: Case 3: Loading and Boundary Conditions

Figure A.56: Case 4: Loading and Boundary Conditions

Figure A.57: Case MBB: Loading and Boundary Conditions

## A.2 Auxiliary Sample Cases

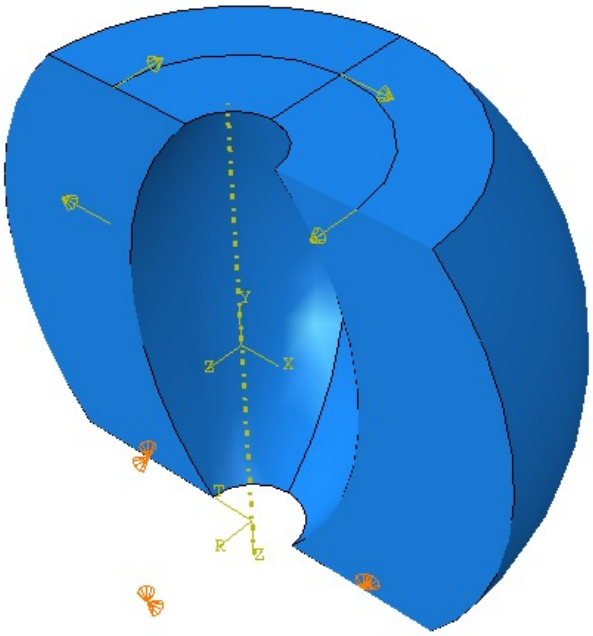

Figure A.58: 3D Loxodrome: Loading and Boundary Conditions

Figure A.59: Pressure Case: Loading and Boundary Conditions

# APPENDIX B

## Additional Figures

### B.1 Integration Point-based PMR Scheme

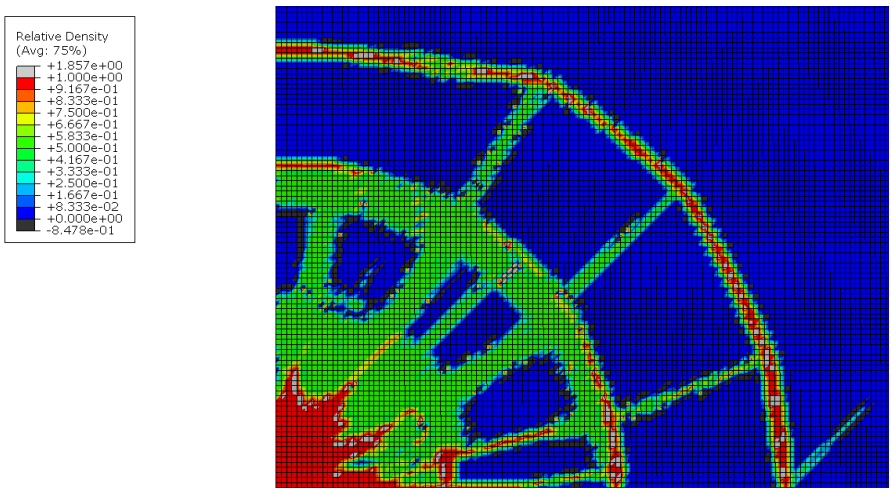


Figure B.60: Integration Point-based PMR Half Michell Arch Result (CPS4)

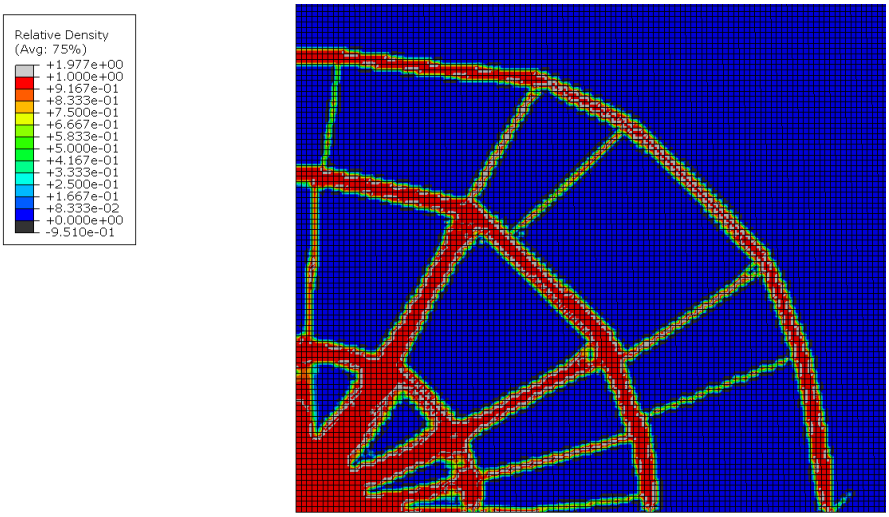


Figure B.61: Integration Point-based PMR Half Michell Arch Result (CPS8R)

## B.2 Varying the Number of Iterations

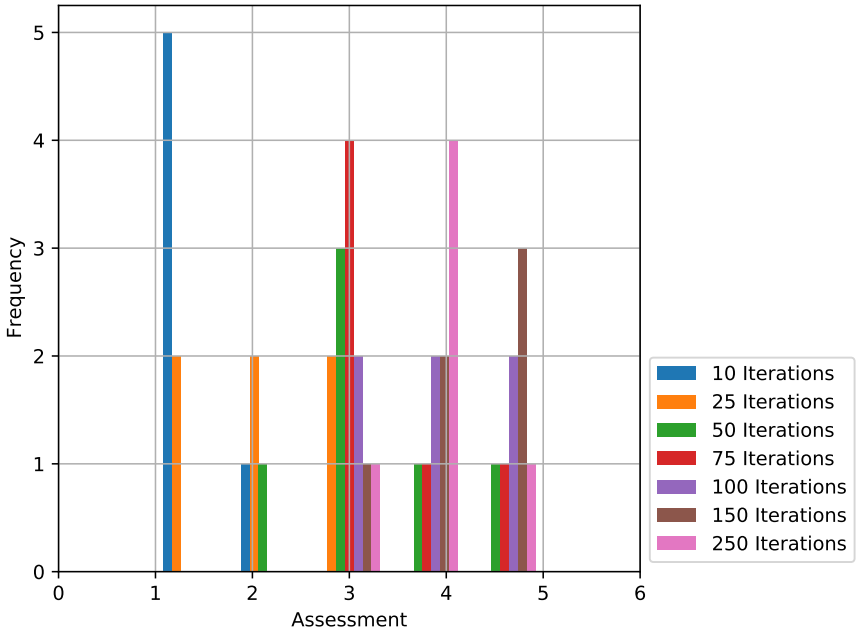


Figure B.62: Assessment Histogram Varying the Number of Iterations

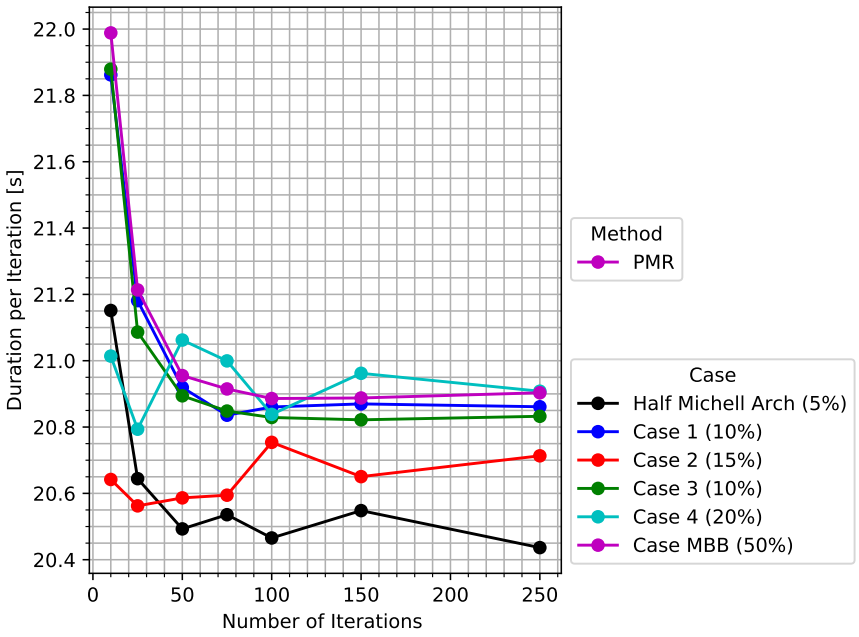


Figure B.63: Specific Duration vs. Number of Iterations (PMR)

# APPENDIX C

## Additional Tables

Table C.4: Edge Seeds to Generate Different Fine Meshes

| Number of Elements | Half Michell Arch | Case 1 | Case 2 | Case 3 | Case 4 | Case MBB |
|---|---|---|---|---|---|---|
| 80,000 (Quad) | 0.00053 | 0.3535 | 0.3535 | 0.5 | 0.53 | 0.11174 |
| 40,000 (Quad) | 0.00075 | 0.5 | 0.5 | 0.707 | 0.750 | 0.1575 |
| 40,000 (Tri) | 0.00106 | 0.707 | 0.707 | 1.0 | 1.061 | 0.224 |
| 20,000 (Quad) | 0.00106 | 0.707 | 0.707 | 1.0 | 1.061 | 0.224 |
| 10,000 (Quad) | 0.0015 | 1.0 | 1.0 | 1.41 | 1.505 | 0.316 |
| 5,000 (Quad) | 0.00215 | 1.41 | 1.41 | 2.0 | 2.12 | 0.445 |

Table C.5: Parameter Used to Generate Level Set Results

| Parameter | Scheme by Challis | Scheme by TopOpt Group |
|---|---|---|
| # Elements in $x$ Direction | 70 | 200 |
| # Elements in $y$ Direction | 70 | 200 |
| Volume Fraction | 7% | 8% |
| Additional Parameters | stepLength = 12<br>numReinit = 6<br>topWeight 3 | optimization delta = 0.01 |

Table C.6: Results of the ANOVAs

| Test object | F | p | Means |
|---|---|---|---|
| Hypothesis 1 | 15.444596 | 0.000000 | 1.33, 2.28, 3.17, 3.56, 3.78, 3.94 |
| Hypothesis 3 | 8.545259 | 0.000364 | 2.92, 2.145, 2.99 |
| Hypothesis 5 | 2.656151 | 0.038340 | 3.67, 3.30, 3.39, 3.17, 2.74 |
| Hypothesis 7 | 0.854370 | 0.514597 | 3.30, 3.43, 3.26, 3.28, 2.87, 3.09 |
| Iterations needed by method | 5.492543 | 0.020202 | 40.70, 36.22 |
| Duration by method | 2.878206 | 0.091534 | 173.57 s, 155.43 s |
| Specific duration by method | 0.067248 | 0.795686 | 4.30 s, 4.35 s |
| Iterations needed by case | 8.297426 | 0.000000 | 35.50, 39.33, 31.40, 33.30, 47.20, 44.03 |
| Duration by case | 0.367856 | 0.870105 | 4.50 s, 4.36 s, 4.45 s, 4.22 s, 4.19 s, 4.23 s |
| Specific duration by case | 3.710552 | 0.003237 | 158.55 s, 165.27 s, 137.59 s,<br>139.99 s, 196.57 s, 189.03 s |
| Difference of methods | 33.672318 | 0.000000 | 3.29, 2.35, 3.28 |
| Difference of cases | 7.896284 | 0.000000 | 2.48, 3.52, 2.98, 3.29, 3.03, 2.99 |
| Methods (Half Michell) | 16.880932 | 0.000002 | 2.99, 1.51, 2.75 |
| Methods (Case 1) | 35.146339 | 0.000000 | 4.15, 2.18, 3.98 |
| Methods (Case 2) | 2.710612 | 0.075381 | 3.24, 2.84, 2.76 |
| Methods (Case 3) | 0.405464 | 0.668643 | 3.26, 3.20, 3.41 |
| Methods (Case 4) | 5.638141 | 0.005924 | 2.96, 2.757, 3.41 |
| Methods (Case MBB) | 21.712859 | 0.000000 | 3.33, 1.76, 3.73 |

Table C.7: Results of the Tukey HSDs 1/2

| Test object | group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|---|
| Hypothesis 1 | 10 Iterations | 25 Iterations | 0.9444 | 0.1477 | -0.1779 | 2.0668 | False |
| | 10 Iterations | 50 Iterations | 1.8333 | 0.001 | 0.711 | 2.9557 | True |
| | 10 Iterations | 75 Iterations | 2.2222 | 0.001 | 1.0998 | 3.3446 | True |
| | 10 Iterations | 100 Iterations | 2.4444 | 0.001 | 1.3221 | 3.5668 | True |
| | 10 Iterations | 150 Iterations | 2.6111 | 0.001 | 1.4887 | 3.7335 | True |
| | 10 Iterations | 250 Iterations | 2.6667 | 0.001 | 1.5443 | 3.789 | True |
| | 25 Iterations | 50 Iterations | 0.8889 | 0.1992 | -0.2335 | 2.0113 | False |
| | 25 Iterations | 75 Iterations | 1.2778 | 0.0171 | 0.1554 | 2.4002 | True |
| | 25 Iterations | 100 Iterations | 1.5 | 0.0032 | 0.3776 | 2.6224 | True |
| | 25 Iterations | 150 Iterations | 1.6667 | 0.001 | 0.5443 | 2.789 | True |
| | 25 Iterations | 250 Iterations | 1.7222 | 0.001 | 0.5998 | 2.8446 | True |
| | 50 Iterations | 75 Iterations | 0.3889 | 0.9 | -0.7335 | 1.5113 | False |
| | 50 Iterations | 100 Iterations | 0.6111 | 0.6045 | -0.5113 | 1.7335 | False |
| | 50 Iterations | 150 Iterations | 0.7778 | 0.3391 | -0.3446 | 1.9002 | False |
| | 50 Iterations | 250 Iterations | 0.8333 | 0.263 | -0.289 | 1.9557 | False |
| | 75 Iterations | 100 Iterations | 0.2222 | 0.9 | -0.9002 | 1.3446 | False |
| | 75 Iterations | 150 Iterations | 0.3889 | 0.9 | -0.7335 | 1.5113 | False |
| | 75 Iterations | 250 Iterations | 0.4444 | 0.8637 | -0.6779 | 1.5668 | False |
| | 100 Iterations | 150 Iterations | 0.1667 | 0.9 | -0.9557 | 1.289 | False |
| | 100 Iterations | 250 Iterations | 0.2222 | 0.9 | -0.9002 | 1.3446 | False |
| | 150 Iterations | 250 Iterations | 0.0556 | 0.9 | -1.0668 | 1.1779 | False |
| Hypothesis 3 | PMR | RAMP | -0.7685 | 0.0027 | -1.3052 | -0.2319 | True |
| | PMR | SIMP | 0.0741 | 0.9 | -0.4626 | 0.6107 | False |
| | RAMP | SIMP | 0.8426 | 0.001 | 0.3059 | 1.3792 | True |
| Hypothesis 5 | 10k Elements | 20k Elements | 0.2222 | 0.9 | -0.5991 | 1.0436 | False |
| | 10k Elements | 40k Elements | 0.1296 | 0.9 | -0.6917 | 0.951 | False |
| | 10k Elements | 5k Elements | -0.4259 | 0.5876 | -1.2473 | 0.3954 | False |
| | 10k Elements | 80k Elements | 0.5 | 0.443 | -0.3213 | 1.3213 | False |
| | 20k Elements | 40k Elements | -0.0926 | 0.9 | -0.9139 | 0.7288 | False |
| | 20k Elements | 5k Elements | -0.6481 | 0.1898 | -1.4695 | 0.1732 | False |
| | 20k Elements | 80k Elements | 0.2778 | 0.8694 | -0.5436 | 1.0991 | False |
| | 40k Elements | 5k Elements | -0.5556 | 0.3332 | -1.3769 | 0.2658 | False |
| | 40k Elements | 80k Elements | 0.3704 | 0.6933 | -0.451 | 1.1917 | False |
| | 5k Elements | 80k Elements | 0.9259 | 0.0191 | 0.1046 | 1.7473 | True |
| Hypothesis 7 | CPS4 | CPS4R | -0.1296 | 0.9 | -0.9959 | 0.7366 | False |
| | CPS4 | CPS3 | -0.5556 | 0.4329 | -1.4218 | 0.3107 | False |
| | CPS4 | CPS8 | -0.1481 | 0.9 | -1.0144 | 0.7181 | False |
| | CPS4 | CPS8R | -0.1667 | 0.9 | -1.0329 | 0.6996 | False |
| | CPS4 | CPS6M | -0.3333 | 0.8618 | -1.1996 | 0.5329 | False |
| | CPS4R | CPS3 | -0.4259 | 0.685 | -1.2922 | 0.4403 | False |
| | CPS4R | CPS8 | -0.0185 | 0.9 | -0.8848 | 0.8477 | False |
| | CPS4R | CPS8R | -0.037 | 0.9 | -0.9033 | 0.8292 | False |
| | CPS4R | CPS6M | -0.2037 | 0.9 | -1.07 | 0.6626 | False |
| | CPS3 | CPS8 | 0.4074 | 0.7203 | -0.4589 | 1.2737 | False |
| | CPS3 | CPS8R | 0.3889 | 0.7557 | -0.4774 | 1.2552 | False |
| | CPS3 | CPS6M | 0.2222 | 0.9 | -0.644 | 1.0885 | False |
| | CPS8 | CPS8R | -0.0185 | 0.9 | -0.8848 | 0.8477 | False |
| | CPS8 | CPS6M | -0.1852 | 0.9 | -1.0514 | 0.6811 | False |
| | CPS8R | CPS6M | -0.1667 | 0.9 | -1.0329 | 0.6996 | False |

Table C.8: Results of the Tukey HSDs 2/2

| Test object | group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|---|
| Difference of methods | PMR | RAMP | -0.9402 | 0.001 | -1.2362 | -0.6443 | True |
| | PMR | SIMP | -0.0069 | 0.9 | -0.3028 | 0.2891 | False |
| | RAMP | SIMP | 0.9333 | 0.001 | 0.6137 | 1.253 | True |
| Methods (Half Michell) | PMR | RAMP | -1.4763 | 0.001 | -2.1112 | -0.8414 | True |
| | PMR | SIMP | -0.241 | 0.6237 | -0.8759 | 0.3939 | False |
| | RAMP | SIMP | 1.2353 | 0.001 | 0.5483 | 1.9223 | True |
| Methods (Case 1) | PMR | RAMP | -1.9763 | 0.001 | -2.5797 | -1.3729 | True |
| | PMR | SIMP | -0.1724 | 0.7521 | -0.7758 | 0.431 | False |
| | RAMP | SIMP | 1.8039 | 0.001 | 1.151 | 2.4568 | True |
| Methods (Case 2) | PMR | RAMP | -0.393 | 0.1917 | -0.9299 | 0.1439 | False |
| | PMR | SIMP | -0.4714 | 0.0961 | -1.0083 | 0.0655 | False |
| | RAMP | SIMP | -0.0784 | 0.9 | -0.6593 | 0.5025 | False |
| Methods (Case 3) | PMR | RAMP | -0.0678 | 0.9 | -0.616 | 0.4804 | False |
| | PMR | SIMP | 0.1479 | 0.7739 | -0.4003 | 0.6961 | False |
| | RAMP | SIMP | 0.2157 | 0.6455 | -0.3775 | 0.8088 | False |
| Methods (Case 4) | PMR | RAMP | -0.2132 | 0.5 | -0.6669 | 0.2404 | False |
| | PMR | SIMP | 0.4534 | 0.0502 | -0.0002 | 0.9071 | False |
| | RAMP | SIMP | 0.6667 | 0.0052 | 0.1758 | 1.1575 | True |
| Methods (Case MBB) | PMR | RAMP | -1.5686 | 0.001 | -2.2786 | -0.8586 | True |
| | PMR | SIMP | 0.3922 | 0.386 | -0.3179 | 1.1022 | False |
| | RAMP | SIMP | 1.9608 | 0.001 | 1.1925 | 2.729 | True |

Table C.9: Results of the Linear Regression

| Test object | $R^2$ | p | intercept | slope |
|---|---|---|---|---|
| Mesh density (PMR) | 0.990577 | 0.000000 | 13.267619 s | 0.000171 s/node |
| Mesh density (RAMP) | 0.956974 | 0.000000 | 2.009753 s | 0.000053 s/node |
| Mesh density (SIMP) | 0.991593 | 0.000000 | 1.913398 s | 0.000057 s/node |
| Same case (loop 1) | 0.986812 | 0.000000 | 21.127775 s | 0.023123 s/try |
| Same case (loop 2) | 0.891070 | 0.000000 | 20.966208 s | 0.028180 s/try |

# APPENDIX D

## Abaqus PMR Python Code

### D.1 Code

```
# ***************************************
# Author: Tjard Baetge
# Created on: 24.01.21
# Description: This Prescribed Material Redistribution script is a heuristic
    ↪ scheme for minimizing compliance under a
#              single volume constraint. The script needs to be run from within
    ↪ the Abaqus CAE. The input parameter
#              are the volume fraction and the number of iterations. The PMR
    ↪ method has been developed
#              by Taggart et al. in 2008.
#
#              Taggart, D. G., Dewhurst, P., Dobrot, L., & Gill, D. D.
#              "Development of a beta function based topology optimization
    ↪ procedure."
#              2008 Abaqus Users Conference, Newport, RI. 2008.
#
# ***************************************
import numpy as np
import time
import scipy.special as sc
from scipy import interpolate as sinterpolate
from abaqus import *
from abaqusConstants import *
from caeModules import *
import visualization
import pickle
import shutil
import os

def pmrgen(volfrac, n):
    tic = time.time()
    print('--------- PMR scheme started ---------')
    print('Volume fraction: %3.2f%%' % (volfrac * 100))
    print('Number of Iterations: %d' % (n))
    print('--------------------------------------')

    # settings
    time_smooth = 1     # (1 = on, 0 = off)
    use_check_sym = 1   # (1 = on, 0 = off)
    numCPUs = 7         # number of CPUs that should be used

    # get current working directory and file path
    cwd = os.getcwd()
    pathFolder = cwd + '/'
    fileName = ''
    for root, dirs, files in os.walk(cwd):
        for file in files:
            if file.find(".cae") != -1:     # only one .cae file per folder
                ↪ allowed!
                fileName = file
                break
    pathFile = pathFolder+fileName

    # read out information and manipulate the .inp file (noe = number of elements,
        ↪  non = # of nodes)
    noe, non, ModelName, PartName, InstanceName, JobName, ele_nodes, StepName =
        ↪ get_first_info(pathFile)
    E, PR = manipulate_inp_file(pathFolder+JobName+'.inp')

    # initialize node and element arrays
    numdesign = int(non)
    numtot = numdesign  # this scheme is not using predefined void or dense nodes
    node_design = [i for i in range(1, numdesign+1)]
    element_design = [i for i in range(1, int(noe) + 1)]
    node_design_array = np.array(node_design) - 1
```

```
element_label_array = np.array(element_design) - 1

# initialize density parameters
xsave = np.zeros((numtot, n+1))
xsave[:, 0] = volfrac
x0 = volfrac * np.ones((numdesign, 1))
x1 = x0
x2 = x0
x4 = np.zeros((numtot, 1))
Usave = np.zeros((numtot, n))
x4[node_design_array] = x0
xvec = x4

# compute first cumulative distribution function
t = np.linspace(start=1./n, stop=.999, num=n)
Ut = np.zeros((n, 1))
cdf = np.linspace(0, 1, numdesign)
rhor = (1 - volfrac) / volfrac
xp = np.linspace(0, 1, 100)

# main loop
for i in range(0, n):
    iterstart = time.time()

    # beta function calculation
    lambda_i = t[i] * (volfrac - volfrac**2)**.5
    r = (-1 + rhor / (lambda_i ** 2 * (1 + rhor) ** 2)) / (1 + rhor)
    s = r * rhor
    yp = sc.betainc(r, s, xp)
    f = sinterpolate.interp1d(np.append(np.insert(yp[(yp > 0) & (yp < 1)], 1,
        ↪ 0), 1),
                              np.append(np.insert(xp[(yp > 0) & (yp < 1)], 1,
                                  ↪ 0), 1))
    rho = f(cdf)    # density distribution for this iteration

    # FE-Analysis
    print('FEA start')
    call_FE(noe, ele_nodes, E, PR, xvec, JobName, InstanceName, pathFolder,
        ↪ element_label_array, numCPUs)
    print("FEA finished")
    # read in strain energy values
    U, Ut[i] = read_values(non, StepName)

    Usave[:, i] = U[:, 0]
    # update x
    Udesign = U[node_design_array]
    rank = np.argsort(Udesign[:, 0])    # mask for sorted U values
    x2[rank, 0] = rho   # assign densities to nodes based on rank

    if time_smooth == 0:
        # no time step smoothing:
        if use_check_sym == 1:
            x2 = check_sym(Udesign, x2)

        x2[x2 < .01] = .01
        x2[x2 > 1] = 1
        x3 = np.zeros((numtot, 1))
        x3[node_design_array] = x2

        xsave[:, i+1] = x3[:, 0]
        xvec = x3
    else:
        # time step smoothing
        if i < 2:
            x3 = x2
            x1 = x3
        else:
            x3 = x1 + .5 * (x2 - x0)
            x0 = x1
            x1 = x3

        if use_check_sym == 1:
            x3 = check_sym(Udesign, x3)

        x3[x3 < .01] = .01
        x3[x3 > 1] = 1
        x4[node_design_array] = x3

        xvec = x4
```

```
            xsave[:, i+1] = x4[:, 0]

        # print results of this iteration
        print('Iteration, Time, Total strain energy = %d/%d, %f, %e' %(i+1, n, t[i
            ↪ ], Ut[i]))

        if i == 0:  # copy file in first iteration
            src = 'pmr_job.odb'
            dst = 'vis_pmr_job.odb'
            shutil.copyfile(src, dst)

        # visualize results in Abaqus
        visualize_densities(xvec, i, node_design, Ut, n, StepName, t)

        iterstop = time.time()
        print('Total iteration duration: %f'%(iterstop-iterstart))
        rest = (iterstop-iterstart)/60*(n-(i+1))
        seconds = rest%1.0*60
        print('Estimated time left: %d min %d sec'%(rest, seconds))

    toc = time.time()
    total_calc_duration = toc-tic
    print("Elapsed time is:  {} min".format((total_calc_duration)/60))

    # save results for plotting in a dictionary and dump it
    information = {}
    information['Volume Fraction'] = volfrac
    information['Number of Iterations'] = n
    information['Number of Elements'] = noe
    information['Number of Nodes'] = non
    information['Total Strain Energy'] = Ut
    information['Duration'] = total_calc_duration
    information['Densities'] = xsave
    pickle.dump(information, open("pmr_info.p", "wb"))

    # View Cut options
    session.viewports['Viewport: 1'].odbDisplay.ViewCut(name='Cut-4',
                                                        shape=ISOSURFACE)
    session.viewports['Viewport: 1'].odbDisplay.viewCuts['Cut-4'].setValues(
        showModelBelowCut=False)
    session.viewports['Viewport: 1'].odbDisplay.viewCuts['Cut-4'].setValues(
        showModelOnCut=False)
    session.viewports['Viewport: 1'].odbDisplay.viewCuts['Cut-4'].setValues(
        showModelAboveCut=True)
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        UNDEFORMED,))
    session.viewports['Viewport: 1'].view.fitView()


def check_sym(U, x):
    """
    This function assigns nodes with the same strain energy density the same
        ↪ density
    :param U: strain energy density vector
    :param x: density vector
    :return:  revised density vector
    """
    check_start = time.time()

    # calculate unique strain energy density values
    values, index, count = np.unique(U, return_counts=True, return_index=True)
    index2 = index[count > 1]   # indices of nodes with strain energy values that
        ↪ occur more than once

    for i in index2:
        y = np.argwhere(U[:, 0] == U[i])    # create mask of nodes with the same
            ↪ strain energy value
        x[y] = float(np.mean(x[y]))     # average densities to conserve mass

    check_stop = time.time()
    print('Check sym duration: %f'%(check_stop-check_start))
    return x

def get_first_info(pathFile):
    """
    This function reads the model database and initially extracts the important
        ↪ information
    :param pathFile: path to the .cae file
```

```
    :return: Number of Elements, Number of Nodes, Dictionary which element has
        ↪ which nodes,
             and Names of Model, Part, Instance, Job and Step
    """
    openMdb(pathName=pathFile)  # open file and read info
    ModelName = mdb.models.keys()[0]
    PartName = mdb.models[ModelName].parts.keys()[0]
    InstanceName = mdb.models[ModelName].rootAssembly.instances.keys()[0]
    JobName = mdb.jobs.keys()[0]
    StepName = mdb.models[ModelName].steps.keys()[1]

    # ensure that strain energy measurement is selected
    FieldOutputName = mdb.models[ModelName].fieldOutputRequests.keys()[0]
    mdb.models[ModelName].fieldOutputRequests[FieldOutputName].setValues(variables
        ↪ =(
        'S', 'PE', 'PEEQ', 'PEMAG', 'LE', 'U', 'RF', 'CF', 'CSTRESS', 'CDISP',
        'ENER'))

    # write the .inp file that can be manipulated later on
    mdb.jobs[JobName].writeInput(consistencyChecking=OFF)

    # read number of nodes and number of elements
    p = mdb.models[ModelName].parts[PartName]
    stats = p.getMeshStats()
    non = stats.numNodes
    noe_point = stats.numPointElems
    noe_line = stats.numLineElems
    noe_tri = stats.numTriElems
    noe_hex = stats.numHexElems
    noe_wedge = stats.numWedgeElems
    noe_tet = stats.numTetElems
    noe_pyramid = stats.numPyramidElems
    noe_quad = stats.numQuadElems
    noe = noe_quad+noe_tet+noe_tri+noe_hex+noe_pyramid+noe_point+noe_wedge+
        ↪ noe_line

    ele_nodes = {}  # dictionary which element hold which nodes
    for element in p.elements:
        nodes = element.getNodes()
        ele_nodes[element.label] = [node.label for node in nodes]

    return noe, non, ModelName, PartName, InstanceName, JobName, ele_nodes,
        ↪ StepName

def call_FE(noe, ele_node, E, PR, xvec, JobName, InstanceName, pathFolder,
    ↪ element_label_array, numCPUs):
    """
    This function solves the FE problem by running an Abaqus job
    :param noe: Number of Elements
    :param ele_node: Dictionary which elements holds which nodes
    :param E: Young's Modulus
    :param PR: Poisson Ratio
    :param xvec: density distribution vector
    :param JobName: name of job
    :param InstanceName: name of instance
    :param pathFolder: path of the file's folder
    :param element_label_array: array with element labels
    """
    # calculate stiffness based on nodal densities
    element_stiffness = calculate_element_stiffness(noe, ele_node, E, PR, xvec)

    # write the distribution table in the new input file
    write_inp_file(E, PR, element_stiffness, InstanceName, pathFolder,
        ↪ element_label_array)

    # run job
    start = time.time()
    myjob = mdb.JobFromInputFile('pmr_job', JobName+'.inp', numCpus=numCPUs,
        ↪ numDomains=numCPUs)
    myjob.submit(consistencyChecking=OFF)
    myjob.waitForCompletion()
    stop = time.time()
    print('job duration: %f'%(stop-start))

def calculate_element_stiffness(noe, ele_node, E, PR, xvec):
    """
    This function linearly calculates elemental stiffness based on average nodal
        ↪ densities
```

```
    :param noe: Number of Elements
    :param ele_node: Dictionary which elements holds which nodes
    :param E: Young's Modulus
    :param PR: Poisson Ratio
    :param xvec: density distribution vector
    :return: elemental stiffness (and Poisson Ratio)
    """
    element_stiffness = 0.01*np.ones((noe, 2))  # pre-allocate
    element_stiffness[:, 1] = PR
    avg_element_density = [np.mean(xvec[np.array(node_ids) - 1]) for element_id,
        ↪ node_ids in ele_node.items()]
    element_stiffness[:, 0] = E*np.array(avg_element_density)  # linear material
        ↪ law
    return element_stiffness

def write_inp_file(E, PR, data, instance_name, pathFolder, element_label_array):
    """
    This function writes the elemental stiffness into the Abaqus distribution
        ↪ table, which is located in an additional
    .inp file. Only this .inp file is manipulated in this function.
    :param E: Young's Modulus
    :param PR: Poisson Ratio
    :param data: array with elemental stiffness and PR
    :param instance_name: name of the instance
    :param pathFolder: path of the file's folder
    :param element_label_array: array with element labels
    """
    filename = pathFolder+'Distribution1.inp'
    try:    # if file does not exist, it is created
        f = open(filename, "x")
    except:
        f = open(filename, "w")

    f.write(', %f, %f\n'%(E, PR))
    for i in element_label_array:   # stiffness values for each design element
        f.write('%s.%d, %f, %f\n'%(instance_name, i+1, data[i, 0], data[i, 1]))
    f.close()

def read_values(non, StepName):
    """
    This function reads the strain energy from the .odb file the job execution
        ↪ created.
    :param non: number of nodes
    :param StepName: name of the step
    :return: vector with strain energy densities and the total strain energy of
        ↪ the system
    """
    read_value_start = time.time()
    # open database
    myOdb = visualization.openOdb(path='pmr_job.odb')
    # get first step
    firstStep = myOdb.steps[StepName]
    # get last frame
    frame1 = firstStep.frames[-1]
    # get strain energy and strain energy density values
    strainenergy = frame1.fieldOutputs['SENER']
    region = firstStep.historyRegions['Assembly ASSEMBLY']
    total_strain_energy = region.historyOutputs['ALLIE'].data[-1][1]
    # transform outputs to element nodal values
    myfield = strainenergy.getSubset(position=ELEMENT_NODAL)
    myfieldValues = myfield.values
    # average the values for the same node for the different elements the node
        ↪ belongs to and save it to numpy array
    strain_energy_values = np.zeros((non, 1))
    counter = np.zeros((non, 1))
    for v in myfieldValues:
        strain_energy_values[v.nodeLabel - 1] += v.data
        counter[v.nodeLabel - 1] += 1
    strain_energy_values = strain_energy_values / counter

    myOdb.close()
    read_value_stop = time.time()
    print('read value duration: %f'%(read_value_stop-read_value_start))
    return strain_energy_values, total_strain_energy
```

```
def manipulate_inp_file(filename):
    """
    This function manipulates the initial .inp file to include the distribution
        ↪ table from an auxiliary file
    :param filename: name of the file
    :return: Young's modulus and Poisson ratio of the fully dense material as
        ↪ specified in the FE model
    """
    # 1) read
    with open(filename, "r") as file:
        data = file.read()

    # 2) manipulate text
    # find the important passage in the .inp file and read out E and PR
    text = data[data.find("** MATERIALS"):data.find("** -----")]
    textlist = text.split("\n")
    materialname = textlist[2][textlist[2].find("=") + 1:]
    properties = [float(item) for item in textlist[-2].split(",")]
    E0 = properties[0]
    v0 = properties[1]
    # rewrite the material definition to include the distribution table
    newtext = '** MATERIALS\n**\n*DISTRIBUTION TABLE, name=Table1\n Modulus, Ratio
        ↪ \n' \
              '*DISTRIBUTION, name=Distro1, Location=element, Table=Table1\n*
                  ↪ INCLUDE, INPUT=Distribution1.inp\n' \
              '*Material, name=%s\n*Elastic\n Distro1,\n'%(materialname)
    newdata = data.replace(text, newtext)

    # 3) write again
    with open(filename, "w") as file:
        file.write(newdata)

    return E0, v0

def visualize_densities(xvec, i, node_design, Ut, n, StepName, t):
    """
    This function writes density distribution and objective value into the .odb
        ↪ file and displays
     the relative density field in the viewport
    :param xvec: density distribution vector
    :param i: iteration number
    :param node_design: list of design nodes
    :param Ut: total strain energy vector
    :param n: number of iterations
    :param StepName: name of the step
    :param t: artificial time vector
    """
    # open
    o3 = visualization.openOdb(path='vis_pmr_job.odb')
    InstanceName = o3.rootAssembly.instances.keys()[0]
    myPart = o3.rootAssembly.instances[InstanceName]
    firstStep = o3.steps[StepName]
    # get last frame
    frame2 = firstStep.Frame(incrementNumber=int(i+2), frameValue=float(t[i]))
    if i == n-1:    # in the last iteration the strain energy history is added
        region = firstStep.historyRegions['Assembly ASSEMBLY']
        history = region.HistoryOutput(name='TSE', description='Total Strain
            ↪ Energy', type=SCALAR)
        values = Ut[:i+1, 0]
        history.addData(frameValue=t, value=values)

    # generate Relative Density Field Variable
    tField = frame2.FieldOutput(name='Relative Density',
                                description='Densities', type=SCALAR)
    # Create the node labels.
    nodeLabelData = node_design
    # Each set of density data corresponds to a node label.
    dispData = xvec
    # Add nodal data to the FieldOutput object using the
    # node labels and the nodal data for this part instance.
    tField.addData(position=NODAL, instance=myPart, labels=nodeLabelData, data=
        ↪ dispData)
    # make sure everything is saved and displayed properly
    firstStep.setDefaultField(tField)
    o3.update()
```

```
    o3.save()
    o3.close()
    o3 = visualization.openOdb(path='vis_pmr_job.odb')
    # set viewport to this new fiels
    session.viewports['Viewport: 1'].setValues(displayedObject=o3)
    session.viewports['Viewport: 1'].makeCurrent()
    session.viewports['Viewport: 1'].view.fitView()
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        CONTOURS_ON_DEF, ))
    session.viewports['Viewport: 1'].odbDisplay.setFrame(frame=frame2)
    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='Relative Density', outputPosition=NODAL)

if __name__ == '__main__':
    # popup for volfrac and iterations input
    fields = (('Volume Fraction [0,1]:', '0.07'), ('Iterations:', '100'))
    volfrac, num_iteration = getInputs(fields=fields, label='Specify PMR settings:
        ↪ ',
            dialogTitle='PMR Topology Optimization', )
    volfrac = float(volfrac)
    num_iteration = int(num_iteration)

    # run PMR algorithm
    pmrgen(volfrac, num_iteration)

    # popup for filename and save results under that name
    fields = (('File name [.odb / .p]:', 'myResult'))
    filename = getInputs(fields=fields, label='Save OutputFile as:', dialogTitle='
        ↪ Calculations Finished!', )
    filename = str(filename[0])
    src = 'vis_pmr_job.odb'
    dst = filename+'.odb'
    scr_info = 'pmr_info.p'
    dst_info = filename+'.p'
    shutil.copyfile(src, dst)
    shutil.copyfile(scr_info, dst_info)
```

## D.2 User Guide (Cantilever Example)

1. Create a new folder

2. Copy the file Abaqus_PMR_TO_Scheme.py into that folder

3. Open Abaqus CAE

4. Click File, click Save, and save the *model_filename*.cae file in the new folder

5. Close Abaqus CAE

6. Double click the *model_filename*.cae file in your file browser to open it Abaqus CAE again. Now your working directory is set to your folder, and all temporary files will be saved in that folder.

7. Create the Abaqus Model

a. Part Module: Click Part, Create, choose 2D Planar, click Continue. Choose "Create line rectangle" (second row, second column). Enter (0,0), hit enter, enter (100,100), hit enter. Cancel Procedure, click Done.

b. Property Module: Create Material, mechanical → elasticity → elastic (E=210e9 GPa, PR=0.3). Click OK. Create Section, Continue, OK. Assign section, click on part, Done, OK.

c. Assembly Module: Create Instance, OK.

d. Step Module: Create Step, Continue, OK.

e. Load Module: Create Load, Continue, select bottom right corner of your part, click Done, CF1=0, CF2=-1, OK. Create Boundary Condition, select Displacement/Rotation, continue, select left edge of your part, click Done, and check U1 and U2, OK.

f. Mesh Module: Select Part instead of Assembly under Object in the top bar above the viewport, Seed Part Instance, Approximate Global Size=1, OK. Click Mesh Part, Yes.

g. Job Module: Create Job, Continue, OK.

8. Click File, click Save.

9. Close Abaqus CAE.

10. Double click the *model_filename*.cae file in your file browser to open it Abaqus CAE again. Closing and opening makes sure all changes are saved.

11. Click File, click Run Script, select Abaqus_PMR_TO_Scheme.py, click OK.

12. Specify Volume Fraction (example case=0.4)

13. Specify Number of Iterations (example case=75)

14. Click OK.

15. Once finished: Specify *save_filename* to save results and click OK. Now the *save_filename*.odb file and the *save_filename*.p file are saved to your folder containing all the information of the scheme.

16. To visualizes the result more comprehensively: close Abaqus, you do not need to save the changes, and then double click on the new *save_filename*.odb file.

17. Visualization options

    a. Click "plot contours on deformed shape" to see the final density distribution. This variable can also be animated over the time history.

    b. Click Viewcut manager, Create, select shape → isosurface, OK. Check the second and the third box of the new cut, uncheck the first box.

    c. Alternative to the view cut: Click Tools, Display Group, create, select result value, specify min=0.5 and max=1, click Replace, save or close the display group window.

    d. Strain Energy History: Click Result, History Output, Select Total Strain Energy (TSE), Plot

Notes (for advanced users):

- The names of the part, instance, job, etc. are irrelevant.
- The step must be general static.
- The material must be linear elastic.
- There should only be one part/one instance in the assembly.
- There should only be one *.cae file in the folder.

- The script changes the field output variables. If other variables than default are wanted, the line must be commented out of the script, and ENER must be specified in the model database.
- The script specifies the number of CPUs used in the beginning (default=7). This value can be changed for the individual PCs and might lead to an error if the number of processing cores does not match your system.
- Time smoothing and symmetry checking can be turned off in the beginning of the script by setting the values to 0.
- If the mesh density is changed, the assembly needs to be regenerated before saving and running the PMR script.

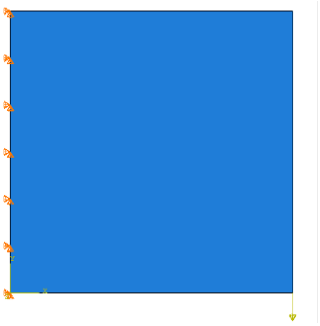

Figure D.64: Cantilever Example Loading Conditions

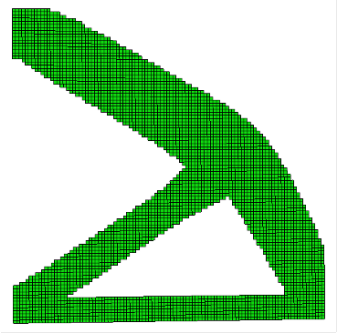

Figure D.65: Cantilever Example Solution

# BIBLIOGRAPHY


Andreasen, C. S., Elingaard, M. O., and Aage, N., "Level set topology and shape optimization by density methods using cut elements with length scale control," *Structural and Multidisciplinary Optimization*, pp. 1–23, 2020.

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., and Sigmund, O., "Efficient topology optimization in matlab using 88 lines of code," *Structural and Multidisciplinary Optimization*, vol. 43, no. 1, pp. 1–16, 2011.

Aremu, A., Ashcroft, I., Hague, R., Wildman, R., and Tuck, C., "Suitability of simp and beso topology optimization algorithms for additive manufacture," in *21st annual international solid freeform fabrication symposium (SFF)–An additive manufacturing conference*, 2010, pp. 679–692.

Bathe, K.-J., *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

Bendsøe, M. P., "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, no. 4, pp. 193–202, 1989.

Bendsøe, M. P. and Sigmund, O., "Material interpolation schemes in topology optimization," *Archive of applied mechanics*, vol. 69, no. 9, pp. 635–654, 1999.

Brackett, D., Ashcroft, I., and Hague, R., "Topology optimization for additive manufacturing," in *Proceedings of the solid freeform fabrication symposium, Austin, TX*, vol. 1, 2011, pp. 348–362.

Bruns, T., "A reevaluation of the simp method with filtering and an alternative formulation for solid–void topology optimization," *Structural and Multidisciplinary Optimization*, vol. 30, no. 6, pp. 428–436, 2005.

Bulman, S., Sienz, J., and Hinton, E., "Comparisons between algorithms for structural topology optimization using a series of benchmark studies," *Computers & structures*, vol. 79, no. 12, pp. 1203–1218, 2001.

Cai, X., Langtangen, H. P., and Moe, H., "On the performance of the python programming language for serial and parallel scientific computations," *Scientific Programming*, vol. 13, no. 1, pp. 31–56, 2005.

Challis, V. J., "A discrete level-set topology optimization code written in matlab," *Structural and multidisciplinary optimization*, vol. 41, no. 3, pp. 453–464, 2010.

Chandrasekhar, A. and Suresh, K., "Tounn: Topology optimization using neural networks," *Structural and Multidisciplinary Optimization*, vol. 63, no. 3, pp. 1135–1149, 2021.

Dean, A., Voss, D., Draguljić, D., *et al.*, *Design and analysis of experiments*, 2nd ed. Springer, 2007.

Deaton, J. D. and Grandhi, R. V., "A survey of structural and multidisciplinary continuum topology optimization: post 2000," *Structural and Multidisciplinary Optimization*, vol. 49, no. 1, pp. 1–38, 2014.

Diaz, A. and Sigmund, O., "Checkerboard patterns in layout optimization," *Structural optimization*, vol. 10, no. 1, pp. 40–45, 1995.

Guest, J. K., Prévost, J. H., and Belytschko, T., "Achieving minimum length scale in topology optimization using nodal design variables and projection functions," *International journal for numerical methods in engineering*, vol. 61, no. 2, pp. 238–254, 2004.

Guo, X. and Cheng, G.-D., "Recent development in structural design and optimization," *Acta Mechanica Sinica*, vol. 26, no. 6, pp. 807–823, 2010.

Haslwanter, T., *An Introduction to Statistics with Python*. Springer, 2016.

Hughes, T. J., *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

Lewiński, T. and Rozvany, G., "Exact analytical solutions for some popular benchmark problems in topology optimization ii: three-sided polygonal supports," *Structural and Multidisciplinary Optimization*, vol. 33, no. 4, pp. 337–349, 2007.

Lewiński, T. and Rozvany, G., "Exact analytical solutions for some popular benchmark problems in topology optimization iii: L-shaped domains," *Structural and Multidisciplinary Optimization*, vol. 35, no. 2, pp. 165–174, 2008.

Lewiński, T., Rozvany, G., Sokół, T., and Bołbotowski, K., "Exact analytical solutions for some popular benchmark problems in topology optimization iii: L-shaped domains revisited," *Structural and Multidisciplinary Optimization*, vol. 47, no. 6, pp. 937–942, 2013.

Logan, D. L., *A first course in the finite element method*. Cengage Learning, 2016.

Michell, A. G. M., "Lviii. the limits of economy of material in frame-structures," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 47, pp. 589–597, 1904.

Osher, S. and Sethian, J. A., "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.

Pichugin, A. V., Tyas, A., and Gilbert, M., "Michell structure for a uniform load over multiple spans," in *9th World Congress on Structural & Multidisciplinary Optimization*. Sheffield, 2011.

Querin, O. M., Steven, G. P., and Xie, Y. M., "Evolutionary structural optimisation (eso) using a bidirectional algorithm," *Engineering computations*, 1998.

Reddy K, S. N., Ferguson, I., Frecker, M., Simpson, T. W., and Dickman, C. J., "Topology optimization software for additive manufacturing: A review of current capabilities and a real-world example," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2016.

Rojas-Labanda, S. and Stolpe, M., "Benchmarking optimization solvers for structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 52, no. 3, pp. 527–547, 2015.

Rozvany, G. I., "A critical review of established methods of structural topology optimization," *Structural and multidisciplinary optimization*, vol. 37, no. 3, pp. 217–237, 2009.

Rozvany, G. I., Zhou, M., and Birker, T., "Generalized shape optimization without homogenization," *Structural optimization*, vol. 4, no. 3-4, pp. 250–252, 1992.

Rozvany, G., "Exact analytical solutions for some popular benchmark problems in topology optimization," *Structural optimization*, vol. 15, no. 1, pp. 42–48, 1998.

The SciPy community. "Numpy documentation." Accessed on 07/08/2021. 2021. [Online]. Available: https://numpy.org/doc/stable/

SciPy developers. "Scipy documentation." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.scipy.org/docs.html

Shukla, A., Misra, A., and Kumar, S., "Checkerboard problem in finite element based topology optimization," *International Journal of Advances in Engineering & Technology*, vol. 6, no. 4, p. 1769, 2013.

Sigmund, O., "A 99 line topology optimization code written in matlab," *Structural and multidisciplinary optimization*, vol. 21, no. 2, pp. 120–127, 2001.

Sigmund, O. and Maute, K., "Topology optimization approaches," *Structural and Multidisciplinary Optimization*, vol. 48, no. 6, pp. 1031–1055, 2013.

SIMULIA by Dassault Systems. "Abaqus documentation." Accessed on 07/08/2021. 2016. [Online]. Available: http://130.149.89.49:2080/v2016/index.html

SIMULIA by Dassault Systems. "Abaqus documentation - the optimization task." Accessed on 07/08/2021. 2016. [Online]. Available: https://abaqus-docs.mit.edu/2017/English/TsoUserMap/tso-c-usr-sizing-optTask.htm

SIMULIA by Dassault Systems. "Abaqus CAE." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.3ds.com/products-services/simulia/products/abaqus/abaquscae/

SIMULIA by Dassault Systems. "Abaqus scripting user's guide." Accessed on 07/08/2021. 2021. [Online]. Available: http://130.149.89.49:2080/v6.14/books/cmd/default.htm

Stolpe, M. and Svanberg, K., "An alternative interpolation scheme for minimum compliance topology optimization," *Structural and Multidisciplinary Optimization*, vol. 22, no. 2, pp. 116–124, 2001.

Svanberg, K., "The method of moving asymptotes—a new method for structural optimization," *International journal for numerical methods in engineering*, vol. 24, no. 2, pp. 359–373, 1987.

Taggart, D. G., Dewhurst, P., Dobrot, L., and Gill, D. D., "Development of a beta function based topology optimization procedure," in *2008 Abaqus Users Conference, Newport, RI*, 2008.

Taggart, D. G., Dewhurst, P., and Nair, A. U., "Systems and methods for finite element based topology optimization," Dec. 18 2012, US Patent 8,335,668.

Taggart, D. and Dewhurst, P., "Development and validation of a numerical topology optimization scheme for two and three dimensional structures," *Advances in Engineering Software*, vol. 41, no. 7-8, pp. 910–915, 2010.

TopOpt. "Apps/software - topopt." Accessed on 07/08/2021. 2021. [Online]. Available: https://www.topopt.mek.dtu.dk/apps-and-software

Valdez, S. I., Botello, S., Ochoa, M. A., Marroquín, J. L., and Cardoso, V., "Topology optimization benchmarks in 2d: Results for minimum compliance and minimum volume in planar stress problems," *Archives of Computational Methods in Engineering*, vol. 24, no. 4, pp. 803–839, 2017.

Wang, M. Y., Wang, X., and Guo, D., "A level set method for structural topology optimization," *Computer methods in applied mechanics and engineering*, vol. 192, no. 1-2, pp. 227–246, 2003.

Xie, Y. M. and Steven, G. P., "A simple evolutionary procedure for structural optimization," *Computers & structures*, vol. 49, no. 5, pp. 885–896, 1993.

Xu, Y., Zhu, J., Wu, Z., Cao, Y., Zhao, Y., and Zhang, W., "A review on the design of laminated composite structures: constant and variable stiffness design and topology optimization," *Advanced Composites and Hybrid Materials*, vol. 1, no. 3, pp. 460–477, 2018.

Zegard, T. and Paulino, G. H., "Bridging topology optimization and additive manufacturing," *Structural and Multidisciplinary Optimization*, vol. 53, no. 1, pp. 175–192, 2016.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z., *The finite element method: its basis and fundamentals*. Elsevier, 2005.