Efficient Methods for High Fidelity Topology Optimization

Träff, Erik Albert

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

DTU Construct
Department of Civil and Mechanical Engineering

Efficient Methods for High Fidelity Topology Optimization

Erik Albert Träff

PhD Thesis

Efficient Methods for High Fidelity Topology Optimization

Erik A. Träff

Efficient Methods for High Fidelity Topology Optimization
March 2023

Ph.D. Project Title:
Efficient Solutions for Large-Scale Topology Optimization Problems

Author:
Erik Albert Träff

Main supervisor:
Professor Ole Sigmund
Technical University of Denmark

Co-supervisor:
Associate Professor Niels Aage
Technical University of Denmark

Funding and interests:
This work was funded by the Villum Foundation as part of the InnoTop Villum Investigator project. The author declares that they have no competing interests.

Technical University of Denmark
Department of Civil and Mechanical Engineering
Koppels Allé, Building 404
2800 Kgs. Lyngby, Denmark
Denmark
Phone: +45 4525 4250 Fax: +45 4525 1961
URL: www.construct.dtu.dk

ISSN 0903-1685 ISBN 978-87-7475-724-5

Preface

This thesis is submitted in partial fulfillment of the requirements for obtaining the degree of PhD in mechanical engineering at the Technical University of Denmark (DTU). The PhD project has been funded by Villum Fonden through the Villum Investigator project InnoTop. The work has been carried out at the Department of Civil and Mechanical Engineering, Section of Solid Mechanics during the period from 1st of February 2020 to the 2nd of April 2023, with exception of a leave of absence in the period 1st of September 2022 to 31 October 2022. The main supervisor has been Professor Ole Sigmund and the co-supervisor has been Associate Professor Niels Aage.

It is said to take a village to raise a child. It also takes a lot of people to raise a young researcher. The brunt of the burden in making me able to write this dissertation has been carried by my supervisors Niels Aage and Ole Sigmund, ever encouraging me to make up my own mind.

I spent three months in Delft with the guidance of Matthijs Langelaar and Fred van Keulen and the companionship of many bright students of mechanics. Additional thanks to Stijn, Arnoud, Breno, and Dirk for our many escapades. In the Netherlands I learnt the absolute importance of "putting my feet on the desk". To put it differently; direction matters far more than speed in research.

It is difficult to overstate the importance my co-workers in the FAM section, which have formed my day-to-day life, ever ready to discuss any topic, technical or otherwise. An additional thanks to Andreas, Rebekka, Christoffer, Peter, and Lukas, with whom I had the pleasure of sharing an office. I hope you enjoyed the experience as well, although your participation in discussions became less than voluntary at times.

Other people have stood out. Federico taught me many details of nonlinear analysis. Peter made effort to deliver dehomogenized structures in time to include preliminary results in this work. Lukas became a partner on several unexpected turns for our studies, which arose from putting our feet on the table during a Friday bar. He also generously proofread parts of this dissertation.

To all of the people above I remain grateful for raising me as a researcher. I am just as grateful to, and for, all my friends and family for their company during the three years.

Finally, a special thanks to Josefine.

Abstract

This thesis presents work related to efficient numerical solution schemes and implementations of large-scale, or high-resolution, topology optimization. The purpose of pursuing efficient methods is to increase the resolution of computed structures, which in turn enable new potential applications. The thesis has then been split into two, each part focusing on different types of computing hardware.

The first part treats the implementation of topology optimization on high-performance compute clusters. We therefore deal with distributed memory computing, primarily on distributed unstructured meshes. We begin by describing the multigrid preconditioner, which is a key component in solving the elasticity equations on the desired scale. Afterwards, a new filter for ensuring manufacturability by milling or casting is presented, specifically designed with distributed unstructured meshes in mind. Finally, a short case study of a direct drive wind turbine is performed.

The second part treats the implementation of structural optimization on a single desktop machine. We study GPU acceleration, and how far we can push the performance of a single high-end machine by limiting the problem to structured grids while in a shared memory setting. We then move on to shape optimization of shell structures, which emphasizes how high-resolution is not necessarily required to provide meaningful structural optimization results.

Finally, we summarize the results of this work with a critical reflection on the value of high-resolution topology optimization compared to the cost of required hardware and effort of implementation.

Resumé

Denne afhandling præsenterer arbejde udført indenfor effektive numeriske løsningsmetoder og implementeringer af højopløst topologioptimering. Formålet ved effektive metoder er at kunne øge opløsningsgraden af de beregnede strukturer, hvilket åbner for nye anvendelser. Afhandlingen er todelt, hver del er fokuseret på hver sin type beregningsmaskinel.

Første del omhandler udførelse af topologioptimering på højtydende beregningsklynger, såkaldte "supercomputere". Derfor benyttes programmering for delte hukommelsespladser, med henblik på ustrukturerede fordelte gitre. Først beskrives multigitterforkonditioneringsmetoder, som er nødvendige for at løse elasticitetsligningerne ved den ønskede opløsning. Herefter præsenteres et nyt filter, der sikrer at resulterende strukturer kan fremstilles ved fræsning. Dette filter er udviklet særligt til ustrukturerede fordelte gitre i høj opløsning. Til slut vises et kort studie af optimering af et svinghjul anvendt i en direkte drevet vindmølle.

Anden del omhandler udførsel af strukturoptimering på en enkelt computer. Vi studerer grænserne for hvor høj opløsning, der kan opnås på en enkelt avanceret computer ved brug af en grafikbehandlingsenhed samt udnyttelse af den orden i beregninger, der opstår ved brug af strukturerede gitre. Herefter undersøges formoptimering af skalstrukturer. I kontrast til den højopløste topologioptimering understreger dette studie, at høj opløsning ikke altid er nødvendig for at opnå tilfredstillende resultater i strukturoptimering.

Til slut opsummeres resultaterne med en kritisk overvejelse over værdien af højopløst topologioptimering sammenlignet med prisen for beregningsmaskinel og anstrengelsen ved implementering.

Included works

The following articles are considered part of this dissertation:

[P1] E. A. Träff, O. Sigmund, and N. Aage. *"Topology Optimization of Ultra High Resolution Shell Structures."* Thin-Walled Structures 160 (March 2021): 107349. doi:10.1016/j.tws.2020.107349. [Published]

[P2] L. C. Høghøj and E. A. Träff. *"An Advection–Diffusion Based Filter for Machinable Designs in Topology Optimization."* Computer Methods in Applied Mechanics and Engineering 391 (March 2022): 114488. doi:10.1016/j.cma.2021.114488. [Published]

[P3] A. C. Hayes, E. A. Träff, C. V. Sørensen, S. V. Willems, N. Aage, O. Sigmund and G. L. Whiting. *"Topology Optimization For Structural Mass Reduction of Direct Drive Electric Machines." Sustainable Energy Technologies and Assessments. [In review]*

[P4] E. A. Träff, A. Rydahl, S. Karlsson, O. Sigmund and N. Aage. *"Simple and Efficient GPU accelerated Topology Optimisation: codes and applications." Computer Methods in Applied Mechanics and Engineering. [Accepted]*

[P5] E. A. Träff, N. Aage, M. Langelaar, O. Sigmund and F. van Keulen. *"Optimisation of Shape and Thickness of Shell Structures." Thin-Walled Structures. [In preparation]*

The following article was published during the Ph.D. studies, but is not considered as part of this dissertation:

[P6] R. N. Glaesener, E. A. Träff, B. Telgen, R. M. Canonica and D. M. Kochmann. *"Continuum Representation of Nonlinear Three-Dimensional Periodic Truss Networks by on-the-Fly Homogenization."* International Journal of Solids and Structures 206 (December 2020): 101–13. doi:10.1016/j.ijsolstr.2020.08.013.

Contents

Chapter 1

Introduction

The only difference between screwing around and science is writing it down.

MythBusters
Adam Savage

1.1 Motivation and Goals

Topology optimization is a rapidly maturing technology. By specifying a design domain, objective, and constraints, users are able to generate optimized structures with little prior knowledge of the design. In essence, this works by decomposing the initial domain into elements, cells, or voxels and identifying whether each element should be considered part of the resulting structure.

Due to this representation of the structure, the amount of detail available to describe the resulting structure is directly tied to the resolution of the physics. To obtain a desired length-scale it is necessary to ensure that sufficiently many elements are used. If larger structures are considered for topology optimization, or if small length-scales are required, many elements may become necessary. It is not uncommon to have requirements ranging from 4 to 50 million elements, resulting in long computation times when solving the state equations.

Not only is the desired resolution an important aspect of the usability of topology optimization as a tool for designers. The differences between waiting a second, a minute, an hour, or a day for the result are difficult to overstate. Iteration in the design process is sped up significantly when results and feedback are provided quickly.

There is therefore a need for fast, large-scale topology optimization programs if the method is to become widely useful. The goal of this Ph.D. project is to develop efficient methods for topology optimization, in order to reduce computational time sufficiently to make topology optimization better suited for commercial applications. This is a goal which can be approached in a multitude of ways. This work has taken the form of a series of efficient implementations, which leverage a variety of formulations and hardware. By comparing the merits of these implementations, the goal is to gain insight on how to build fast large-scale topology optimization programs.

1.2 A Reader's Guide

As this work builds upon many existing concepts, it would be excessive to introduce all underlying theory thoroughly. It is expected that the reader has familiarity with linear algebra [1], standard solution techniques for linear systems [2], finite element analysis [3], and topology optimization [4, 5].

In this thesis several approaches to efficient large-scale topology optimization are presented. Initially, some background is given to multigrid methods in chapter 2. These method are central in all following topology optimization methods. The following two chapters deal with using unstructured meshes in topology optimization. Chapter 3 proposes a method to ensure designs which can be manufactured by machining, which trivially generalizes to unstructured and distributed finite element meshes. Chapter 4 performs a case study of a large-scale topology optimization problem performed on a distributed unstructured mesh. As a contrast to the unstructured distributed topology optimization problems, chapter 5 studies the performance of structured meshes solved on a single computer. Several implementations are presented, both with and without the use of GPU acceleration. Chapter 6 studies shape optimization of shell structures through a formulation reminiscent of topology optimization. Here great improvements in performance are attained when additional assumptions about the resulting structure are adopted to the problem formulation. Finally, some concluding remarks are given in chapter 7.

1.3 Topology Optimization

Although it is assumed that the reader is familiar with standard concepts within density based topology optimization, it is useful to briefly formulate the problem at hand. The well-studied linear elastic minimum compliance problem is presented here, as this problem is studied in all papers included in this dissertation. The optimization problem is stated as;

$$\underset{\mathbf{x}}{\text{minimize}} \qquad \mathcal{C} = \mathbf{u}^\top \mathbf{K}\mathbf{u} \tag{1.1a}$$

subject to

$$\text{state equation} \qquad \mathbf{K}\mathbf{u} = \mathbf{f} \tag{1.1b}$$

$$\text{filter} \qquad \tilde{\mathbf{x}} = \mathbf{F}\mathbf{x} \tag{1.1c}$$

$$\text{volume} \qquad \sum_i v_i \tilde{\mathbf{x}}_i \leq V^* \sum_i v_i \tag{1.1d}$$

$$\text{bounds} \qquad 0 \leq \mathbf{x}_i \leq 1 \; \forall i \tag{1.1e}$$

Here $\mathbf{x}$ denotes the vector of design variables which has a value for every element in the mesh, each variable is bounded, eq. (1.1e), here a value of 1 indicates solid material and 0 indicates void. The volume constraint, eq. (1.1d), is defined using the element volumes v_i and a volume fraction $0 < V^* < 1$, which indicates the fraction of the domain which can be solid material. The used filter operator F in eq. (1.1c) can be either of the convolution [6, 7] or PDE based type [8]. The stiffness matrix $\mathbf{K}$ is the operator associated with solving the elasticity equations discretized with FEM, it is implicitly dependent on the filtered densities $\tilde{\mathbf{x}}$, which couples the design and objective function through eqs. (1.1b) and (1.1c).

This problem can be solved in two ways. In the so-called "simultaneous analysis and design" (SAND) approach the displacement vector $\mathbf{u}$ is included as design variables, and all eq. (1.1) is treated as an optimization problem with equality and inequality constraints. The alternate approach, which is used throughout this work, is the so-called "nested analysis and design" (NAND). Here, the displacement vector $\mathbf{u}$ is considered as the result of a function of $\mathbf{x}$, defined through eq. (1.1c) first, and then by solving the linear system of eq. (1.1b). As both steps are differentiable the chain rule can be used to obtain the gradients $\frac{\partial \mathcal{C}}{\partial \mathbf{x}}$.

The SAND approach is, in a mathematical sense, elegant. It is also well-equipped to solve hard topology optimization problems, where small changes in $\mathbf{x}$ can lead to large changes in the objective, e.g. Bluhm et al. [9]. However, considering the approaches from an efficiency standpoint, the NAND approach is most attractive. Here, decades of research on efficiently solving the linear algebra systems arising from discretized PDEs, such as eq. (1.1b), can be used. By decomposing the problem into smaller nested problems, each step is solved efficiently with a specialized method, resulting in an efficient solution of the full problem.

1.4 Performance and Efficiency

Few people set out to create slow software. However, available software tends to vary widely in speed of computation. Many factors affect the speed of computation, or performance, of an implementation. Both internal factors of the software itself and external factors such as the used hardware affect the resulting speed.

Program efficiency is tied to both to the efficiency of the chosen algorithms,

and how the implementation itself is performed. These factors are to some degree codependent with the used hardware. Some algorithms lends themselves better to efficient implementations on some hardware, by e.g. being trivially parallelized. Similarly, implementations can be tuned for specific hardware, improving their efficiency on the target platforms, at the cost of performance when using other platforms and program simplicity.

Efficiency of implementations can be characterized by the amount of unnecessary resource usage. When considering compute efficiency, time which is spent performing unnecessary, possibly unintended, work decreases the efficiency of the implementation. This can e.g. be waiting for a cache-miss or the overhead of an interpreter performing just-in-time compilation. Avoiding as much computational work as possible, while providing correct results, makes a program more efficient. Similar efficiency considerations can be made for other metrics of interest, such as memory usage.

Topology optimization is a complex problem, consisting of many sub-problems some of which are themselves complex. Therefore, ample opportunity exists for both performance optimization and pessimization. Some of these opportunities include;

Linear solver Most topology optimization implementations spend a majority of the computing time solving systems of equations which arise from partial differential equations. The chosen method used to solve the resulting system of equations has a great effect on the overall topology implementation.

Design parametrization Some topology optimization approaches reduce computational complexity by removing void elements, using higher order elements, or other approaches which result in reduced system sizes for the linear equations.

Optimization problem formulation Some regularization methods for the topology optimization problem are known to result in slower convergence. For instance, the differentiable approximation to the heaviside projection [10] usually relies on continuation of a projection sharpness parameter. This drastically increases the required number of linear system solutions to reach convergence.

Optimization solver The method used to solve the optimization problem itself affects the convergence rate, which in turn affects the required number of linear system solutions.

All of these choices are strictly pertaining to the problem formulation itself, and affect the performance of the final program through the algorithmic efficiency of the chosen methods.

Benchmarking Performance

Comparing performance across implementations can more complex than apparent at first glance. The interdependence between hardware and implementation for the measured performance imply that results are difficult to generalize, as changing hardware might change the performance characteristics.

Measurements of computational time depend on many external factors, some which can be controlled for, and some which cannot. Hence, proper measurement of performance must realize that the measured value is nondeterministic. This implies that multiple measurements are necessary, and that proper statistical methods must be taken when analyzing the results. The interested reader may find an in depth discussion on how to treat performance in Hoefler and Belli [11].

In general, most topology optimization problems take long to execute, and have small variation in computation time compared to the total runtime. Therefore, for simple comparisons it is possible to treat these values deterministically.

Chapter 2

Geometric Multigrid Methods in Topology Optimization

[Truth] is much too complicated to allow anything but approximations.

The Mathematician
John von Neumann

Multigrid methods are commonly used in large-scale topology optimization methods [12–16]. These methods scale linearly in time when solving the linearized elasticity equations, if tuned properly. Since solving these equations tend to be the most computationally intensive aspect of performing topology optimization, multigrid methods are key for achieving good efficiency. This chapter summarizes multigrid methods which were not developed as part of the dissertation, but are included nevertheless for completeness. A more complete treatment of multigrid methods can be found in Briggs et al. [17], Trottenberg et al. [18].

2.1 Introduction to Structured Multigrid

In this section we study the specialization of multigrid methods for Cartesian grids, in order to develop many of the basic concepts related to multigrid methods. This specialization simplifies some concepts, and has been used for all implementations presented in **[P4]**.

Prolongation and Restriction

The central notion of geometric multigrid methods is to discretise the considered domain using a hierarchy of grids of varying spatial resolution. We wish to solve the equations on the finest mesh, while the coarser meshes are used to efficiently reduce the error. Figure 2.1 illustrates a hierarchy of two-dimensional Cartesian meshes.

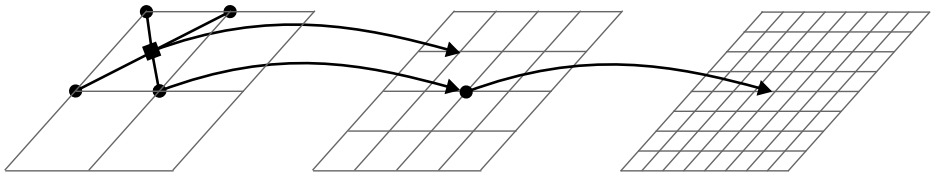


Figure 2.1: Three grids of varying spatial resolution representing the same two-dimensional box. Some prolongation operations are illustrated indicating which nodes of a coarser grid are used to interpolate the value at some node of the finer grid.

The prolongation operation maps a field from a coarse grid to a finer grid. The classic way of performing this mapping is to use the shape-functions of the coarse elements to evaluate field values at the nodes of the fine mesh. No other prolongations are considered in this dissertation.

Due to the Cartesian structure of the mesh, we can guarantee to find valid prolongations for all fine nodes of the mesh. It is even possible to compute the indices of nearby coarse or fine nodes solely by permuting the index of a node. This allows to write implementations where indices are computed on-the-fly, and underlying grids are defined implicitly.

Since fine nodal values are a linear combination of nearby coarse values, the resulting operator is linear, and can be represented as a sparse matrix $\mathbf{P}_i$. Resulting in the the mapping from grid i to $i+1$ as $\mathbf{u}_i = \mathbf{P}_{i+1}\mathbf{u}_{i+1}$. We note that $i = 0$ is used to denote the finest mesh, increasing the index for coarser meshes.

Restriction is the operation of mapping from a fine grid to a coarse grid. The usual choice for restriction is the matrix $\mathbf{R}_i = \mathbf{P}_i^\top$. Note however, that this will also scale values in the resulting coarse grid, as columns in the prolongation $\mathbf{P}_i$ do not represent a partition of unity, i.e. they do not sum to 1. While this attribute might seem problematic, as the field is scaled when transforming in one direction, it works well in practice.

An alternate choice of restriction would be to re-scale the rows of $\mathbf{P}_i^\top$, such that they become a partition of unity. This would enable a preservation of scaling when restricting the displacement field. This method can be used, although the choice of $\mathbf{P}_i^\top$ has some desirable properties which will be discussed in the next section.

Coarse Approximation of Elasticity Operator

Not only displacement fields need be defined for the coarse grids, also the corresponding operators, or stiffness matrices are necessary. These are important, as they form the foundation of reducing the solution error at every grid refinement.

The most common approach for generating coarse grid corrections is the Galerkin projection. Here, the operator for the finest level $\mathbf{K}_0$ is used to generate all other matrices by prolongation and restriction.

$$\mathbf{K}_i = \mathbf{R}_i \mathbf{K}_{i-1} \mathbf{P}_i \tag{2.1}$$

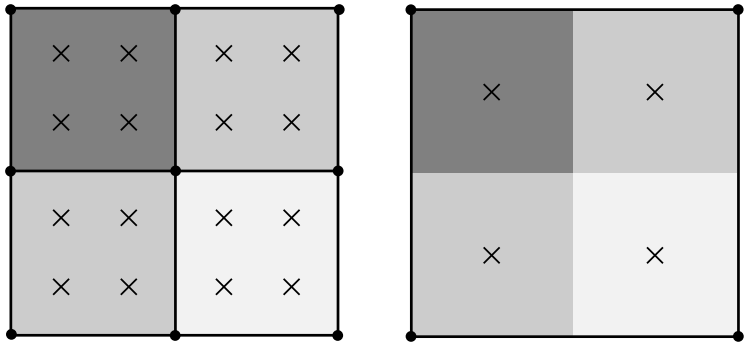


Figure 2.2: Illustration of four material cells, discretized by four quadrilaterals on the left, and one quadrilateral on the right. The crosses indicate integration points. The four quadrilaterals have constant density, and thus use a Gauss quadrature for exact numerical integration. The single quadrilateral integrates using a Riemann sum with one quadrature point for each density.

Here we can note that choosing $\mathbf{R}_i = \mathbf{P}_i^\top$ ensures that all coarse level operators of the Galerkin projection are symmetric by construction.

An alternate approach to the Galerkin projection, is to build the coarse space operators directly from the discretization of the studied elasticity equations. If a homogeneous medium is considered, this is relatively straightforward. The finite element assembly is simply performed as usual on all coarse grids. When the material parameters are spatially varying, as is the case for topology optimization, care must be taken when choosing how to build the coarse space element matrices.

Inspired by the approach used in Nguyen et al. [19], it is possible to perform integration of local elements with spatially varying material parameters within these elements. By introducing sub-cells of an element, as illustrated in fig. 2.2, it is possible to construct an element with spatially varying stiffness corresponding to a finer underlying mesh.

These multi-resolution elements are still linear in their displacements. Therefore, they perform poorly in terms of accuracy if there is a high variation of material properties within the elements. However, the accuracy is still sufficient to reduce the residual in a multigrid context. Furthermore, the element matrices can be pre-integrated for the mesh, and scaled with the respective Young's modulus during optimization. This approach can be quite computationally efficient.

The multi-resolution element approach was successfully used in the OpenMP implementations presented in **[P4]**. Here it was observed that this approach did require more iterations to solve the resulting system, compared to using a Galerkin projection. However, the benefit is that all coarse grid operators are constructed efficiently with a comparably simple implementation.

Smoothing

Smoothing is the operation of reducing the error at some grid level. Several candidates for smoothing operation exist, and the choice can have a drastic effect on the computational efficiency of the resulting multigrid strategy.

The purpose is to reduce the residual of some level $||\mathbf{f}_i - \mathbf{K}_i\mathbf{u}_i||$ with respect to some force, displacement, and matrix. Usual smoothing is either application of a preconditioner or few iterations of an iterative solver, as these have been designed for similar purposes. However, these are usually only applied for a small fixed number of iterations, usually between 1 and 4.

Usual choices for preconditioning as a smoother are

Jacobi. A diagonal approximation of the stiffness matrix. It is easy to implement and embarrassingly parallel during execution. However, it is not very effective at reducing the residual compared to other choices [2, 17].

SOR - Successive Over Relaxation but offers much better reduction of residual compared to a Jacobi iteration, while being slightly more involved to implement. However, the standard definition of SOR cannot be parallelized. Several variations exist to allow parallel execution of the preconditioner. Examples include performing SOR independently on every distributed partition, as done in PETSc [20], performing SOR independently on every node of the finite element mesh [21], or using a so-called colored variation [2].

In general, any variation of SOR tends to perform better than the Jacobi preconditioner. When using an iterative solver as a smoother, one of the above preconditioners are usually applied to the iterative solving method. Good strategies for iterative solvers as smoothers include

GMRES Generalized Minimum Residual, which is a general iterative solver. It works well with non-symmetric matrices, and is generally very robust [2].

One downside is that it requires additional memory compared to alternate methods, as it needs to store a history of vectors. However, when used for a small fixed number of iterations as a smoother the memory requirements tend to be low.

CG Conjugate Gradients, which only works for symmetric positive definite matrices [2]. Fortunately, the stiffness matrix associated with linear elasticity is symmetric positive definite.

Chebyshev is an iterative solver that works very well for smoothing. First of all, it can be tuned to very effectively reduce the high-frequency part of the residual, as desired. Secondly, it requires no inner-products which take significant time to compute in a parallel setting. Unfortunately, the Chebyshev iteration requires an estimate of the upper eigenvalue of the considered matrix, which can be costly to compute. The Chebyshev solver also requires the considered matrix to be positive symmetric definite.

A more sophisticated smoother will reduce the error faster, resulting in fewer total V-cycle applications before the desired tolerance is achieved. However, more sophisticated smoothers usually require more computational effort at every application, resulting in an increase of work required for every V-cycle application. Hence, a good strategy is one which results in the fastest solution requires some balance between error reduction and time spent smoothing.

Another practical consideration is effort of implementation. More complex smoothing strategies, such as SOR preconditioned Chebyshev, are significantly more demanding to implement than simpler approaches. If the considered application is within a framework which has access to many iterative solvers and preconditioners, such as PETSc [20], effort of implementation can be neglected. More importantly, access to a wide range of smoothers allow fast and easy comparison of performance between all available approaches.

Coarse Grid Correction

A special case of smoothing is the coarsest level in the grid hierarchy, the so-called coarse grid correction. Usually a direct solver is employed, to solve the system exactly. Here, the matrix factorization can be stored and reused for every V-cycle application, reducing the computational time. Our observation is that a direct solver usually is the best approach in terms of total time spent. If

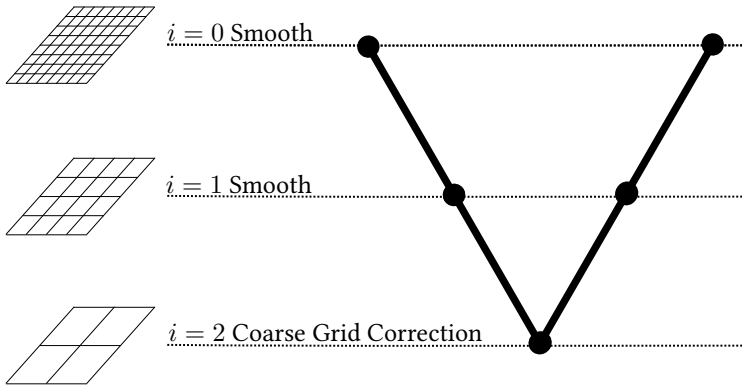


Figure 2.3: Illustration of the V-cycle application. The procedure moves left to right, smoothing from finest to coarsest grid. After the coarse grid correction, smoothing is performed from coarsest to finest grid.

possible, enough grids should be employed to make the direct solver feasible on the coarsest grid.

An alternate approach, if a direct solver is not feasible, is to run an iterative solver for a fixed number of iterations. Usually an iterative solver as coarse grid correction is run using a large fixed number of iterations, typically 30 to 1000, to reduce the error significantly. For instance, the large-scale results presented in Aage et al. [22], Baandrup et al. [23] used an algebraic multigrid scheme as the coarse grid correction to the used geometric multigrid approach.

V-cycle

The used multigrid approach can now be introduced. We consider the so-called V-cycle, which is a simple traversal scheme between grids. During one V-cycle application the grids are initially traversed from finest to coarsest. At each level the problem is smoothed, and the residual of the result is restricted to the coarser grid. When the coarsest grid is reached, the coarse grid correction is performed, and the result is projected to the finer grid and smoothed. This continues until the finest grid is reached. The traversal pattern is illustrated in fig. 2.3.

The V-cycle is shown in pseudo-code in algorithm 1. The algorithm shows that it is the residual of the smoothed field which is projected down to the coarser grids, while the correction of that residual is projected back to the finer grid.

Algorithm 1: V-cycle multigrid

Data: Input force $\mathbf{f}_0$, prolongations $\mathbf{P}_i$, Operators $\mathbf{K}_i$
Result: Smoothed field $\tilde{\mathbf{u}}_0$

for $i \in \{0, 1, .., n-1\}$ **do**
  $\mathbf{u}_i \leftarrow \mathbf{0}$;
  $\mathbf{u}_i \leftarrow \text{smoothing}(\mathbf{u}_i, \mathbf{f}_i, \mathbf{K}_i)$;
  $\mathbf{r}_i \leftarrow \mathbf{f}_i - \mathbf{K}_i\mathbf{u}_i$;
  $\mathbf{f}_{i+1} \leftarrow \mathbf{R}_{i+1}\mathbf{r}_i$;
end
$\mathbf{u}_n \leftarrow \mathbf{0}$;
$\mathbf{u}_n \leftarrow \text{coarse correction}(\mathbf{u}_n, \mathbf{f}_n, \mathbf{K}_n)$;
for $i \in \{n-1, n-2, .., 0\}$ **do**
  $\mathbf{u}_i \leftarrow \mathbf{u}_i + \mathbf{P}_{i+1}\mathbf{u}_{i+1}$;
  $\mathbf{u}_i \leftarrow \text{smoothing}(\mathbf{u}_i, \mathbf{f}_i, \mathbf{K}_i)$;
end

By repeatedly applying the V-cycle the system of equations can be solved. Another, more effective, way of solving the equations is to use the V-cycle as a preconditioner for an iterative solver. Since the stiffness matrix is symmetric and positive definite (SPD), the conjugate gradient method is an apt choice [2]. Formally, the requirement is that the preconditioner should also be SPD, although all of the presented approaches have been found to work acceptably in practice.

While only the V-cycle is used in this dissertation and associated articles, there exist other cycle types. Both W- and Full-cycles perform the coarse grid correction multiple times for every cycle application. The efficiency of the various cycle types is dependent on the considered problem, and should therefore be chosen accordingly. For the elasticity problems with heterogeneous material parameters, the V-cycle decreases the residual sufficiently fast, while using comparably little computational effort every cycle, hence the usage throughout the dissertation.

2.2 Multigrid on Unstructured Grids

When considering unstructured grids most definitions and considerations from section 2.1 still hold. The main noteworthy differences between these methods

are that coarse grids are no longer trivially constructed from fine grids, this extends to the construction of prolongation and restriction operators.

Some coarsening algorithms exist, which remove nodes from a grid to obtain a coarser representation. Although these algorithms are usually restricted to simplex meshes [24–26]. Since topology optimization usually favours non-simplex elements, usually tri-linear hexahedrals, this work is focused on manually generating all meshes of the hierarchy from the initial geometry representation. As the meshes are not generated by coarsening, special care needs to be taken when constructing the prolongation operator as discrepancies between the discretizations might exist.

Constructing Prolongation Operators

Constructing the prolongation between unstructured meshes by the element shape-functions is fundamentally no different from structured meshes. The key differences are bookkeeping and catching edge-cases.

Bookkeeping refers to the challenge of finding nodes of the fine mesh, which are located inside the elements of the coarse mesh. A naive approach is to check every fine node for every coarse element, to check whether the node resides within the element. This process is sped-up significantly by using a KD-tree, or similar data-structure that allows fast spatial look-ups. Figure 2.4 illustrates an example of nodes of the fine grid found in the vicinity of the considered element.

Special care needs to be taken when a node of the fine mesh is not within any coarse elements. This can occur on the boundary of the domain, where the meshes might not overlap, as illustrated by the red squares in fig. 2.4. In this case alternate weights must be used in order to generate a suitable interpolation for these fine nodes. These nodes can be coupled to nearby coarse nodes with weights found by the modified Shepard's method [27].

$$w_k = \left(\frac{\max[0, R - ||\mathbf{x} - \mathbf{x}_k||_2]}{R||\mathbf{x} - \mathbf{x}_k||_2} \right)^2 \tag{2.2}$$

Here w_k denotes the partition-of-unity weight used to couple a point with coordinates $\mathbf{x}$, to a neighboring point with coordinates $\mathbf{x}_k$. R is a suitable radius parameter, in the same order of magnitude as the coarse element side length. Shepard's method provides a partition of unity by construction, similarly to the the shape-function.

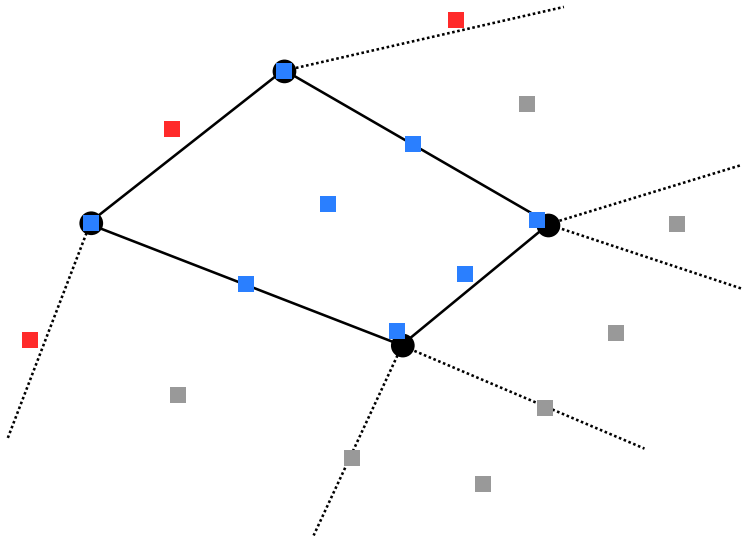


Figure 2.4: Illustration of matching nodes of the fine mesh, squares, to an element of the coarse mesh. Blue squares indicate fine nodes which were found to be inside the considered element. Grey squares indicate nodes which were not found to be inside the element. Red squares indicate nodes which were not found to be inside any element.

Algorithm 2 informally summarizes an algorithm to generate prolongations between two unstructured meshes. Note that the construction of a KD tree K is not strictly necessary, but in practice greatly improves the efficiency of the algorithm. The first part of algorithm 2, lines 1 to 11, attempts to generate all interpolation weights using shape-functions, while keeping track of which nodes succeeded. The second part, lines 12 to 19, identifies nodes for which no successful shape-function weights were found, and uses Shepard's method to generate weights for these nodes.

Due to the complexity of constructing the prolongation, using the prolongation for restriction $\mathbf{R}_i = \mathbf{P}_i^\top$ can simplify unstructured implementations. When using the prolongation as restriction all rows of $\mathbf{P}_i^\top$ should have at least one non-zero value, corresponding to all coarse nodes receiving a value during restriction. Otherwise, Galerkin projected matrices become rank-deficient. Usually, this is not an issue for well-formed meshes. If however this should be ensured

algorithm 2 can be extended with an additional step similar to lines 12-19 for the coarse nodes.

Algorithm 2: Construction of prolongation between unstructured grids

Data: Fine mesh $\mathcal{H}^{i-1}$, coarse mesh $\mathcal{H}^{i}$
Result: prolongation operator $\mathbf{P}_i$

```
1  K ← KD tree of all nodes in H^i;
2  n_touched ← zero valued array of size nodes in H^{i-1};
3  forall elements e ∈ H^i do
4      N_found ← nodes in K near center of e;
5      forall nodes n ∈ N_found do
6          if n is within e then
7              n_touched[n] ← n_touched[n] + 1;
8              Insert shape functions of e, for position of n, into appropriate
                 indices of P_i;
9          end
10     end
11 end
12 K_c ← KD tree of all nodes in H^i;
13 forall nodes n ∈ H^i do
14     if n_touched[n] is 0 then
15         R ← sufficiently large local radius, such that n_found ≠ ∅;
16         n_found ← nodes in K_c with distance to n less than R;
17         Insert weights from eq. (2.2) into appropriate indices of P_i;
18     end
19 end
```

The presented methodology can also be employed with distributed meshes, i.e. a mesh distributed between several memory-spaces, usually implemented with MPI. Some care must be taken to ensure that the coarse grids are distributed in memory according to the spatial location of the fine grids. Also, some synchronization of the n_{touched} array in algorithm 2 must be included after line 11, to ensure agreement between processes.

Practicalities of Generating Coarse Grids

When generating coarse meshes independently, several practical issues might arise. Firstly, narrow features might exist in the geometry. These are hard to resolve in the coarser grids. This can become an issue in generating well-formed coarse grids, which are crucial to attain a sufficiently fast convergence. To overcome this issue, one can manually enlarge these small regions, or merge them with neighboring regions, in order to attain a workable coarse grid.

Secondly, the multigrid method improves drastically if care is taken such that nodes of all grids in the hierarchy coincide as much as possible, similar to the structured case. The coinciding nodes will result in much sparser prolongation matrices $\mathbf{P}_i$, as coinciding nodes, as well-as nodes on surfaces and edges will result in fewer non-zeros. This in turn drastically increases the sparsity of the Galerkin projection coarse grid matrices, eq. (2.1), reducing computational effort of smoothers and the coarse grid correction. Methods to achieve coinciding nodes depend on the used meshing algorithm. For a cut-and-smooth strategy as used in the Sculpt program [28], care must be taken that the initial structured grids match.

For other meshing strategies, it can be harder to ensure overlapping nodes between grids. If feasible, the coarsest mesh can be generated first, and refined subsequently to generate the hierarchy. This is usually only feasible for domains without curved surfaces.

2.3 Multigrid for Shell Elements

Applying the multigrid method to shell elements is a somewhat straightforward extension to the unstructured multigrid approach. Similar to extending the structured to unstructured approach, the main difference is defining an appropriate prolongation operator. This section briefly introduces the extension of the unstructured multigrid to shell elements. Example applications are presented in **[P1]**.

For this section we consider a standard thin-shell element formulation, such as the curved isoparametric quadrilateral shell element described in Cook [3]. The techniques discussed in this chapter are general for most shell element formulations, so long as the degrees-of-freedom are defined on the element nodes.

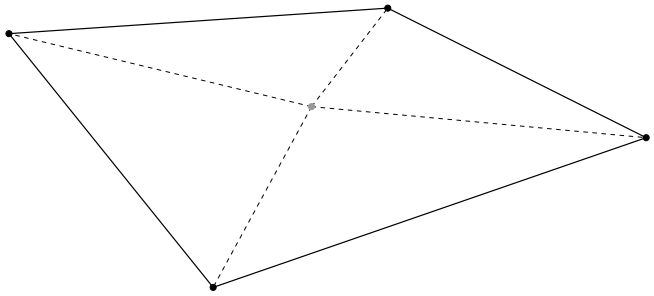


Figure 2.5: Illustration of a quadrilateral shell element. The center point, found by interpolation of corner nodes, is also shown. The illustration shows that the four corner nodes do not lie in a plane.

Constructing Prolongation Operators

The process of constructing prolongation operators is complicated further for shell elements. The main issue to tackle, is defining when a node is inside a surface element, that is line 5 in algorithm 2. Since the elements are now surfaces with no volume, the notion of "inside" needs reconsideration.

The point which is tested can be projected to the surface of the shell element, where we can test whether the node is inside the element. Unfortunately, quadrilateral surface elements do not uniquely define a surface. This is illustrated in fig. 2.5, where the center node is drawn explicitly along with guiding lines.

To overcome the issue of multiple surfaces, we need to choose some planes onto which we project. One robust approach is to identify the three nearest corners of the quadrilateral, and consider the plane spanned by these. Alternately, the two nearest corners can be used along with the center-node, obtained by interpolation of the corner nodes using the element shape-functions. Either of these approaches work well enough, as they will crucially project correctly near the element edges.

Issues with Condition Numbers

The condition number of the resulting matrix is an important factor to remark when working with shells. The definition of the condition number is $\frac{\lambda_{\max}}{\lambda_{\min}}$, i.e. the ratio between highest and lowest eigenvalue. High condition numbers of matrices usually complicate solving their associated linear system of equations. A

rule of thumb is that n digits of precision are lost when the matrix has condition number 10^n, when solving with an otherwise "exact" method such as a LU factorization [29]. Most implementations try to mitigate this as much as possible by some preconditioning of the system.

Shell structures have a tendency to exhibit high condition numbers, compared to regular three-dimensional elastic discretizaitons. This can be due to the inclusion of rotational degrees-of-freedom, which are usually a different order of magnitude in the numerical representation, compared to displacements. High-condition numbers also arise from the physical system itself, due to the large difference in stiffness of in-plane and out-of-plane deformation modes, which manifests in the highest and lowest eigenvalues. High condition numbers are usually also found in the resulting black-and-white designs from topology optimization due to the high contrast in stiffness, hence topology optimization with shell structures include two factors which are known for high condition numbers.

Iterative solver strategies, such as the discussed multigrid approach, are also affected by the high condition numbers. The high condition numbers usually result in a smaller decrease of residual every iteration. This is problematic, as it results in an increase of iterations required to achieve any desired tolerance. Since this is caused by the physical system and discretization, few options are available to solve the conditioning problem. Hence, the largest problems presented in **[P1]** are "only" of 11 million shell elements, and require many conjugate gradient iterations for convergence compared to similarly sized full 3D problems. Luckily, due to the two dimensional properties of surfaces, 11 million elements is sufficient to achieve very low element sizes.

Concluding Remarks

In this chapter we have summarized various multigrid methods for structured, unstructured, and shell meshes. These methods form the foundations of all topology optimization approaches presented in the following chapters. Special care must be taken when composing a multigrid method as the efficiency of the method is dependent on grids and the problem itself. For unstructured and shell meshes definition of the coarse grids and prolongation and restriction operators greatly affect the resulting efficiency.

Chapter 3

An Advection-Diffusion based Filter for Milling

Topology optimized designs are usually optimized solely for structural performance, resulting in final designs which can be impossible, or prohibitively expensive, to manufacture. This has motivated the creation of modifications to the topology optimization problems which ensure manufactureability. These modifications are usually defined as filtering techniques or constraints to the optimization problem. This chapter treats such a modification which ensures that a design can be machined through the application of a filter which is designed for efficient application to unstructured meshes **[P2]**.

3.1 The importance of PDE-based filters in compute clusters

To avoid modeling artifacts during topology optimization filtering techniques are commonly used [6, 30]. These techniques, which were originally designed to regularize the problem, have since then been extended to provide control of the resulting geometry such as ensuring a minimum length-scale in the resulting structure [31, 32] or generating porous structures [33] through constraints. The filtering formulations are also beginning to see use in shape optimization, where they are used primarily for regularization [34–36]. The presented methodology to ensure manufacturability by milling is implemented as a control of the optimized geometry by filtering. That is, the underlying design variables are mapped to a "physical" density field which represents the structure. This mapping ensures that the resulting structure can be machined.

The two most common approaches in filtering are either applying a convolution operation, inspired by image analysis techniques, or solving a Partial Differential Equation (PDE). The main novelty of our proposed method is to substitute a convolution filter with a PDE-based filter.

When using a convolution type approach some, usually compact, filter kernel is chosen [6, 7]. The filtered field is found by computing the convolution of the input field and the filter kernel. Larger filter kernels result in larger local domains for the computation of every element value. Hence, there is an undesirable increase in required computation effort.

A filter formulation based on solving partial differential equations, the so-called PDE-filter, was developed by Lazarov and Sigmund [8]. This filter was developed specifically in order to circumvent the computational effort associated with large filter kernels, along with significantly easing the computation on

unstructured and distributed meshes. Here an effect similar to the convolution is obtained by solving a modified Helmholtz-type partial differential equation. The main advantage of using this PDE-filter is that constructing the equations of the discretized modified Helmholtz problem only requires local knowledge of every element, regardless of the chosen radius for the filter or structure of the finite element mesh. This significantly eases computation for distributed and unstructured meshes. The drawback of using a PDE-based filter is that a system of equations now need to be solved to find the filtered field. However, the system of equations arising from the modified Helmholtz equations are in practice solved very fast by iterative solvers.

3.2 The Advection-Diffusion Based Filter

The publication **[P2]** presents a method to ensure that designs obtained by topology optimisation can be manufactured by milling. The method is inspired by the described Helmholtz-based filter technique, and the milling filter formulation presented by Langelaar [37]. In fact, the general outline of this approach is similar to the that proposed by Langelaar [37], the main difference being substituting a transfer of density variables between meshes and a cumulative summation with a solution to the advection-diffusion equation.

The notion of whether a design is manufacturable by milling needs further elaboration. In the presented approach, it is simplified to mean that all void regions of the design can be reached by some predefined tool direction from a border of the considered design domain. This means, the presented approach does not consider some important physical aspects of milling, such as tool thickness and vibration due to long overhangs. These potential issues need to be addressed in post-processing. Also, the formulation includes an implicit assumption of a convex design domain. In case the design domain is not convex, the advection diffusion equation should, in principle, be solved on the convex hull of the domain.

Both Langelaar [37] and Lee et al. [38] present milling filters based on the classical convolution based filtering techniques. These require efficient spatial lookup of elements to be efficiently implemented, which is not necessarily trivial in unstructured or distributed meshes. However, the convolution approach lends itself to explicit control tool size and shape, something which is not trivial in this variation.

An alternative formulation is presented in Hur et al. [39], which is based on the advection-diffusion equation like the approach presented in this chapter, but only using Dirichlet boundary conditions of value zero, which requires solving the filtering equations on an extended domain to enable material placement on the boundaries of the design domain. Similarly, Gasick and Qian [40] also uses the advection-diffusion, splitting boundary conditions to Dirichlet upstream and Neumann downstream of a given tool direction, and formulates the manufacturability as a constraint. The presented method from **[P2]** uses the advection-diffusion equation similarly, but using Robin boundary conditions, which enables solving the advection-diffusion equations on the mesh used for the elasticity equation using a single boundary condition.

The steps of the proposed filtering technique is shown in fig. 3.1. Initially, a smoothing operation is applied through either the classic density filter [6, 7], or the PDE-based filter [8]. Afterwards, the advection-diffusion equation is solved for every tool direction, resulting in a "shadowed" field for every tool direction. The purpose of these shadowed fields is to project to solid everything which is not reachable from that specific tool direction. These fields are combined using a differentiable approximation to the min function for each element value. Here, an element becomes void if any tool is able to reach it. Finally a differentiable approximation of the Heaviside projection is applied in order to project the resulting values to the feasible range $[0, 1]$.

The advection-diffusion equation used for step 2 in fig. 3.1 is shown in eq. (3.1) along with the used boundary conditions. Here $\tilde{x}$ denotes the density filtered field, while $\hat{\tilde{x}}$ denotes the resulting solution. The advection direction u_i is kept constant for the entire domain, usually chosen such that it has unit norm $||u_i||_2 = 1$. A scaling parameter $s \geq 1$ is added to the problem, with the purpose of increasing the solution magnitude, ensuring that downstream areas are projected towards the upper density bound.

$$u_i \frac{\partial \hat{\tilde{x}}}{\partial x_i} - \frac{1}{Pe} \frac{\partial^2 \hat{\tilde{x}}}{\partial x_i^2} = s\tilde{x} \quad \text{in } \Omega \tag{3.1}$$

$$\hat{\tilde{x}} + \mathbf{n} \frac{1}{s} \nabla \hat{\tilde{x}} = 0 \quad \text{on } \Gamma_R \tag{3.2}$$

$$\hat{\tilde{x}} = s \quad \text{on } \Gamma_D \tag{3.3}$$

Dirichlet boundary conditions, eq. (3.3), can be applied to the surfaces border-

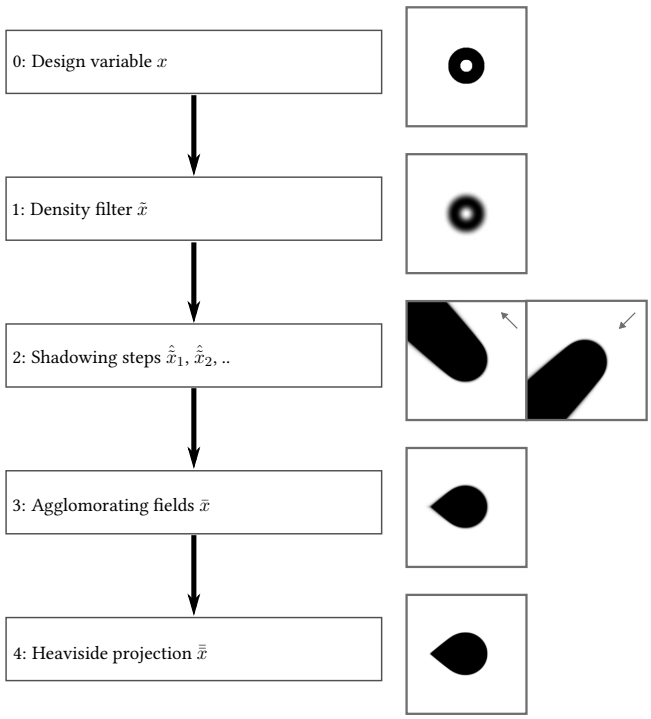


Figure 3.1: Diagram of the various steps in the advection-diffusion based milling filter. Figure taken from **[P2]**.

ing passive solid domains, denoted Γ_D, ensuring that the passive solid domains will cause a "shadowing" effect into the design domain, avoiding unreachable void caused by the solid domains. The Robin boundary conditions, eq. (3.2), are used for all other domain boundaries, which are denoted Γ_R. These boundary conditions allow material near the upstream boundary and work as Neumann boundary conditions downstream.

The Peclet number $Pe > 0$ determines the ratio of advection and diffusion in the equations. A high Peclet number indicates an advection dominated problem, while a low Peclet number indicates a diffusion dominated problem. Since we are interested in solving an advection-like problem, high values are usually chosen, such as 10^4. The choice of advection-diffusion, instead of a pure advection problem is mainly due to the increased numerical stability when including a

small amount of diffusion to the problem. It was however found sufficiently fast to be workable for the three-dimensional results presented in the following section. If a sufficiently stabilized solution to the advection equations is available, this can be used instead of the advection-diffusion equations, as shown in a subsequent commercial implementation of this method [41].

Solving the Advection-Diffusion equations

Numerical instabilities are important to consider when discretizing the advection-diffusion equations with high Peclet numbers. In the presented studies a first order up-winding finite volume scheme was used, which stabilizes the equations to enable solutions without spurious oscillations.

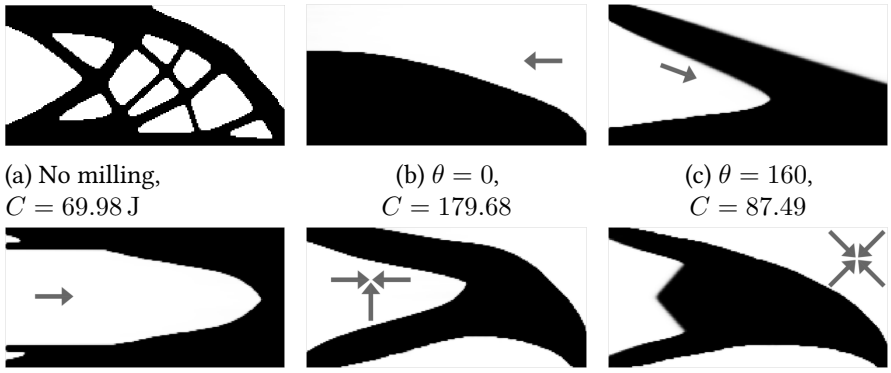
If the system is sufficiently small, as is the case for the presented two-dimensional results, a LU-factorization can be employed to solve the system of equations. A big advantage of the direct approach, is that the factorization needs only be computed once, as the filter system is not dependent on the design.

The resulting equations are however not trivial to solve with iterative solvers, due to the high degree of non-locality in the resulting solutions. It was found that for distributed systems the flexible generalized minimal residual solver with an additive Schwartz preconditioner, implemented in the PETSc framework [20] worked well. This approach still requires a considerable amount of computational effort compared to a specialized method, such as a multigrid preconditioning scheme.

3.3 Examples using the Filter

The advection-diffusion filter is used for linear compliance minimization with a volume constraint in two and three dimensions. For the two dimensional case a classic cantilever problem is considered. The filter equations are solved using a LU-factorization. The full details of the loading and boundary conditions can be found in **[P2]**. Some results for various milling directions are shown in fig. 3.2.

A reference case using the robust formulation [31, 32], is shown in fig. 3.2a. The robust formulation is chosen as a reference as it produces a near discrete design consisting of only 0 and 1 density values, like the proposed milling filter. Here, we see the resulting compliance value of $69.98\,\mathrm{J}$, which will provide a best-case performance for the more constrained designs with milling constraints.



(a) No milling, $C = 69.98\,\mathrm{J}$ (b) $\theta = 0$, $C = 179.68$ (c) $\theta = 160$, $C = 87.49$

(d) $\theta = 180$, $C = 287.93$ (e) $\theta = -90, 0, 180$, $C = 86.92$ (f) $\theta = \pm 45, \pm 135$, $C = 115.14$

Figure 3.2: Reference design and results with one or multiple tool directions. Computed using 200×100 elements. The final projected variable is shown. Figures taken from **[P2]**.

All designs presented in fig. 3.2 only have void regions which are reachable from outside the domain by the desired tool directions, illustrated by the arrows in the respective figures. Also, all compliance values of structures with milling filter are higher than the reference using the robust formulation. The best performing structures with milling constraints, figs. 3.2c and 3.2e, still have an increase of approximately 25%. Less favorable milling directions increase the compliance value fourfold.

The drastic reduction in compliance value compared to the reference design is expected since the milling filter results in a great reduction of design freedom. Notably, no interior voids are allowed with the milling filter, although these are heavily used in the reference design. Also, some directions, such as the one shown in fig. 3.2d, disallow removal of material in unloaded regions.

In the three-dimensional case a similar study is performed. Here the GE-engine bracket design is considered. Details of the used loading conditions are given in **[P2]**. Again, the robust formulation is used to generate a reference design without milling constraints, shown in fig. 3.3. Here, the material envelopes the volume, creating large hard-to-reach areas inside the structure.

Adding a single milling direction from "above" we obtain the structure shown in fig. 3.4 which consists of a series of plates. Note that some overhang exist

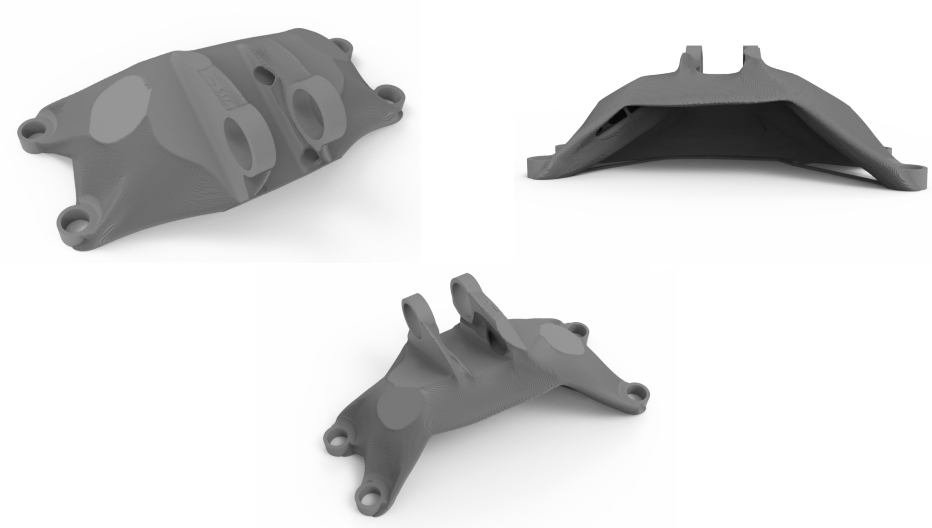


Figure 3.3: Reference design of the bracket example with Robust formulation and consistent boundary condition for PDE-filter. $C = 4.21 \times 10^7$ J. Figures taken from **[P2]**.

beneath the fastening rings where the load is applied. This is due to the design domain mesh, which contains no elements in this area. This structure is found to perform significantly worse than the reference design, which is to be expected due to the limitation of design freedom imposed by using only a single milling direction.

Using five milling directions we achieve the structure shown in fig. 3.5, which also indicates the used tool directions. We see that this structure is more compact, and uses the tools from the side to remove material underneath the structure to create thick beam-like structures. The compliance value of this structure is quite close to the reference value, where the design restriction results in a reduction of around 20% compliance, similar to the two-dimensional results.

As a final example, to show a deficiency of the proposed milling filter, a single milling direction at a slight angle to the direction from "above" is considered. The resulting structure, shown in fig. 3.6, indicates a clear flaw in the filter. A large central part of the structure has a hole which violates the milling direction. This is due to the diffusive effects of the solution to the advection-diffusion equation, which then result in values under the "bridge" falling under the threshold value

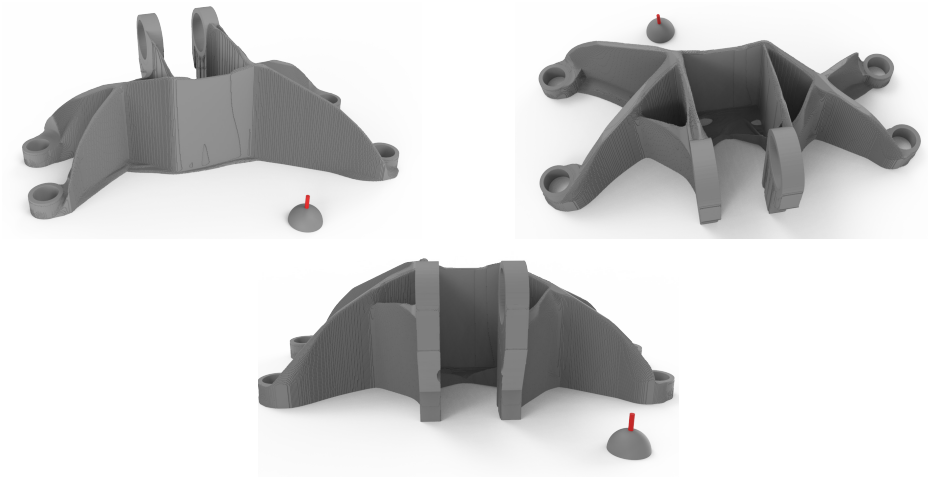


Figure 3.4: Design of the bracket example with one milling direction in the z-axis. The tool direction is indicated by the red line next to the structure. $C = 9.28 \times 10^7$ J. Figures taken from **[P2]**.

of the Heaviside pass. This type of phenomenon can occur in thin structures with a single tool direction.

Concluding Remarks

Solving finely resolved topology optimization problems is not possible without efficient solution of the state equations. However, this is not sufficient if advanced design control such as design for a specific manufacturing method is desired. The presented method for ensuring manufacturing by milling is shown to work for large unstructured grids, enabling fine resolution for the designs with manufacturing constraints.

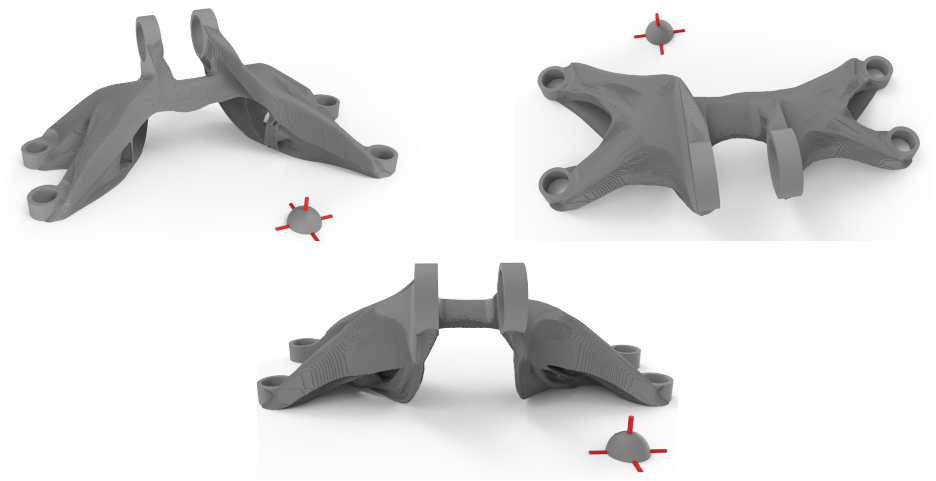


Figure 3.5: Design with five milling directions. The tool directions are indicated by the red lines next to the structure. $C = 4.89 \times 10^7$ J. Figures taken from **[P2]**.

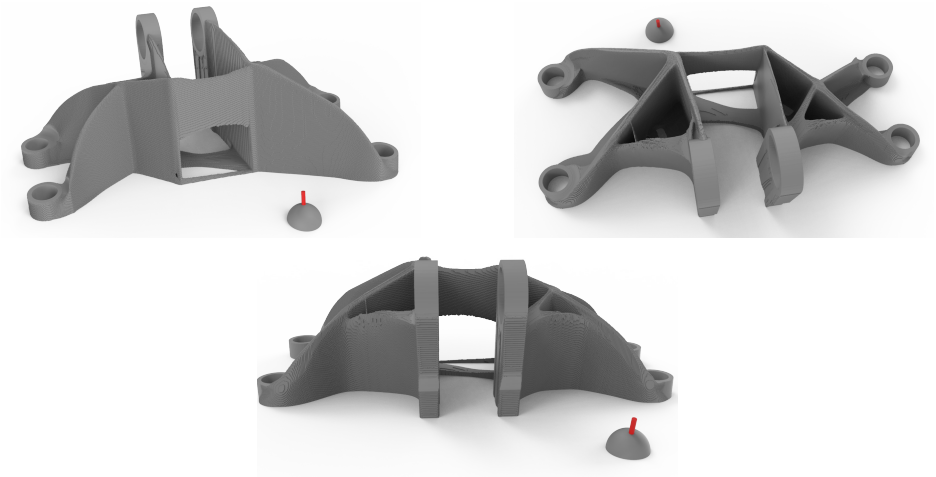


Figure 3.6: Design with one milling directions at an 14.4 degree angle. The tool direction is indicated by the red line next to the structure. $C = 9.52 \times 10^7$ J. Figures taken from **[P2]**.

Chapter 4

A Case Study of a Direct-Drive Wind Turbine

In order to show an example usage of the large-scale distributed multigrid discussed in chapter 2, we perform a case-study on the rotor of a direct drive wind turbine. Here a rotor used to connect the shaft of the wind turbine wings with the generator is considered. Here, it is sought to reduce the weight of the rotor while maintaining maximal displacements in axial and radial directions. We study this problem to show the potential of large-scale topology optimization with unstructured grids. This study is a brief summary of **[P3]**, with focus on the optimization aspects of the article.

4.1 Defining the Optimisation Problem

The rotor for the direct drive wind turbine is fitted with a series of magnets on the larger radius, which are placed inside a coil to generate electricity. A force is induced on the magnets during this process, which is dependent on the distance from the magnet to the coils. In order to obtain a simpler mechanical model several simplifications are made. Firstly, the resulting forces are treated as evenly distributed surface tractions over the outer surface. Secondly, it is assumed that there is no interaction between the rotor and the magnetic field, i.e. the forces are not design dependent. Finally, it is assumed that the forces are constant, and not affected by the reduction of air-gap due to displacement of the rotor. The final simplification is done, as we will assume small deformations, and constrain the allowable deformation of the outer surface in the optimization problem.

The rotor is subject to three loads, a radial load from the coil, a torsional load which occurs during breaking, and an axial load which occurs during transport of the rotor. These loads are all assumed to be static for simplicity. All loads are illustrated in fig. 4.1, while their magnitudes are given in table 4.1. The air-gap between rotor an coil gives rise to some maximally allowed displacements, summarized in table 4.1. The radial and torsional displacements are defined at the outer surface of the rotor, while the axial displacement is defined at the central part of the rotor.

The loads are separated during modeling, to ensure that cross-interactions of the loads are not used by the optimization. The three displacement constraints are matched to the corresponding load-case in the same direction. It was found that the gravitational loading and axial displacement constraint were trivially satisfied by nearly all structures. Therefore this load and constraint were removed

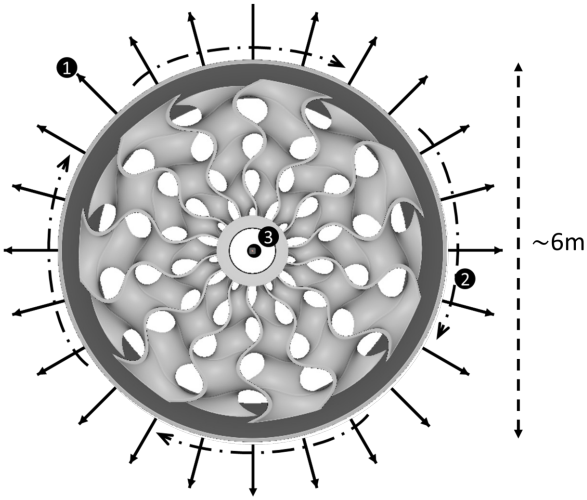


Figure 4.1: Rotor loading conditions. Figure from **[P3]**.

Loading		**Critical Deflections**	
1 Normal Maxwell Stress	$0.2\,\mathrm{MPa}$	Radial	$< 0.65\,\mathrm{mm}$
2 Shear Stress	$40\,\mathrm{kPa}$	Torsional	$< 2.84\,\mathrm{mm}$
3 Gravitational Loading	$9.81\,\mathrm{m/s^2}$	Axial	$< 32.17\,\mathrm{mm}$

Table 4.1: Rotor loading conditions and critical deflections.

from the optimization problem.

Based on the simplifications of the mechanical problem, we now define the

optimization problem as

$$
\begin{aligned}
&\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} && \frac{1}{\sum_{e=1}^{n_e} v_e} \sum_{e=1}^{n_e} v_e \hat{\hat{\mathbf{x}}}_e \\
&\text{subject to:} && \\
&\text{state equation} && \mathbf{K}(\hat{\mathbf{x}})\,\mathbf{u_i} = \mathbf{f}_i, \quad \forall i \in \{1,2\} \\
&\text{radial deformation} && P_m(D_{\text{radial}}(u_1)) \leq \frac{0.65\text{mm}}{s_f} \\
&\text{torsional deformation} && P_m(D_{\text{torsional}}(u_2)) \leq \frac{2.84\text{mm}}{s_f} \\
&\text{box constraint} && 0 \leq \mathbf{x}_e \leq 1, \quad \forall e \in \{1,..,n_e\}
\end{aligned}
\tag{4.1}
$$

Where the objective is the total volume used by the design, subject to several constraints. Firstly, the state equations are included as an equality constraint, although it is in practise treated as an implicit computational step to find the radial and torsional deformation constraints. Special operators are introduced to extract the deformation in radial D_{radial} and torsional $D_{\text{torsional}}$ directions of the outer surface. An approximation to the maximal value of these deformations is found using the p-mean aggregation P_m. Of special interest is the safety-factor parameter s_f, which is used to tighten the allowed deformation. Details on several operations are omitted for brevity. Omitted details are found in **[P3]**.

The full cylindrical domain is modeled. In order to ensure that symmetric structures are able to appear, the structure was meshed by a single symmetric wedge, which was replicated six times. Furthermore, care was taken to solve the state equations to a high degree of accuracy, of relative residual below 10^{-11}, to ensure no asymmetry is introduced through errors in the displacement field.

4.2 Results

Three different designs were reported in **[P3]**. One design with safety-factor $s_f = 1$, and two different designs for safety-factor $s_f = 2$. It was chosen to present two different designs for the same problem to highlight that the problem was found to be sensitive to initial conditions. Specifically the initial homogeneous density parameter value gave rise to different designs. This is a direct consequence of the non-convexity of the optimization problem.

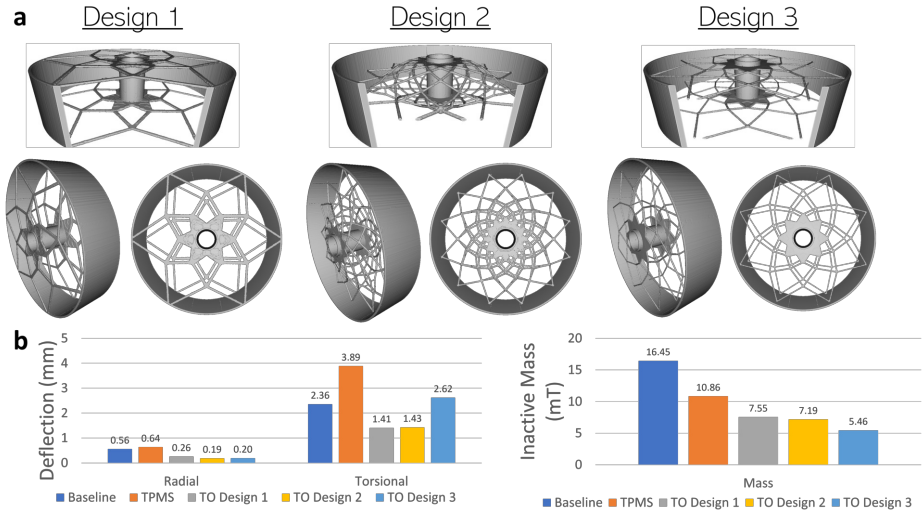


Figure 4.2: Resulting designs and their respective mass and critical deflections. Figure from **[P3]**.

The three presented designs are shown with their performances in fig. 4.2. Design 1 and 2 are made for the high safety-factor of $s_f = 2$, while design 3 is made for the lower $s_f = 1$. It can be seen that the torsional case is dominating for all three designs, as they all satisfy the radial displacement constraint with a comfortable margin. It is also clear that design 3 has a significantly decreased mass, as the maximal torsional deflection is significantly larger than the two other designs. This is also notable in the design itself, where all structural members are thinner for design 3.

Upon close inspection it can be seen that the torsional deformation of design 2, $1.43\,\mathrm{mm}$, is slightly higher than the allowed value of $1.42\,\mathrm{mm}$. This is due to the approximation nature of the P-mean aggregation, which can result in inexact control of the actual maximal value.

Both design 1 and 3 use only in-plane spokes near the boundaries of the rotor. Their shapes are both six-fold rotational symmetric, similarly to the underlying mesh. Notably, design 1 uses two layers thicker spokes, which are fixed to plates near the rotor center. Design 3 has a smaller inner plate, and instead adds a small truss near the inner plate. Design 2 is drastically different, where spokes begin near the boundaries, and merge to a single spoke in the central plane of

the structure. This allows design 2 to obtain a lower volume than design 1, while still remaining feasible in the aggregated maximal displacement constraint.

Concluding Remarks

The value of large-scale topology optimization has been shown by study of a direct drive wind turbine. It is clear that resolving the thin spokes and plates of the final designs require a high spatial fidelity of both design and elastic equations. While not discussed in this summary, it can be seen in **[P3]** that the topology optimized designs outperform the considered alternatives significantly.

Chapter 5

GPU Acceleration

The focus on how to implement large-scale topology optimization is changed in this chapter. Instead of considering traditional compute clusters, which use a set of computers connected through some fast interconnect, we now consider using graphical processing units (GPUs) to accelerate the computations. By accelerating the computation sufficiently, it becomes unnecessary to utilize multiple computers, as a single desktop computer becomes able to solve sufficiently large topology optimization problems.

In order to achieve the goal of solving large systems on a single desktop, Cartesian grids, and the problem simplifications they enable, are used. This includes offline integration of stiffness matrices, implicit indices for neighbors and coarsening, and the ability to convert the finite element method into a stencil approach.

Modern compute clusters also include computers with GPU acceleration, which can be used to reach even larger scales. However, in this work only single-GPU implementations have been considered. The reasoning for this is two-fold; single-GPU implementations are also a precursor for multi-GPU implementations, so developing an efficient single-GPU implementation a necessary step towards multi-GPU methods. Secondly, not all would-be practitioners of large-scale topology optimization have access to the large compute clusters. If it can be shown that it is feasible to solve sufficiently fine discretizations using a single GPU an expansion to a distributed framework is no longer strictly necessary.

The structure of the Cartesian grid is exploited to improve the computational efficiency in these implementations. This is a common approach within simple topology optimization applications, e.g. Sigmund [42] uses the 2D grid extensively. This approach can be limiting when considering optimizing an arbitrary design domain with passive regions. Some solutions exist for this issue, such as adaptive refinement or hard-kill methods, although not explored in this work.

5.1 Strategies for GPU Acceleration

In many ways, a GPU used for general purpose computing can be thought of as a large vector processor. Through this lens, programs for GPUs should be similar in structure to those targeting vector extensions on modern CPUs. Both programs express the parallelism by applying a single instruction to multiple pieces of data. For scientific computing it is common to implement the parallelism in a

single program, multiple data (SPMD) type architecture. As the name implies, a single program is run for multiple pieces of data. This model maps well to vector processors if the programs are kept sufficiently free of branching.

Several practicalities must be considered when choosing an approach to implement GPU acceleration. Several parameters are important to consider when choosing a method to implement GPU acceleration. Ease of implementation should be considered, as this will be reflected in development time. Ease of achieving the desired performance is equally important, as the use of GPU acceleration implies that the considered software is performance critical.

The "kernel-based" frameworks for general purpose GPU programming CUDA [43] and OpenCL [44] define so-called kernels, which are functions called for every piece of data. These approaches are in accordance with the SPMD model, as the single program is made explicit. These frameworks also give explicit access to the different types of memory on the GPU, which enable the laborious programmer to achieve very high performance. However, as "kernel-based" approaches require a high degree of explicit specification, initial implementations can perform poorly, due to lack of optimization.

An alternate approach is the "pragma-based" approaches OpenACC [45] and OpenMP version 4.0 and onward [46]. These approaches take a notably different approach, by marking loops in the source code to be executed on the GPU. Then the loop-body is extracted to a kernel, to be executed over the same iteration space. These approaches give some control of the GPU memory, but in general less than the "kernel-based" frameworks.

A final approach is to use languages specifically designed to generate GPU code. Specifically, several array-based functional languages have recently appeared, where the parallelism is implicitly defined. These languages include Lift [47], Dex [48], and Futhark [49]. By only allowing homogeneous parallel operation on arrays at the language level, these languages are able to compile to kernels for the GPU. This transformation typically includes optimization passes, which improve the performance of the resulting implementations. These languages typically offer little control over memory layout and kernel formulation, as this is left to the optimizing compiler. However, they are sometimes comparable to hand-tuned "kernel-based" implementations in terms of performance [49]. Hence, a very good performance can be achieved with little effort compared to other approaches. However, the achievable performance is limited by the capabilities of the optimizing compiler, potentially rendering this approach unable

to achieve performance observed by hand-tuned kernel implementations.

5.2 A reference CPU implementation

In order to properly evaluate GPU accelerated implementations, a baseline CPU implementation was written. The motivation for writing yet another CPU based implementation was to have a CPU version which used a similar strategy for solving the system, used the same problem specific optimization, and was optimized for modern compute hardware. To ensure fair comparison it was necessary to implement a CPU version where the same structural properties of the problem are exploited, and where similar care was given to implementation details.

Another motivation for implementing an optimized CPU implementation was the implementation presented by Liu et al. [50], which showed that astonishing performance is attainable using a single CPU when proper care is taken. The implementation from Liu et al. [50] itself was unfortunately not chosen as a baseline for GPU acceleration, due to important differences in the mesh representation, and extensive software dependencies. Hence, there was a motivation to build a reference implementation using only widely available technologies.

The implementation was performed in C99 using OpenMP to enable multithreading and SIMD instructions. The CHOLMOD implementation of Cholesky factorization was used for the coarse-space correction, and the CBLAS interface was used for some non performance critical matrix-matrix products [51].

The stiffness matrix multiplication of the finest grid was found to be a performance bottleneck. Therefore several performance improvements were included.

- The iteration space was changed from elements, which is the natural choice in FEM, to nodes. This was done to simplify parallelization and avoid data-races in a lock-free manner. Every node is updated based on a stencil which includes weights based on the neighboring element values.
- A domain padding, or halo, of zero valued nodes and elements was added to the domain boundary in order to avoid branching in the stencils. Through this mechanism, boundaries are handled by reading zero values into the stencil, rather than requiring an edge-case on the domain boundaries.

- The stiffness matrix multiplication was also made to operate on an array, or pencil, of consecutive nodes within the inner iteration. This was done to enable SIMD instructions for all nodal operations. The padding was extended to ensure that no edge-case with a shorter array was necessary.

Using all of these performance improvements, the throughput of the stiffness matrix multiplication was found to be 12 GB/s, when compiled using GCC with aggressive optimization flags. This is comparable with Liu et al. [50], which achieved 16 GB/s by writing AVX512 assembly and aligning memory with memory pages.

5.3 GPU implementations

Two GPU accelerated implementations were considered for this work. The fist used the specialized Futhark language, which an array based language designed specifically for GPU programming. The second implementation is based on the CPU reference implementation, and uses OpenMP offloading to transform loops to GPU kernels. As with the CPU reference program, all of the implementations utilize the structure of a Cartesian grid to simplify the required computations.

All implementations were validated to solve the linear elasticity problem, and the topology optimization problem correctly. Figure 5.1 shows a resulting cantilever structure computed using the Futhark based implementation.

It was found that both approaches to GPU acceleration were able to implement the desired programs with relative ease. The article **[P4]** compares a code snippet of the various implementations, and refers to similar snippets written in other frameworks for GPU implementation. The conclusion of these comparisons is that it is simpler to implement GPU acceleration through high-abstraction approaches.

Performance benchmarks

Performance benchmarks are carried out for the GPU accelerated implementations. Here, the reference CPU implementation is crucial, as it allows us to compare performance to an optimized CPU implementation.

The publicly available PETSc CPU based implementation of Aage et al. [13] is also considered, to show the performance of publicly available prior work. Notice that this implementation is targeting a distributed memory setting, which

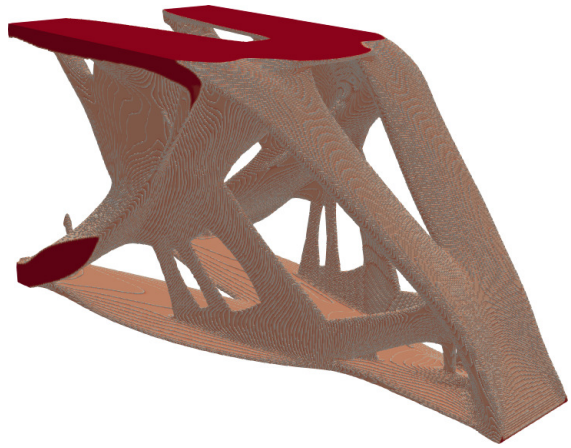


Figure 5.1: Cantilever structure optimized with 65.5 million elements, filter radius of 2.5 elements, and a volume fraction of 10%. The 100 design iterations were solved in approximately 2 hours using the Futhark implementation, using a NVIDIA A100 GPU. Figure taken from **[P4]**.

introduces additional overhead compared to our CPU reference. Also, the coarse-space correction was changed from the default settings to a LU decomposition, to better match all other implementations. This was found to reduce the wall-clock run-time across all tested mesh refinements for this implementation, compared to default settings.

The benchmarks were performed on two test machines. The implementations which do not use GPU acceleration, OpenMP-CPU and Aage et al., were tested on a machine with two Intel Xeon 8160 processors, launched in 2017 with a suggested retail price of 4.700$ each. The two implementations which did use a GPU accelerator were ran on a machine with a Intel Xeon 6226R CPU and a NVIDIA A100 GPU. The A100 was launched in 2020 at a recommended retail price of 12.500$. The purpose of using a different machine for the non-accelerated implementations was to use computing hardware in a similar price-range to the A100 in order to give some possibility of comparison. However, it is worth noting that the used GPU is significantly newer than the CPUs, resulting in better compute capabilities at the given price.

Figure 5.2 shows the wall-clock run-times for all implementations across a series of mesh refinements. It can be seen that the implementation of Aage et al.

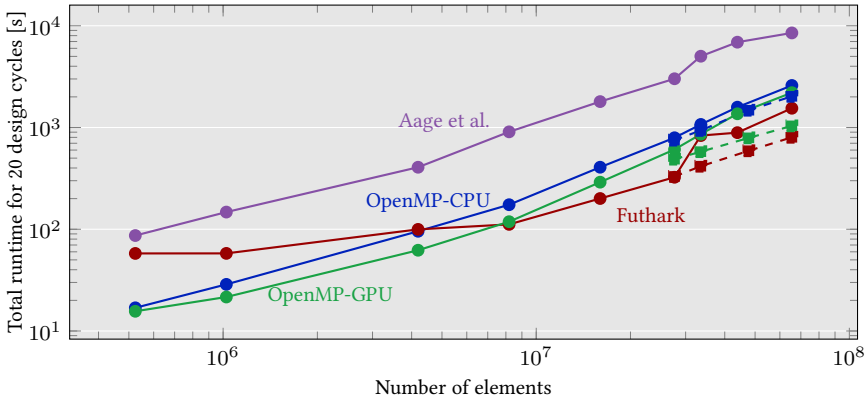


Figure 5.2: Wall-clock run-times for various topology optimization implementations and mesh refinements. All problems solve a cantilever problem, similar to that shown in fig. 5.1, with a filter radius to 5% of the domain depth. All testes are performed with 4 multigrid levels, with the exception of square nodes, which are computed using 5 multigrid levels. Figure taken from **[P4]**.

[13] performs slowest across all mesh sizes. This is in due to a lack of problem specific optimization, as this implementation uses PETSc [20], which is a general purpose distributed linear algebra library written using MPI. Furthermore, this implementation uses assembled matrices for the finest level, unlike all other implementations.

It can be sen that the Futhark based implementation is slow for small problem sizes compared to both OpenMP based implementations. This is likely due to some overhead present when launching the GPU computation kernels. Both GPU accelerated implementations are faster than the CPU reference for larger problem sizes, with the Futhark implementation being the fastest. However, it is also noteworthy that the Futhark implementation also consumes significantly more memory on the GPU, compared to the OpenMP-GPU implementation, limiting the largest problem solved by the Futhark implementation to 65M elements.

For large problems, it is found that adding a fifth multigrid level speeds-up the computation, due to a reduced computational effort in performing the coarse-grid correction, as indicated by the square nodes in fig. 5.2. The performance is increased significantly more for the GPU accelerated implementations, due to

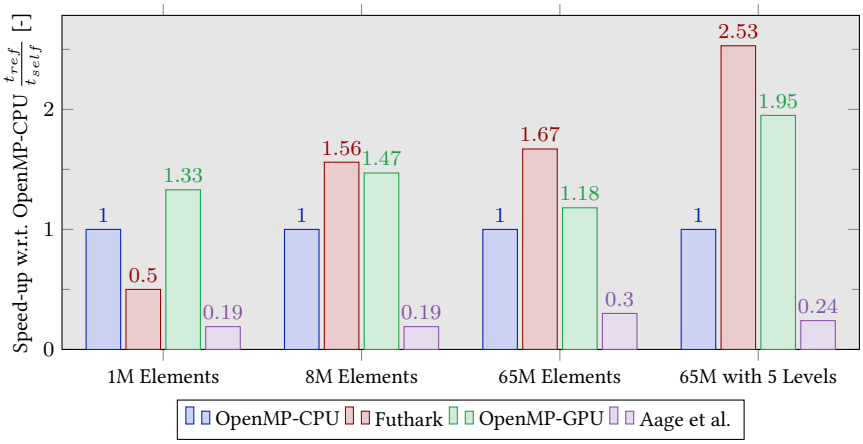


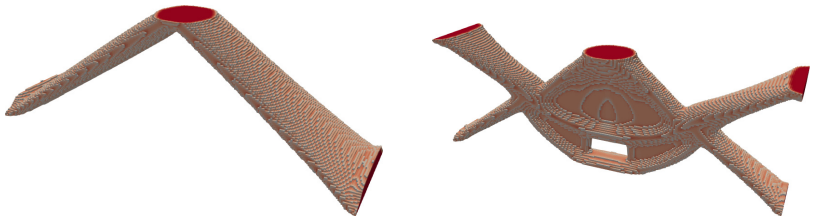
Figure 5.3: Relative speedup at selected refinements of fig. 5.1. All wall-clock times are normalized to the OpenMP-CPU implementation. Higher is better. Figure taken from **[P4]**.

the poor coarse space correction in the Futhark implementation, and reduced data transfer in the GPU accelerated OpenMP implementation.

Figure 5.3 shows the relative improvements in wall-clock time at select refinements from fig. 5.2. The used metric is improvement over the OpenMP CPU reference implementation, as this implementation is our best-case CPU performance. It can now be seen that the GPU accelerated OpenMP implementation always is slightly faster than the reference, until five levels are used for a large resolution. While a speedup of a factor of two might not seem impressive, remember that we compare towards an optimized reference, which is itself four times faster than the PETSc based implementation. The performance of the GPU accelerated OpenMP implementation with 5 levels and 65M elements, still corresponds to about 17 minutes for the first 20 design iterations in absolute terms.

Extension to Non-linear Elasticity

The Futhark based implementation was extended to nonlinear elasticity, in order to show that the language is sufficiently simple to allow extensibility. The approach from Buhl et al. [52] was used to perform optimization of end-



(a) Linear. (b) Nonlinear.

Figure 5.4: Double clamped beam example computed with 768 000 elements, a volume fraction of 0.1, a filter radius of 2.5 elements, and a load of 0.00015N. Figure taken from **[P4]**.

compliance. The elastic material was modeled using Green-Lagrange strains and a Neo-Hookean material model [52, 53].

It was found feasible to implement the non-linear approach in Futhark, although some key implementation details had to be reworked to accommodate the new problem. Most significantly, all local element matrices were no longer identical save a scaling, as they were now strain dependent. This meant that the nodal-stencil approach became less feasible, as computational effort was required to recompute all element matrices. Therefore a typical element based assembly was chosen, using the generalized histogram of Futhark [54] to implement the assembly operation itself. Figure 5.4 shows a double clamped cantilever optimized for both linear compliance and nonlinear elastic end-compliance. More structures, and a comparison of compliance measures, is given in **[P4]**.

Due to the additional need for element-wise integration of matrices, and the changed assembly routine, the nonlinear implementation is significantly slower than the linear counterpart. For instance, one application of the stiffness matrix takes $1.3\,\mathrm{ms}$ in the linear case and $1075\,\mathrm{ms}$ in the nonlinear case, using 2 million elements and the A100 GPU. This is indeed a limiting factor, and is why the nonlinear example uses less than a million elements. To implement a more efficient nonlinear approach, an implementation structured around the non-linearity should be devised. However, this implementation shows that the used approaches are extensible.

Concluding Remarks

Taking care to implement the topology optimization program in a way which utilizes available hardware efficiently is key for performing the computations in a reasonable amount of time. This is seen by comparing the presented reference OpenMP-CPU program with the PETSc based program from Aage et al. [13] shows a large improvement in efficiency on the same hardware. By using GPU acceleration, even through abstract offloading mechanisms, we see further improvement of performance using similarly priced hardware.

Chapter 6

Shape Optimisation of Shell Structures

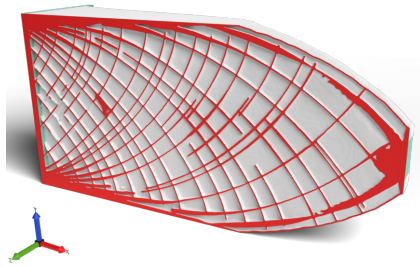


Figure 6.1: Example interior of dehomogenized structure. Image courtesy of Peter D. L. Jensen.

This chapter deals with the shape optimization of shell structures. On first glance, this subject should be outside the scope of a dissertation dealing with topology optimization. However, some motivation exist for this seeming deviation of topic.

6.1 Motivation for Shape Optimization

This dissertation is written as part of a larger research project. Here, several approaches were considered for efficient solution of very large-scale topology optimization problems. One of these approaches was the so-called dehomogenization. The method first solves a coarse topology optimization problem using some homogenized anisotropic microstructure, which is then interpreted to generate a finely resolved shell structure, such as the one shown in fig. 6.1. The choice of recreating shell structures was motivated by the known optimallity of shell structures in many situations [55]. This conversion from microstructure distribution to shell structure is a heuristic method, which does not consider mechanical performance. The conversion also breaks the underlying assumptions of the homogenization, such as separation of scales, which might also result in a changed mechanical performance.

This lack of control of mechanical performance in the final shell structure motivated the current chapter. By applying shape and thickness optimization to the resulting structures, as a final post-processing step, mechanical performance of the resulting structures would be ensured. Special modeling requirements arise from these dehomogenized structures, specially the ability to handle complex

input geometries with many shell intersections.

The optimization of shape and thickness of shell structures is interesting to consider as a contrast to large-scale topology optimization. By injecting prior knowledge of an initial structure and limiting the design-space to permutations of this initial structure, it is possible to reduce the computational complexity of the problem by orders of magnitude.

6.2 A Formulation for Shape Optimization

When performing shape optimization two fundamental approaches are usually considered. The first type of approaches rely on some abstract geometry representation, such as splines or other CAD parameters. These parameters are then used as design variables to ensure that the resulting design can be interpreted easily, and is sufficiently regular. Alternately, so-called "parameter-free" or nodal methods work with the nodal positions of the finite element mesh directly. Here the term nodal methods is preferred, as the methods use plenty of parameters which are not related to the CAD representation. It will be seen that these methods have similarities to topology optimization methods, due to the high number of design variables. For this work a nodal method is employed as we do not have access to any underlying CAD parameters.

This work uses a simple triangular shell element formulation, which trivially enables accurate modeling of shell intersections [56, 57]. The element is originally formulated for finite deformations, but is restricted to the linear infinitesimal deformations in this work.

Parameterizing and Regularizing Shape

The used nodal shape parametrization is based on nodal relocation, i.e. a change from an initial configuration to the current configuration. The unchanging initial nodal locations $\mathbf{x}_0$ are coupled to the current node locations $\mathbf{x}$ by some coordinate change $\Delta\mathbf{x}$. This is illustrated in fig. 6.2.

$$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x} \tag{6.1}$$

In principle, this change of coordinates $\Delta\mathbf{x}$ could be used directly as design variables. In practice, however, it is found to result in an optimization problem which is very hard to solve satisfactorily. We therefore introduce a regularization

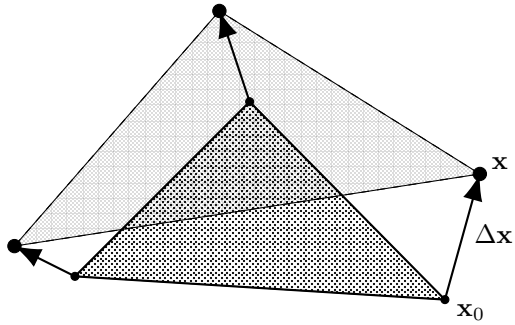


Figure 6.2: Sketch of updated nodal positions $\mathbf{x}$ defined as a relocation $\Delta\mathbf{x}$ from the initial mesh configuration $\mathbf{x}_0$. Figure from **[P5]**.

by filtering to this formulation. By solving a partial differential equations similar to the Helmholtz equation, on the initial mesh defined by $\mathbf{x}_0$, we smooth the changes in coordinates with neighboring nodes. When discretized with finite elements, this leads to the following expression.

$$\Delta\mathbf{x} = \left[r^2\mathbf{L} + \mathbf{M}\right]^{-1} \mathbf{M}\, \mathbf{x}_\mathrm{d} \tag{6.2}$$

Here, the matrices $\mathbf{L}$ and $\mathbf{M}$ denote the assembled finite element operators, r denotes a radius, which controls the smoothness of the resulting solution. A full explanation of this filtering technique can be found in **[P5]** or Asl and Bletzinger [36]. The unsmoothed coordinate relocations $\mathbf{x}_\mathrm{d}$ are used as design variables to control the shape of the shell structure.

This approach uses the initial mesh $\mathbf{x}_0$ as a constant reference, and defines shape as an relocation of this surface. This inherently puts the formulation at a disadvantage when considering arbitrarily large relocations, as $\mathbf{x}_0$ and $\mathbf{x}$ begin to differ significantly. However, when the resulting structures are "near" the initial structure, this approach was found to be effective.

However, filtering the shape changes is not sufficient to ensure regularity of the resulting finite element mesh with node coordinates $\mathbf{x}$. Elements can become distorted, resulting in unacceptably poor accuracy in the finite element approximation of the deformation. This will ultimately result in shapes which are optimized for spurious stiffness in the finite element modeling of malformed meshes.

To mitigate the potentially malformed meshes a second regularization is implemented as a constraint on a mesh quality measure. Here the so-called radius ratio of the triangular elements ρ is used as a quality metric.

$$\rho = \frac{R}{2r} = \frac{abc}{(b+c-a)(c+a-b)(a+b-c)} \tag{6.3}$$

The radius ratio ρ is computed from the three side-lengths of the triangle a, b, c. This metric for every triangle is aggregated into a single value ρ_{agg} using the p-norm, which can then be constrained.

$$\rho_{\text{agg}} = \left(\frac{1}{n_e} \sum_{i=1}^{n_e} {\rho_i}^p \right)^{\frac{1}{p}} \tag{6.4}$$

Optimisation Problem

Using the described parametrization of shape, along with a similar parametrization of thickness described in detail in **[P5]** we can now formulate the shape optimization problem.

$$\underset{\mathbf{x}_\text{d}, \mathbf{h}_\text{d}}{\text{minimize}} \qquad \mathcal{C} = \mathbf{u}^\top \mathbf{K}\mathbf{u} \tag{6.5a}$$

subject to

$$\begin{aligned}
&\text{state equation} && \mathbf{K}\mathbf{u} = \mathbf{f} && \text{(6.5b)}\\
&\text{filters} && \Delta\mathbf{x} = \mathbf{F}\mathbf{x}_\text{d} && \text{(6.5c)}\\
& && \mathbf{h}_\text{f} = \mathbf{F}^h \mathbf{h}_\text{d} && \text{(6.5d)}\\
&\text{volume} && v^* \leq V^* && \text{(6.5e)}\\
&\text{radius ratio} && \rho_{\text{agg}} \leq \rho_{\text{lim}} && \text{(6.5f)}\\
&\text{bounds} && \mathbf{x}_{\text{low}} \leq [\mathbf{x}_\text{d}]_i \leq \mathbf{x}_{\text{upp}} \; \forall i && \text{(6.5g)}\\
& && \mathbf{h}_{\text{low}} \leq [\mathbf{h}_\text{d}]_i \leq \mathbf{h}_{\text{upp}} \; \forall i && \text{(6.5h)}
\end{aligned}$$

The presented approach for shape parametrization can be extended to cover non-linear elasticity or alternate objective functions and constraints. In the considered formulation linear elastic compliance is minimized subject to nodal

relocations and nodal thicknesses and a constraint on the used volume as shown in eqs. (6.5a) and (6.5e). The remaining equations are only present to make explicit all relations and constraints which are introduced to regularize the problem or ensure correctness.

The resulting problem resembles a topology optimization problem with nodal variables. Some nodal field is regularized through filtering, and is used to define the resulting stiffness matrix of the system. However, due to the dimensionality of surfaces, we note that the resulting finite element meshes are much smaller than those required to solve similar topology optimization problems, resulting in smaller optimization problems in practice.

6.3 Examples

A few examples of shape and thickness optimization of shell structures are considered. We use one example to show the effect of the regularization parameters, and one example to consider the widely used cantilever example from topology optimization. More examples can be found in **[P5]**.

Michell Sphere

We consider a cylinder which is subjected to a torsional load, as shown in fig. 6.3. These boundary conditions are similar to those of the Michell sphere [58]. This example is also considered in a topology optimization problem by Sigmund et al. [55], which find that the solution is a variable thickness sphere, with high thickness near the clamped and loaded boundaries. In the context of shape optimization, the example helps illustrate the averse effect of regularization parameters, as they sometimes prohibit finding mechanically optimal solutions.

If this problem is solved for shape optimization with parameters of $\rho_{\text{lim}} = 1.1$ and $r_{\text{supp}} = 0.5$, which are consistent with suitable default parameters identified in **[P5]**, we get the solution shown in fig. 6.4a. This result is not consistent with the expected solution of a perfect sphere. Upon inspection, it is found that this structure reaches the limit of the quality constraint eq. (6.5f), where "inflating" the cylinder further would result in breaking the constraint.

Increasing the allowed average triangle distortion to $\rho_{\text{lim}} = 1.5$, the example is rerun resulting in the structure shown in fig. 6.4b. Now the structure is able to "inflate" to the size of the full sphere, attaining an aggregated radius ratio

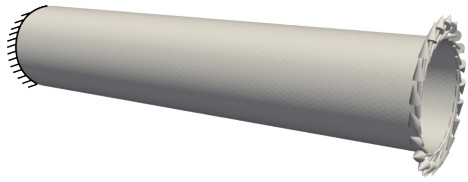
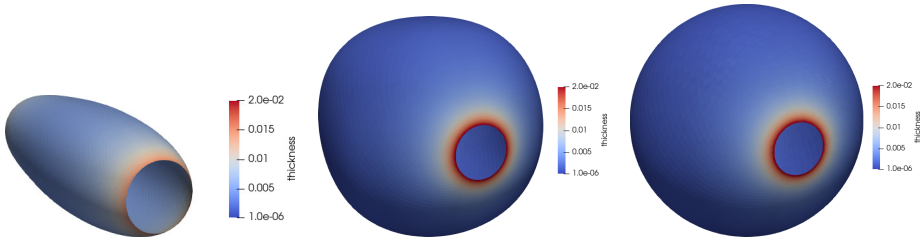


Figure 6.3: Boundary conditions and initial configuration of cylinder under torsion. Both lines with boundary conditions and forces are fixed to have no relocation during shape optimization through Dirichlet boundary conditions in the filter equations.



(a)
$\rho_{\text{lim}} = 1.1$,
$r_{\text{supp}} = 0.5$,
$\mathcal{C} = 0.74\,\text{J}$

(b)
$\rho_{\text{lim}} = 1.5$,
$r_{\text{supp}} = 0.5$,
$\mathcal{C} = 0.350\,\text{J}$

(c)
$\rho_{\text{lim}} = 1.5$,
$r_{\text{supp}} = 0.2$,
$\mathcal{C} = 0.347\,\text{J}$

Figure 6.4: Shape and thickness optimized cylinder under torsion, with various regularization parameters. Figures taken from **[P5]**.

higher than the original bound. However, the resulting structure is still not fully spherical, as some variable curvature is present. Upon inspection, it is found that the resulting structure is aligned with the used Cartesian coordinates. This could imply that the filtering is playing an effect, as every coordinate component is filtered independently.

Reducing the used filter size results in a structure which is perfectly spherical, as shown in fig. 6.4c. This result is now in very good agreement with the presented result from Sigmund et al. [55].

If this example is considered in isolation, it would seem to show that the applied regularization to the problem through filtering and radius-ratio constraints are too strict for many interesting applications. However, it was in general found throughout the examples of **[P5]** that the radius ratio constraint value of

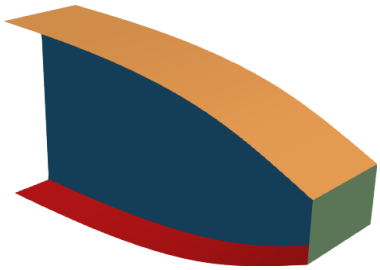


Figure 6.5: Initial configuration of cantilever example. The green surface is subjected to a traction force, and is kept in place during the optimization. Similarly, the opposite end of the structure is clamped and also restricted to remain on the initial plane.

$\rho_{\text{lim}} = 1.1$ gave rise to good results. Likewise, many applications found that increasing the filter radius consistently provided smoother and better performing structures.

Cantilever

A cantilever example was built based on the classical topology optimization structure. The initial structure is shown in fig. 6.5. The bound constraints for the unfiltered node locations eq. (6.5g) are used to make approximate box constraints to the final design corresponding to the usual $2 \times 1 \times 1$ domain usually used in the topology optimization problem. As these constraints are applied to the unfiltered, and not the filtered, nodal relocations, they are indeed only approximate.

Figure 6.6 shows the optimized cantilever structure. As the initial structure is vaguely modeled after a well-performing structure. It is seen that the resulting structure somewhat remains withing the bounding box, with a small error at the top and bottom. The flanges are bent slightly inward, and pushed further out from the initial configuration. The connection between flanges and loaded plate is realigned, such that the surfaces are near continuous. The thicknesses are redistributed, in a manner such that the central plate is similar to the well-studied "variable thickness plate" problem [4].

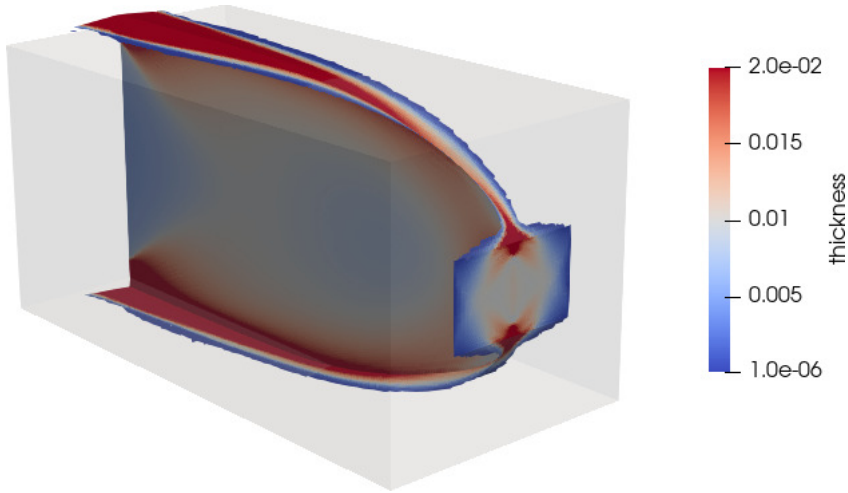


Figure 6.6: Resulting structure of the cantilever example. Elements with a resulting thickness lower than 1×10^{-5} are removed for clarity. The resulting compliance is $\mathcal{C} = 27.17\,\mathrm{J}$ ($\mathcal{C}_m = 27.02\,\mathrm{J}, \mathcal{C}_b = 0.15\,\mathrm{J}$).

Dehomogenized Cantilever

A single example is included of a dehomogenized cantilever structure, similar to that presented in the previous section. Note however, that the used volume and force magnitudes differs from the previous cantilever example, thus we cannot compare compliance values directly.

The tools which transform the dehomogenized structure to a shell based finite element mesh are currently under development. Therefore, several defects are present in the initial mesh, shown in fig. 6.7. The initial mesh is generated using a uniform thickness, as interpreting the thickness from the homogenization was not yet implemented. Some of the surfaces do not connect, as the interior surface stops before intersecting the outer hull. Similarly, some interior surfaces intersect the outer hull, but are not contained within it, resulting in exterior shell segments. The area where the load is applied is meshed irregularly, giving poor control of the actual area which is subject to traction loads. Most shell elements are near equilateral ($\rho \approx 1$), although a significant amount of outliers exist near surface intersections. Efforts to improve element quality near intersections has introduced angled elements, resulting in intersections which are no longer

Figure 6.7: Initial dehomogenzied cantilever beam. Meshed with 190.000 elements and uniform thickness of 0.01. Part of the structure has been made transparent to reveal the internal stiffeners. The initial compliance is $\mathcal{C} = 0.170\,\mathrm{J}$ ($\mathcal{C}_m = 0.163\,\mathrm{J}, \mathcal{C}_b = 0.007\,\mathrm{J}$), and the initial volume is $0.144\,\mathrm{m}^3$.

perfectly perpendicular.

Despite these issues the initial mesh shown in fig. 6.7 is used for an initial study of the effect of optimizing the shape and thickness of these shell structures. It is seen that the initial structure has an initial compliance of $0.170\,\mathrm{J}$ and volume of $0.144\,\mathrm{m}^3$. As the thickness information was not preserved, these numbers should not be expected to match their counterparts in the homogenized setting, although the uniform thickness was chosen to achieve something near the original target volume of $0.2\,\mathrm{m}^3$.

It is noteworthy that bending only composes 4% of the measured compliance in the initial structure. This domination of membrane loading shows a good correspondence with the tensile dominated result from homogenization. As thin shell structures have much better stiffness in the membrane direction, this indicates that the dehomogenized structure is performing quite well before optimizing the thickness and shape.

Optimizing shape and thickness results in the structure shown in fig. 6.8. Here it is clear that the thickness has been redistributed extensively. The inner stiffeners are reduced to a minimal thickness, indicating that they should potentially be removed altogether. This is somewhat to be expected, as the homogenized formulation used to generate this base structure included constraints to force shells to be active in all three direction, irrespective of physical

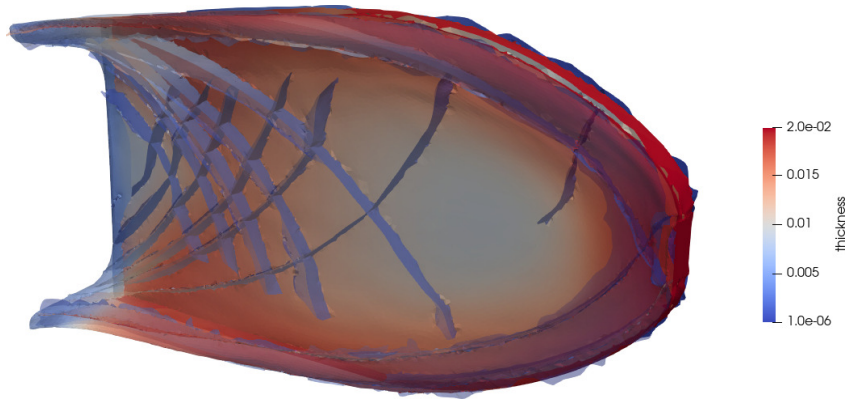


Figure 6.8: Result of shape and thickness optimization of dehomogenized cantilever. Part of the structure has been made transparent to reveal the internal stiffeners. The resulting compliance is $\mathcal{C} = 0.133\,\mathrm{J}$ ($\mathcal{C}_m = 0.131\,\mathrm{J}, \mathcal{C}_b = 0.002\,\mathrm{J}$), and the initial volume is $0.144\,\mathrm{m}^3$.

performance. The presented post-optimization includes no such constraints. The bending contribution towards the final compliance is reduced to 1.5%, possibly due to slight realignments of the shell and increasing thickness in areas with high membrane loading.

Concluding Remarks

This chapter has studied a case of restricting the design optimization problem to one of improving the shape of a known topology. We formulate the optimization problem, eq. (6.5), in a very similar way to a standard topology optimization problem, eq. (1.1), and solve it using the same nested approach. It is notable that suitable designs of shell structures can be found with a high degree of design freedom, within the given bounds, with significantly less computational effort compared to usual topology optimization. However, the resulting designs are only adaptations of the initial base structures, which in turn implies that the specification of the initial structure is paramount to obtain the best resulting structures. The shown preliminary result of optimizing a dehomogenized structure shows great promise of using optimization of shape and thickness as a post-processing tool for dehomogenized structures.

Chapter 7

Concluding Remarks

There is nothing so useless as doing efficiently that which should not be done at all.

Peter Drucker

Several implementation approaches to high-fidelity topology optimization have been studied. The methodologies can be divided into distributed computing discussed in chapters 3 and 4, and single desktop computing discussed in chapter 5. Both methods build upon the multigrid method, chapter 2, to efficiently solve the state equations of the topology optimization problems.

A key question which remains to be answered is what approach is best suited to implement computationally efficient, high fidelity, topology optimization. Chapter 5 shows that it is possible to solve finely resolved topology optimization problems using comparably accessible hardware in a matter of hours. Adopting this approach and extending it to efficiently handle arbitrary design domains and boundary conditions is, in my opinion, the most promising direction for efficient solutions of topology optimization problems with linear state equations. This is doubly true as the hardware cost of a single, albeit very expensive, desktop computer are far lower than those of accessing a high performance computing cluster. This seems to be a promising path to enable a wide range of users to access high fidelity topology optimization.

Nonlinear problems are not able to achieve the same computational improvement by using a Cartesian grid, as individual elements require numerical integration at every tangent evaluation regardless of the identical element shapes. This is apparent in the significant increase of computation time shown in chapter 5. The improved flexibility of unstructured meshes might prove beneficial as complex design domains can be represented directly through the mesh. The main computational bottleneck of integration can be mitigated by storing the matrix or stencil on all multigrid levels, but this comes with an increase in memory requirements of nearly a factor 100, potentially limiting the achievable resolution due to lack of memory. However, Cartesian grids could still offer some benefits. Rewriting the nonlinear finite element formulation to a Cartesian stencil, by could potentially greatly increase the implementation efficiency, and leverage linear implementations. In either case, efficiently implementing the element level integration will be central in achieving acceptable performance. The best approach for nonlinear problems, where local integration is unavoidable, remains an open question.

Restricting the optimization problem to improving an initial structure, as done in the shape optimization problem, presented in chapter 6, reduced the computational cost significantly. Here the full optimization problem was solved in ten minutes for most cases, with software which has not undergone significant

performance analysis or optimization. This is a reminder that solving a simplified optimization problem is much preferable, if feasible. In many industrial contexts, injecting prior knowledge is also a way to ensure sufficiently simple designs and comparability with manufacturing procedures.

Hardware for high performance computing is evolving to be more heterogeneous, and thus complex. Modern systems exhibit heterogeneity both by having various types of computing hardware, e.g. including GPU acceleration, and by introducing additional heterogeneity within every system component, e.g. by non-uniform memory access. Implementing efficient software for these systems is also becoming increasingly complex as the implementation needs reflect system behavior. It does well to remember that a majority of practitioners and researchers in structural optimization do not have profound knowledge of neither hardware architecture or software engineering. Many implementation details required to efficiently utilize modern hardware are distractions from the theory of structural optimization itself. As such, practitioners should be shielded from increasing complexity by some performance aware abstraction to enable users to utilize the performance improvements of modern hardware. More specialized compute acceleration platforms such as custom FPGA accelerators or quantum computing will presumably have similar requirements for abstraction and simplicity for wide adoption.

To summarize, for many topology optimization problems it is possible to use the linearity of the problem along with a Cartesian grid to create very efficient implementations. These can already be developed into powerful tools for large-scale topology optimization on single machines. Even if linearity is not available, it is conjectured that utilizing the structure of a Cartesian grid will prove to be a viable approach for large-scale applications. Shape optimization of dehomogenized structures has been shown to improve their shape and thickness distribution further, even though the base structures perform quite satisfactorily.

Bibliography

[1] Gilbert Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Wellesley, MA, fourth edition, 2009. ISBN 9780980232776.

[2] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition.* Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2003. ISBN 9780898715347.

[3] R.D. Cook. *Concepts and Applications of Finite Element Analysis, 4th ed.* Wiley India Pvt. Limited, 2007. ISBN 9788126513369.

[4] M.P. Bendsøe and O. Sigmund. *Topology Optimization: Theory, Methods, and Applications.* Springer Berlin Heidelberg, 2003. ISBN 9783662050866.

[5] Ole Sigmund and Kurt Maute. Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, 48(6): 1031–1055, Dec 2013. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-013-0978-6.

[6] Blaise Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158, Mar 2001. ISSN 0029-5981, 1097-0207. doi:10.1002/nme.116.

[7] Tyler E. Bruns and Daniel A. Tortorelli. Topology optimization of nonlinear elastic structures and compliant mechanisms. *Computer Methods in*

Applied Mechanics and Engineering, 190(26):3443–3459, 2001. ISSN 0045-7825. doi:10.1016/S0045-7825(00)00278-4.

[8] B. S. Lazarov and O. Sigmund. Filters in topology optimization based on helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering*, 86(6):765–781, May 2011. ISSN 00295981. doi:10.1002/nme.3072.

[9] Gore Lukas Bluhm, Ole Sigmund, and Konstantinos Poulios. Internal contact modeling for finite strain topology optimization. *Computational Mechanics*, 67(4):1099–1114, Apr 2021. ISSN 0178-7675, 1432-0924. doi:10.1007/s00466-021-01974-x.

[10] Ole Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4–5):401–424, Feb 2007. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-006-0087-x.

[11] Torsten Hoefler and Roberto Belli. Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 1–12, Austin Texas, Nov 2015. ACM. ISBN 978-1-4503-3723-6. doi:10.1145/2807591.2807644.

[12] Oded Amir, Niels Aage, and Boyan S Lazarov. On multigrid-cg for efficient topology optimization. *Structural and Multidisciplinary Optimization*, 49 (5):815–829, 2014. doi:10.1007/s00158-013-1015-5.

[13] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, 51 (3):565–572, 2015. doi:10.1007/s00158-014-1157-0.

[14] Niels Aage, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. Gigavoxel computational morphogenesis for structural design. *Nature*, 550 (7674):84–86, 2017. doi:10.1038/nature23911.

[15] Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. doi:10.1145/3272127.3275012.

[16] Darin Peetz and Ahmed Elbanna. On the use of multigrid preconditioners for topology optimization. *Structural and Multidisciplinary Optimization*, 63(2):835–853, 2021. doi:10.1007/s00158-020-02750-w.

[17] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial, Second Edition.* Society for Industrial and Applied Mathematics, second edition, 2000. doi:10.1137/1.9780898719505.

[18] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000. ISBN 9780127010700.

[19] Tam H Nguyen, Glaucio H Paulino, Junho Song, and Chau H Le. A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, 41(4):525–539, 2010. doi:10.1007/s00158-009-0443-8.

[20] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.18, Argonne National Laboratory, 2022.

[21] Jun Wu, Christian Dick, and Rudiger Westermann. A system for high-resolution topology optimization. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1195–1208, Mar 2016. ISSN 1077-2626. doi:10.1109/TVCG.2015.2502588.

[22] Niels Aage, Erik Andreassen, Boyan S. Lazarov, and Ole Sigmund. Giga-voxel computational morphogenesis for structural design. *Nature*, 550(7674): 84–86, Oct 2017. ISSN 0028-0836, 1476-4687. doi:10.1038/nature23911.

[23] Mads Baandrup, Ole Sigmund, Henrik Polk, and Niels Aage. Closing the gap towards super-long suspension bridges using computational morphogenesis. *Nature Communications*, 11(1):2735, Dec 2020. ISSN 2041-1723. doi:10.1038/s41467-020-16599-6.

[24] Carl Ollivier-Gooch. Coarsening unstructured meshes by edge contraction. *International Journal for Numerical Methods in Engineering*, 57(3):391–414, May 2003. ISSN 0029-5981, 1097-0207. doi:10.1002/nme.682.

[25] G.J. Gorman, J. Southern, P.E. Farrell, M.D. Piggott, G. Rokos, and P.H.J. Kelly. Hybrid openmp/mpi anisotropic mesh smoothing. *Procedia Computer Science*, 9:1513–1522, 2012. ISSN 18770509. doi:10.1016/j.procs.2012.04.166.

[26] Luca Cirrottola and Algiane Froehly. Parallel unstructured mesh adaptation using iterative remeshing and repartitioning. Research Report RR-9307, INRIA Bordeaux, équipe CARDAMOM, November 2019. URL `https://hal.inria.fr/hal-02386837`.

[27] Donald S. Shepard. A two-dimensional interpolation function for irregularly-spaced data. *Proceedings of the 1968 23rd ACM national conference*, 1968.

[28] Coreform LLC. Sculpt user manual, 2023. URL `https://coreform.com/manuals/latest/cubit_reference/meshing-schemes.html#%28part._sculpt%29`.

[29] Michael L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics, 2001. doi:10.1137/1.9780898718072.

[30] Ole Sigmund. *Design of material structures using topology optimization*. PhD thesis, Department of Solid Mechanics, Technical University of Denmark, 1994.

[31] Fengwen Wang, Boyan Stefanov Lazarov, and Ole Sigmund. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6):767–784, Jun 2011. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-010-0602-y.

[32] Boyan S. Lazarov, Fengwen Wang, and Ole Sigmund. Length scale and manufacturability in density-based topology optimization. *Archive of Applied Mechanics*, 86(1–2):189–218, Jan 2016. ISSN 0939-1533, 1432-0681. doi:10.1007/s00419-015-1106-4.

[33] Jun Wu, Niels Aage, Rudiger Westermann, and Ole Sigmund. Infill optimization for additive manufacturing—approaching bone-like porous structures. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1127–1140, Feb 2018. ISSN 1077-2626, 1941-0506, 2160-9306. doi:10.1109/TVCG.2017.2655523.

[34] Kai-Uwe Bletzinger. A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Structural and Multidisciplinary Optimization*, 49:873–895, Jun 2014. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-013-1031-5.

[35] Majid Hojjat, Electra Stavropoulou, and Kai-Uwe Bletzinger. The vertex morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268:494–513, Jan 2014. ISSN 00457825. doi:10.1016/j.cma.2013.10.015.

[36] Reza Najian Asl and Kai-Uwe Bletzinger. Implicit bulk-surface filtering method for node-based shape optimization and comparison of explicit and implicit filtering techniques, 2022. URL `https://arxiv.org/abs/2208.00700`.

[37] Matthijs Langelaar. Topology optimization for multi-axis machining. *Computer Methods in Applied Mechanics and Engineering*, page 27, 2019. doi:10.1016/j.cma.2019.03.037.

[38] Hak Yong Lee, Mu Zhu, and James K. Guest. Topology optimization considering multi-axis machining constraints using projection methods. *Computer Methods in Applied Mechanics and Engineering*, 390:114464, Feb 2022. ISSN 00457825. doi:10.1016/j.cma.2021.114464.

[39] Doe YOUNG Hur, Yuki Sato, Takayuki Yamada, Kazuhiro Izui, and Shinji Nishiwaki. Level-set based topology optimization considering milling directions via fictitious physical model. *Mechanical Engineering Journal*, 2020. ISSN 2187-9745. doi:10.1299/mej.20-00226.

[40] Joshua Gasick and Xiaoping Qian. Simultaneous topology and machine orientation optimization for multiaxis machining. *International Journal for Numerical Methods in Engineering*, 122(24):7504 – 7535, 2021. doi:10.1002/nme.6839.

[41] Kristian Ejlebjærg Jensen. Performing topology optimization with milling constraints, 2023. URL www.comsol.com/blogs/performing-topology-optimization-with-milling-constraints/.

[42] O. Sigmund. A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(2):120–127, Apr 2001. ISSN 1615-147X, 1615-1488. doi:10.1007/s001580050176.

[43] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. URL https://developer.nvidia.com/cuda-toolkit.

[44] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering*, 12(3):66–73, 2010. doi:10.1109/MCSE.2010.69.

[45] Rob Farber, editor. *Parallel Programming with OpenACC*. Morgan Kaufmann, Boston, 2017. ISBN 978-0-12-410397-9. doi:https://doi.org/10.1016/B978-0-12-410397-9.09988-1.

[46] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[47] Michel Steuwer, Toomas Remmelg, and Christophe Dubach. Lift: A functional data-parallel ir for high-performance gpu code generation. In *Proceedings of the 2017 International Symposium on Code Generation and Optimization*, CGO '17, page 74–85. IEEE Press, 2017. ISBN 9781509049318.

[48] Adam Paszke, Daniel Johnson, David Duvenaud, Dimitrios Vytiniotis, Alexey Radul, Matthew Johnson, Jonathan Ragan-Kelley, and Dougal Maclaurin. Getting to the point. index sets and parallelism-preserving autodiff for pointful array programming, 2021.

[49] Troels Henriksen, Niels G. W. Serup, Martin Elsman, Fritz Henglein, and Cosmin E. Oancea. Futhark: Purely functional gpu-programming with nested parallelism and in-place array updates. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 556–571, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4988-8. doi:10.1145/3062341.3062354. URL http://doi.acm.org/10.1145/3062341.3062354.

[50] Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics*, 37(6):1–14, Jan 2019. ISSN 0730-0301, 1557-7368. doi:10.1145/3272127.3275012.

[51] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3), oct 2008. ISSN 0098-3500. doi:10.1145/1391989.1391995.

[52] T. Buhl, C.B.W. Pedersen, and O. Sigmund. Stiffness design of geometrically nonlinear structures using topology optimization. *Structural and Multidisciplinary Optimization*, 19(2):93–104, Apr 2000. ISSN 1615-147X, 1615-1488. doi:10.1007/s001580050089.

[53] Anders Klarbring and Niclas Strömberg. Topology optimization of hyperelastic bodies including non-zero prescribed displacements. *Structural and Multidisciplinary Optimization*, 47(1):37–48, Jan 2013. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-012-0819-z.

[54] Troels Henriksen, Sune Hellfritzsch, Ponnuswamy Sadayappan, and Cosmin Oancea. Compiling generalized histograms for gpu. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20, pages 1–14. IEEE Press, 2020. ISBN 9781728199986. doi:10.1109/SC41405.2020.00101.

[55] Ole Sigmund, Niels Aage, and Erik Andreassen. On the (non-)optimality of michell structures. *Structural and Multidisciplinary Optimization*, 54(2): 361–373, Aug 2016. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-016-1420-7.

[56] L S D Morley. The constant-moment plate-bending element. *Journal of Strain Analysis*, 6(1):20–24, Jan 1971. ISSN 0022-4758. doi:10.1243/03093247V061020.

[57] F. Van Keulen and J. Booij. Refined consistent formulation of a curved triangular finite rotation shell element. *International Journal for Numerical Methods in Engineering*, 39(16):2803–2820, Aug 1996. ISSN 0029-5981, 1097-0207. doi:10.1002/(SICI)1097-0207(19960830)39:16<2803::AID-NME977>3.0.CO;2-2.

[58] 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*, 8(47):589–597, Nov 1904. ISSN 1941-5982, 1941-5990. doi:10.1080/14786440409463229.

[P1]

Topology Optimization of Ultra High Resolution Shell Structures.

E. A. Träff, O. Sigmund, and N. Aage.

Thin-Walled Structures 160 (March 2021) 107349.

doi:10.1016/j.tws.2020.107349.

[Published]

Thin–Walled Structures 160 (2021) 107349

Contents lists available at ScienceDirect

Thin-Walled Structures

journal homepage: www.elsevier.com/locate/tws

Full length article

Topology optimization of ultra high resolution shell structures

Erik A. Träff [a,*], Ole Sigmund [a], Niels Aage [a,b]

[a] Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Niels Koppel's Allé, Building 404, 2800, Lyngby, Denmark
[b] Centre for Acoustic-Mechanical Micro Systems, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

ARTICLE INFO

Keywords:
Topology optimization
Shell reinforcement
High-performance computing
Multigrid

ABSTRACT

This work presents a high performance computing framework for ultra large scale, shell-element based topology optimization. The shell elements are formulated using a linear elastic, small strain assumption and are of the solid type, meaning that each quadrilateral shell element is extruded and assigned 24 degrees of freedom. The resulting linear system is solved using a fully parallelized multigrid preconditioned Krylov method, tailored specifically for unstructured quadrilateral shell meshes. The multigrid approach is shown to have good parallel scaling properties and is able to efficiently handle the ill-conditioning arising from the 'Solid Interpolation of Material Properties' (SIMP) method. For the optimization, the classical minimum compliance design problem with multiple load cases, prescribed minimum length scale and a local volume constraint is investigated. The latter is implemented through efficient PDE-filtering in contrast to usual local image filtering based implementations. Finally, the framework is demonstrated on two idealized examples from civil and aerospace engineering, solving shell optimization problems with up to 11 million shell elements on 800 cores. As an example, this resolution corresponds to a minimum feature size of 1.5 cm on a high-riser of height 80 m.

1. Introduction

Shell structures can achieve high stiffness-to-weight ratios, and are therefore often found in weight critical applications including aircrafts, high-risers and ships. On the other hand, topology optimization is a numerical optimization tool used to create high performance structures, tailored to specific load cases, with little or no prior knowledge of the optimal structure [1]. Thus, topology optimization and shell structures provide a near-perfect combination in the pursuit for stiffness optimal and light-weight constructions.

Performing structural optimization using shell elements presents several possibilities and challenges. For example, the shell thickness can be optimized as a varying field throughout the structure [2]. Alternatively, applying the regular SIMP approach [3,4] can be used to determine the optimal perforation of a given shell structure. Another large class of shell topology optimization problems is the orientation of anisotropic material features, whether it is stiffening beads [5–7] or composite laminates [8]. Furthermore, applications of shell element based topology optimization include [9,10], which present simultaneous topology and shape optimization of curved plates, where the resulting structures carry the loads efficiently through in-plane strains. A discrete material optimization framework is presented in [11], which is specifically developed to choose between several laminate directions in shell structures. This framework is expanded several times to include optimizing the shell thickness [12], and in order to improve the discrete formulation [13]. A case study optimizing the thickness of a reinforcing shell layer of a submarine sail, which forms part of the outer submarine structure, is considered in [14]. Optimization of buckling phenomena of shells is studied in [15], where regularization schemes are included to avoid spurious buckling modes in void regions. A framework for the placement of reinforcing patches is presented in [16], where the patches act as a reinforcement onto an existing shell structure. A case study in which a wind turbine wing is optimized is presented in [17], using a genetic algorithm approach for the outer turbine skin, and topology optimization for the reinforcing spars.

Numerical analysis of shell structures using finite elements results in very high condition numbers of the resulting stiffness matrices compared to those of standard solid elements [18]. The high condition numbers have adverse effect on the numerical accuracy of direct solution methods, and have significant impact on the efficiency of iterative solution methods. The ill-conditioning occurs due to the large difference between high frequency in-plane deformation modes, and low frequency out-of-plane deformation modes, as shown in [18]. Usually, direct solution methods are applied to shell structures, even when tackling large scale problems as e.g. done in [19], which solves a finite element problem using approximately 920,000 quadrilateral shell elements. In [20] the authors present a multigrid method for shell structures, which shows that many iterations are needed for

* Corresponding author.
E-mail addresses: eratr@mek.dtu.dk (E.A. Träff), sigmund@dtu.dk (O. Sigmund), naage@mek.dtu.dk (N. Aage).

https://doi.org/10.1016/j.tws.2020.107349
Received 5 August 2020; Received in revised form 13 October 2020; Accepted 28 November 2020
Available online 9 December 2020
0263-8231/

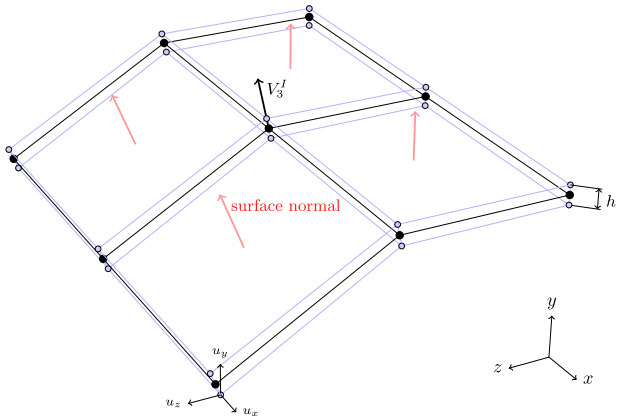
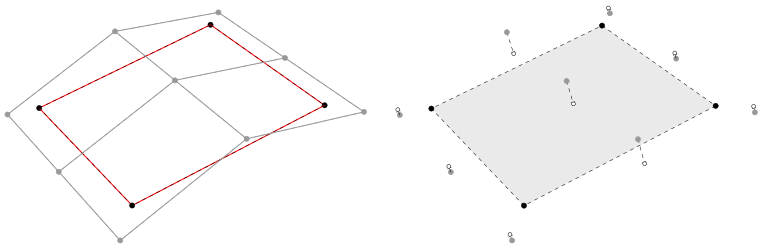


Fig. 1. Illustration of the extrusion process for interpreting quadrilateral shell elements as hexahedrals. The red arrows indicate the normals of the quadrilateral surfaces, while V_3 is used to denote the approximated shell normal at a given node. Each node in the quadrilateral mesh is extruded to two nodes in the corresponding hexahedral mesh, as shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Elements of fine and coarse grid.

(b) Projection of nodes from fine grid onto plane of coarse element.

Fig. 2. Illustration of the projection process used to evaluate if a node is within a quadrilateral.

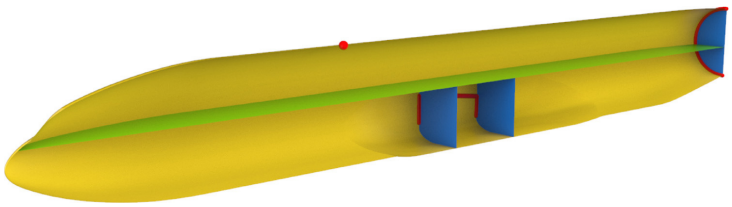


Fig. 3. Sketch of the used geometry for the fuselage example. The outer shell is shown in yellow, the floor plate in green, and the various reinforcing plates are shown in blue. The three regions with boundary conditions, are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

convergence in thin shells, where the conditioning problem is worst. Combining the high condition numbers of shell structures with the highly heterogeneous material parameters encountered in topology optimization, which further increases the condition number, presents a severe challenge that must be overcome in order to allow for ultra high resolution topology optimization of shells structures.

The conjecture that ultra high resolution is a necessity in order to achieve maximum insight into a given design space, has been demonstrated in many recent works including [21–26]. As an example, [22] discretizes an entire 26 m long aircraft wing, with a maximum element size of 0.8 cm, allowing the optimization to create many local features that would not have been possible using a coarse mesh. Moreover, the infill possibilities provided by additive manufacturing, require a highly resolved design space in order to allow for the formation of an intricate infill layout.

In this work, a framework for performing high resolution topology optimization of shell structures is presented. First, a fully parallelized multigrid preconditioner based on the approach from [22] is developed to facilitate the solving of very large scale shell topology optimization problems. The prolongation operators for the unstructured shell grids are obtained using a variation of the approach presented in [20] for a continuum shell element formulation. The proposed solver setup allows

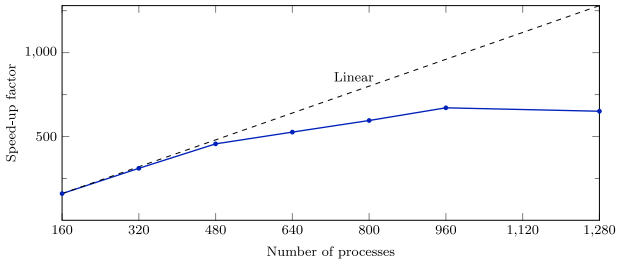


Fig. 4. Speed-up factor for the first design iteration of fuselage example with approximately 45 million degrees of freedom. The parallel version with 160 cores is used as the base case, due to memory constraints the problem is not solved on fewer compute nodes.

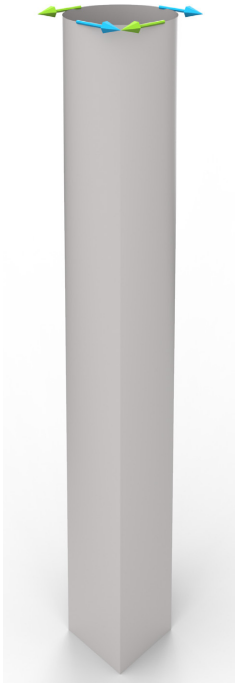


Fig. 5. Geometry of the Lotte tower test example. The square bottom of the tower is clamped. Two load cases, each consisting of two point loads, are depicted with green and blue arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

for the solution of systems with up to 11 million quadrilateral shell elements (69 million degrees of freedom) with highly heterogeneous material coefficients. The framework is then employed on selected design problems from civil and aerospace engineering and the findings are summarized and discussed.

2. Shell finite element model

Throughout this work linear elasticity using small strains for shells is considered. We present a methodology for using a continuum shell formulation defined for hexahedral elements, on a mesh of quadrilateral shell elements. This methodology is introduced, since continuum shell formulations have desirable properties with respect to symmetry and multigrid methods (see Section 3 for details). Furthermore, as most available meshing software focuses on quadrilateral shell elements, the development of this approach was necessary. The method works by creating an equivalent hexahedral element for each quadrilateral when integrating the element.

The methodology for creating an equivalent hexahedral requires access to the shell normals defined on the element nodes. In order to approximate these vectors the normal vector for each quadrilateral is computed. Then the nodal normal vector is found by averaging the normal vectors of all connected elements, as depicted in Fig. 1. When constructing the equivalent hexahedral, all nodes are translated by half the shell thickness h in both directions along the corresponding nodal normal, as also depicted in Fig. 1. During the assembly process the nodes in the quadrilateral mesh can be considered as owning the degrees of freedom corresponding to both extruded nodes in the equivalent hexahedral mesh.

A drawback of this extrusion method, is that arbitrary shell thicknesses cannot be considered. The formation of hourglass hexahedral elements is possible if the shell thickness is large relative to the element side length. Therefore careful monitoring of the validity of the extruded mesh is necessary. In this work the scaled Jacobian metric is computed for all elements, and the computations are stopped if minimum value of the metric is unsatisfactory, which for this work is chosen as less than 0.2.

The shell element is a slightly modified version of the continuum shell element presented in [27]. The used formulation employs both the Mixed Interpolation of Tensorial Components (MITC) correction when interpolating the out-of-plane shear strains and the Assumed Natural Strains (ANS) correction when computing the normal strain in the out-of-plane normal direction. Unlike the original formulation from [27], this version does not employ the Enhanced Assumed Strains (EAS) correction, as this correction negatively affects the efficiency of iterative solvers. As in the original formulation, it is possible to include multiple layers with varying material properties, which is used to implement passive domains during the optimization when formulating a shell reinforcement problem. Finally, the Scaled Thickness Conditioning (STC) [28] is applied to the formulation, in order to reduce the condition number associated with thin shell elements. For completeness, the full shell element formulation including design dependence is given in Appendix A.

The accuracy of the element implementation was verified using a set of standard test examples. For the centrally loaded circular clamped plate and for the hemispherical problem presented in [29], the element is found to converge to the exact deformation value. For the pinched diaphragm supported cylinder study [30], the element implementation was found to deviate by 5% of the analytical displacement value.

Table 1
Summary of the used smoothers, preconditioners and steps used in the multigrid preconditioner.

Level(s)	Smoother	Preconditioner	Steps
Finest	Flexible CG	Multigrid	–
Intermediate	Chebyshev [34]	SOR [34]	4
Coarse	GMRES [34]	Algebraic multigrid (PETSc GAMG) [35]	200

3. Multigrid approach

It is well-known that solving large scale problems using direct methods is not feasible in general. Therefore, iterative methods must be employed. Here we use a Galerkin projection based multigrid preconditioned Krylov subspace solver in order to efficiently solve large scale heterogeneous shell problems [31].

The used multigrid method is based on the scheme presented in [22,32], which uses a standard V-cycle multigrid method with Galerkin projection as preconditioner for the preconditioned flexible conjugate gradients solver [33,34]. For topology optimization problems the solver uses the deformation of the previous design as initial guess for the CG iterations, allowing for fast convergence for smaller design changes. Information about the used hierarchy of smoothers is summarized in Table 1.

The prolongation operator is based on [20] which deals with degenerated, i.e. quadrilateral, shell elements. The prolongation operator maps the displacement fields from a coarse mesh onto a fine mesh. In the presented approach the displacement fields of the interior and exterior nodes of the hexahedrals are treated separately, resulting in a prolongation for quadrilateral shell elements which prolongs six different fields. This treatment of the displacements as two separate fields is performed to avoid smoothing of the field through the shell thickness, as there exists no additional nodes through the thickness in the coarser meshes.

The operator is constructed in parallel using the shape functions of the coarse mesh to interpolate coarse nodal values to a given node in the fine mesh. In practice this requires some additional computations, due to both the lack of structure in the mesh and due to the 2.5D nature of shells. The pseudo-code for computing the prolongation operator between two meshes is shown in algorithm 1. The KD-tree used in algorithm 1 is introduced to avoid performing a search for all potential nodes in the unstructured meshes. Similarly, the second loop, which handles all fine nodes, is introduced to implement a heuristic addition for nodes which were never found to reside within a coarse element. The process of projecting a node onto the element plane, and determining if it is within the said element, is depicted in Fig. 2. The potential nodes are identified, and projected onto the plane of the quadrilateral, wherein it is checked whether they are inside the quadrilateral.

Algorithm 1: Pseudo-code to construct prolongation

Data: Fine mesh $\mathcal{H}^i$, coarse mesh $\mathcal{H}^{i-1}$
Result: prolongation operator P
K := KD tree of all nodes in $\mathcal{H}^i$;
n_{touched} := zero valued array of size nodes in $\mathcal{H}^i$;
forall the *elements* $e \in \mathcal{H}^{i-1}$ **do**
  i_e := global indices associated with e;
  c_e := Center of e;
  l_e := approximate element size of e;
  n_{found} := nodes in K within $1.5 \times l_e$ of c_e;
  forall the *nodes* $n \in n_{found}$ **do**
    $\tilde{n}$:= projection of n onto plane of e;
    if $\tilde{n}$ *is within* e **then**
      $n_{\text{touched}}[n] := n_{\text{touched}}[n] + 1$;
      i_n := global indices associated with n;
      N := shape functions in e corresponding to position of $\tilde{n}$;
      Insert N into the submatrix $i_n \times i_e$ of P;
    end
  end
end
Synchronize values of n_{touched} across processors;
K_c := KD tree of all nodes in $\mathcal{H}^{i-1}$;
forall the *nodes* $n \in \mathcal{H}^i$ **do**
  if $n_{touched}[n]$ *is* 0 **then**
    n_{found} := 3 nearest nodes to n in K_c;
    $w_{\text{found}} := \frac{1}{|n_{\text{found}} - n|^2}$;
    $w^-_{\text{found}} := \frac{w_{\text{found}}}{\sum w_{\text{found}}}$;
    Insert w^-_{found} into P such that n is coupled to n_{found};
  end
end

3.1. Implementation

The multigrid approach is implemented using the PCMG preconditioner in PETSc [36–38], which also contains the multigrid cycles and smoothers. The prolongation operator is constructed in parallel as a preprocessing step. The efficiency of the multigrid approach and element is confirmed by the strong scaling results presented in Fig. 4, where the fuselage example from Fig. 3 (and Section 5.2) is studied with approximately 45 million degrees of freedom. A near linear speedup is observed until 480 processors, or 93 000 degrees of freedom pr. core.

The Lotte tower design problem from Fig. 5 [39] is used to examine the residual as function of conjugate gradient iterations, see more details in Section 5.1. Fig. 6 shows the convergence of the multigrid approach for various multigrid levels using homogeneous material parameters in the domain. As can be seen, the required number of iterations to reach a given tolerance increases with the number of multigrid levels. However, the cost of each iteration decreases with more levels, as the coarse problem size decreases.

A study of the residual decrease for the same domain with various shell thicknesses is shown in Fig. 7. The number of iterations required to reach convergence increases drastically as the thickness decreases relative to the size of the shell. This is due to the increasing condition number, as the difference between the largest and smallest eigenvalues increases. The increasing condition number drastically reduces the convergence rate of Krylov methods.

Fig. 8 shows the required number of CG iterations for convergence during design iterations for a typical topology optimization problem. It can be seen that the required number of iterations increases in the beginning, as the heterogeneity of the material parameters increases fast. After some iterations, however, the design changes become smaller and more localized, allowing the CG iterations to benefit from the non-zero initial guess from the previous design iteration.

The presented multigrid approach, and associated implementation, is not limited to shell elements. Similar performance as that presented in Fig. 4 has been observed when using the framework to solve topology optimization problems using hexahedral elements, although far fewer iterations are needed to reach convergence in the iterative solving process. The scaling tests, along with the numerical results of Section 5.2, were run on the DTU Sophia cluster with two AMD EPYC 7351 16-Core processors and 128 GB memory per node and infiniband interconnect.

4. Optimization formulation

The considered optimization problem is the well-studied minimum compliance problem for linear elasticity with multiple load cases [1]. Each element in the mesh is assigned a design variable $x_e \in [0, 1]$. The

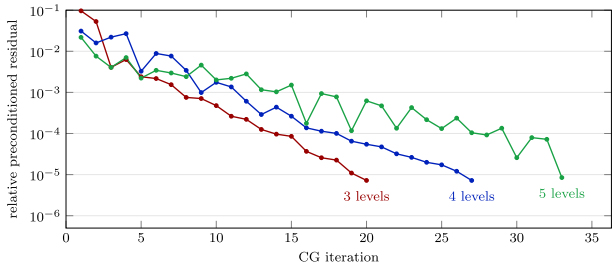


Fig. 6. Residuals for the Lotte tower example (Section 5.1) with various multigrid levels. Meshed using 126,024 elements (757,560 dof) and a shell thickness of 0.1.

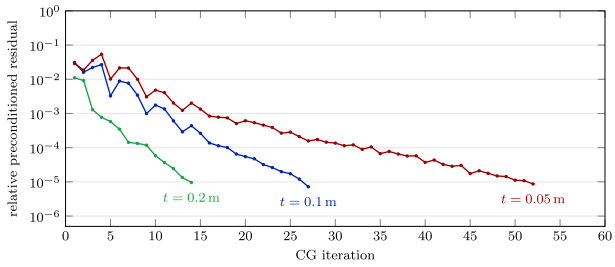


Fig. 7. Residuals for the Lotte tower example (Section 5.1) with 4 multigrid levels, and various shell thicknesses. Meshed using 126,024 elements (757,560 dof).

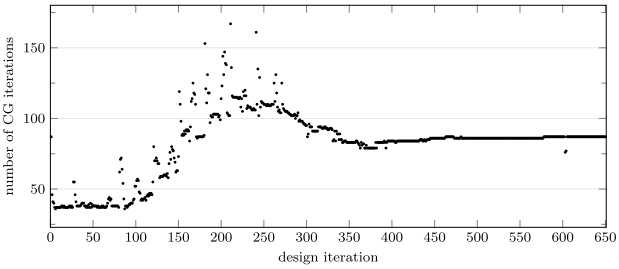


Fig. 8. Used number of CG iterations for all design iterations during the optimization of the Lotte tower example (Section 5.1). The model is meshed using approximately 11 million elements, with a shell thickness of 0.015 m.

field of design variables is modified through a set of filters in order to obtain the so-called physical density, which is used to interpolate the stiffness and mass of the corresponding element. The robust formulation [40,41] is applied to ensure both a minimum length scale and a 0–1 final design. The robust formulation removes the need for the usual penalization of intermediate densities, as discussed in [42]. A linear Young's modulus interpolation scheme is used, corresponding to the 'Simplified Isotropic Material with Penalization' (SIMP) method with penalization value $p = 1$, as shown in Eq. (1).

$$E(x_e) = E_{\min} + x_e(E_0 - E_{\min}), \quad E_{\min} = 10^{-6}E_0, \quad 0 \le x_e \le 1 \tag{1}$$

E_0 denotes the background stiffness and $E_{\min}$ denotes the stiffness of a weak material used to imitate void.

To prevent numerical artifacts such as checkerboards and mesh dependency we add regularizations in the form of the Helmholtz PDE-filter [43], and the robust formulation [41]. Furthermore, a modified local volume constraint [23] is developed for the problem, which uses the Helmholtz PDE-filter instead of a local average to calculate the local volume fraction.

In order to simplify the notation the PDE filtering operator $F_r : \mathbb{R}^n \to \mathbb{R}^n$ for a given radius r is introduced. Here n denotes the number of elements in the finite element mesh. The operator is defined as $\mathbf{y} = F_r(\mathbf{x})$ where y is the solution to the modified Helmholtz PDE with homogeneous Neumann boundary conditions defined on the same mesh. Two realizations of this operator are used when formulating the optimization problem; the solid filtering, which replaces the density filter F_{solid} with $r = r_{\text{solid}}$, and the local volume filter, which replaces the neighborhood average [23] for the local volume filter F_{LV} with $r = r_{\text{LV}}$. Usually the filter radii are chosen such that $r_{\text{LV}} \gg r_{\text{solid}}$. The authors note that all of the presented radii are corrected to obtain the scalar coefficient $r^* = \frac{r}{2\sqrt{3}}$ used in the PDE formulation, as discussed in [42].

As with the PDE filter, the heaviside projection is introduced using an operator $H_\eta : \mathbb{R}^n \to \mathbb{R}^n$ which depends on two parameters η and β. The operator is defined for a given threshold value η, while the value of β is increased throughout the optimization using a continuation scheme. The operator is defined such that $\mathbf{z} = H_\eta(\mathbf{y})$ is applied

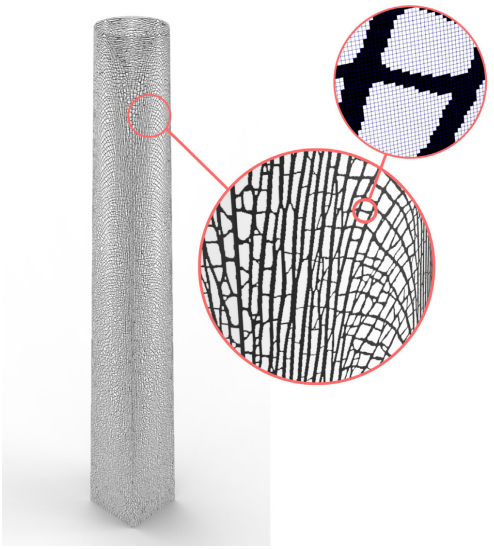


Fig. 9. Close up of resulting topology for the Lotte Tower example using 11.6 million shell elements and filter radii of $r_{\text{solid}} = 8\,\text{cm}$ and $r_{\text{LV}} = 96\,\text{cm}$.

element-wise in the vectors as

$$z_e = \frac{\tanh\beta\eta + \tanh\beta(y_e - \eta)}{\tanh\beta\eta + \tanh\beta(1-\eta)}, \quad \forall e \in \{1,\dots,n_e\}. \tag{2}$$

Three realizations of the Heaviside operator are used in the robust formulation; the nominal H_{η^n}, dilated H_{η^d}, and eroded H_{η^e}. In this work, the used threshold values are $\eta^n = 0.5$, $\eta^d = 0.4$ and $\eta^e = 0.6$. The continuation scheme of β begins by setting $\beta := 0.01$, then after every 30 iterations the value is updated $\beta := \beta + 1$. When $\beta = 8$, or at iteration 240, the update scheme changes to $\beta := \frac{6}{5}\beta$ every 30 iterations. This continues until iteration 600, where the value of beta $\beta \approx 59.4$. The optimization is run for a maximum of 650 iterations. This conservative choice of β-continuation is chosen due to the slower increase in value, compared to the usual continuation scheme used in e.g. [41]. The slower increase is not strictly necessary, but results in smaller jumps in β value, which is found to improve convergence stability.

Given a set of n_l load cases with their respective force vectors $\mathbf{f}_i \in \mathbb{R}^n, i \in \{1,\dots,n_l\}$, and corresponding weights $w_i \in \mathbb{R}$. The optimization formulation can then be written as:

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathbb{R}^{n_e}}{\text{minimize}} && \sum_{i=1}^{n_l} w_i \mathbf{u_i}^\top \mathbf{f}_i \\
&\text{subject to:} \\
&\text{state equation} && \mathbf{K}(H_{\eta^e}(F_{\text{solid}}(\mathbf{x}))\mathbf{u_i} = \mathbf{f}_i, \quad \forall i \in \{1,\dots,n_l\} \\
&\text{global volume} && \frac{1}{\sum_{e=1}^{n_e} v_e} \sum_{e=1}^{n_e} v_e (H_{\eta^d}(F_{\text{solid}}(\mathbf{x})))_e \le V_g^* \\
&\text{local volume} && \|F_{\text{LV}}(H_{\eta^d}(F_{\text{solid}}(\mathbf{x})))\|_{p_{\text{lv}}} \le V_l^*
\end{aligned} \tag{3}$$

Note that this is the reduced version of the robust formulation, which relies on knowledge about the compliance and both constraints. Namely, it is known that the compliance attains its maximum value for the eroded realization, while both volume constraints attain their maximal values for the dilated realization.

The local volume constraint is aggregated using the p-norm approximation, with a penalty value of $p_{\text{lv}} = 16$. All examples presented here are conducted with the global and local volume fractions $V_g^* = V_l^* = 0.5$. It is noted that applying the volume constraint on the dilated field does not directly control the resulting volume in the nominal case. As all examples presented in this paper are purely academic, with arbitrarily chosen volume constraints, a variation in resulting volume fractions is accepted, and the resulting nominal volume fractions are stated. If control of the nominal volume is desired, the volume fraction update scheme presented in [41] may be used. Furthermore, the exact control of the volume fraction is complicated by the approximate nature of the p-norm aggregation used for the local volume constraint. The approximation error in the local volume constraint might prevent

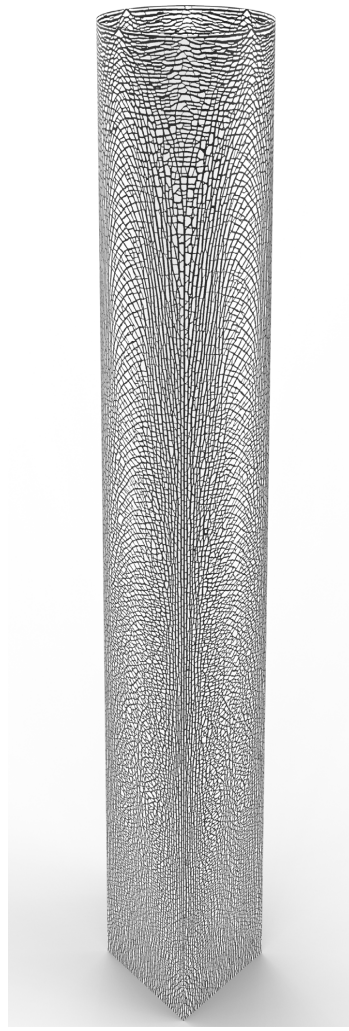


Fig. 10. Resulting topology of the Lotte tower example with filter radii of $r_{\text{solid}} = 8\,\text{cm}$ and $r_{\text{LV}} = 96\,\text{cm}$.

Fig. 11. Resulting topology of the Lotte tower example with filter radii of $r_{\text{solid}} = 22.5\,\text{cm}$ and $r_{\text{LV}} = 4.5\,\text{m}$.

the optimizer from using more material, even if the global volume constraint allows for more material.

Finally, the sensitivities are obtained using the discrete adjoint method and the optimization problem is solved using a fully parallelized implementation of the Method of Moving Asymptotes (MMA) [44,45].

5. Numerical examples

5.1. Lotte tower - perforation design

The first example of high resolution topology optimization using shell elements is the Lotte Tower example, originally suggested in [39]. The tower has a square base of size 10 m by 10 m meters which is clamped, and a circular top with radius 5 m where the loads are applied. The tower is 80 m high, and is subjected to two sets of point-loads of magnitude 1 N at the circular top, as depicted in Fig. 5. The tower itself is modeled as an empty shell where the cross-section linearly varies between the square at the bottom and the circle at the top. The geometry was meshed using Cubit [46] with a mapping algorithm, resulting in 11,637,184 elements on the finest mesh, corresponding to an average element size of 1.6 cm. The coarser meshes were meshed independently using the same approach but using fewer elements. This mesh refinement is so highly resolved that an illustration of the used mesh is difficult to render. Instead, the refinement can be inferred from the close-up of the results presented in Fig. 9.

The tower is optimized using a filter radius of $r_{\text{solid}} = 8\,\text{cm}$ or five times average element size. Using the expression presented in [47] it can be found that this corresponds to imposing a length-scale of 3.2 cm or 2 times the average element size. The local volume constraint is computed based on the dilated realization, with a filter radius of $r_{\text{LV}} = 96\,\text{cm}$. The studied problem is that of optimal perforation of a shell, i.e. no passive shell layers have been included.

A close-up of the resulting structure is shown in Fig. 9 in order to illustrate the refinement of the structure. It can be seen that the small bone-like features are resolved with multiple elements across their thickness.

An overview of the resulting structure is shown in Fig. 10. It can be seen that it is very intricate, consisting of many small bone-like features which are oriented to carry the loads. Arches are formed near the points where the loads are applied and underneath. Near the corners of the square base many vertical substructures can be found, which branch out to the arches throughout the height of the structure. Note that the entire structure is modeled without any symmetry in the design variables. Nevertheless, the resulting designs are near symmetric, although not fully. This is due to numerical noise and non-convexity of the optimization problem.

An alternate version of the tower has been run using the same parameters, with the exception of the filter radii which are changed to $r_{\text{solid}} = 22.5\,\text{cm}$ (15 times the average element size) for the solid filter, and $r_{\text{LV}} = 4.5\,\text{m}$ for the local volume filter. The resulting structure is shown in Fig. 11, which reveals many of the same structural features, albeit with a larger minimum member size, than the example using a lower filter radius. The non-symmetry of the solution is much more apparent in this example due to the large structural features. It can be seen that the non-symmetry is particularly concentrated on the middle of the lower flat sections, and between the loaded arches near the top of the tower. This coincides with the areas with lower strain density, where material is applied late in the optimization process, and only has little effect on minimizing the objective function.

The resulting compliance values are 0.334 J for the $r_{\text{solid}} = 8\,\text{cm}, r_{\text{LV}} = 96\,\text{cm}$ filter radii, and 0.316 J for the $r_{\text{solid}} = 22.5\,\text{cm}, r_{\text{LV}} = 4.5\,\text{m}$ filter radii. As the filter radius r_{solid} is increased, the compliance value should increase, as a larger feature size is enforced on the optimization algorithm. Oppositely, when the local volume filter radius r_{LV} is increased, the compliance values should decrease, as less complexity is forced on the resulting structure. Thus when increasing both radii, it is difficult to predict the effect on the compliance value, which in this case is lower.

The final volume fractions for the two designs were 0.46 for the $r_{\text{solid}} = 8\,\text{cm}, r_{\text{LV}} = 96\,\text{cm}$ filter radii, and 0.45 for the $r_{\text{solid}} = 22.5\,\text{cm}, r_{\text{LV}} = 4.5\,\text{m}$ filter radii. These values are lower than the maximally allowed value of 0.5 for the reasons discussed in Section 4.

Fig. 12 shows the iteration history of the weighted compliance for the Lotte tower example with filter radii $r_{\text{solid}} = 8\,\text{cm}$ and $r_{\text{LV}} = 96\,\text{cm}$ from Figs. 9 and 10. It can be seen that the compliance steadily decreases, with the exception of the discontinuities which occur when the β value is increased through the continuation scheme. As expected, the final compliance is higher than the value found for the homogeneous design in the first iteration. This is a consequence of the linear stiffness

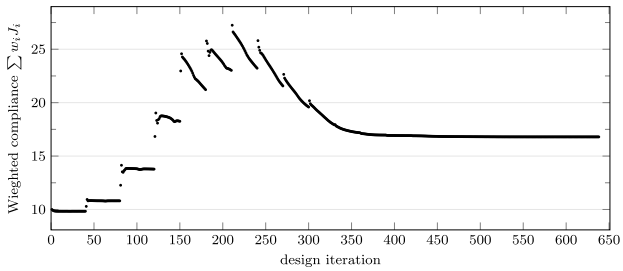


Fig. 12. Weighed compliance over design iterations for the Lotte tower example with filter radius of $r_{\text{solid}} = 8\,\text{cm}$ and local volume filter radius $r_{\text{LV}} = 96\,\text{cm}$. The compliance weights are chosen as $w_1 = w_2 = 1.9898 \times 10^{-2}$, such that $w_1 J_1 = w_2 J_2 = 5$ for the initial conditions.

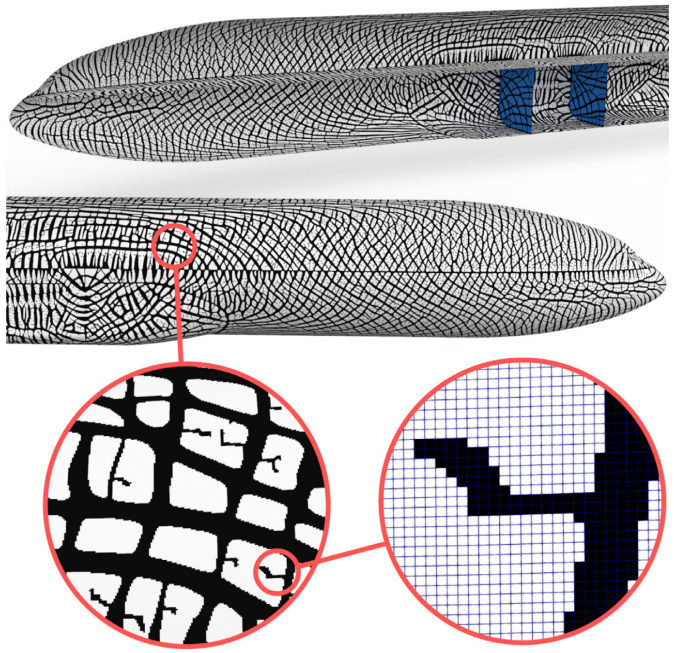


Fig. 13. Small local filter size - Overview of the resulting topology of the fuselage example. A white plate has been added to the components which have been modeled using a passive shell. Likewise, a blue background plate is added to the reinforcing plates near the wing, in order to improve the visualization. The example uses a filter radius of $r_{\text{solid}} = 7.5\,\text{cm}$ (7.9 elements) and local volume filter radius $r_{\text{LV}} = 127\,\text{cm}$ (130 elements). The radii of the two enlarged circles are 127 cm and 15 cm, respectively. The elements are shown in the innermost enlargement, in order to illustrate the mesh refinement.

interpolation, which makes the compliance of the homogeneous design lower than what is usually obtained using standard SIMP with $p = 3$.

Both of the tower examples ran for 650 design iterations on 25 compute nodes on the DTU Sophia cluster, i.e. a total of 800 cores. The full optimization procedure takes around 17 h, or an average of 94 s pr. design iteration, and solves the state field a total of 1300 times, due to the two load cases.

5.2. Fuselage - reinforcement design

The second example concerns the optimal reinforcement of the fuselage of the NASA common research model [48] depicted in Fig. 3. The geometry of the common research model is used to define the outer shell of the fuselage. Additionally, a floor panel and some vertical stiffeners are added. The mesh is generated using a paving algorithm implemented in Cubit [46], and consists of 7,488,576 shell elements, with an average element size of 0.95 cm. The coarse grids for the multi-grid prolongation are generated independently in a similar fashion, targeting larger average element sizes. Like the Lotte tower, the mesh refinement is so highly resolved that an illustration is difficult to render. However, the elements in a close-up are shown in Fig. 13 to give an impression of the mesh. The filter size for the solid filter for the robust formulation is $r_{\text{solid}} = 7.5\,\text{cm}$ or 7.9 times the average element size, corresponding to imposing a length-scale of 3 cm or 3 times the average element size.

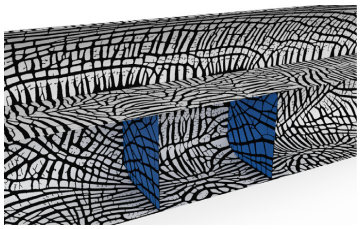


Fig. 14. Small local filter size - Close up of the central part of the fuselage. Computed with a filter radius of r_{solid} = 7.5 cm (7.9 elements) and local volume filter radius r_{LV} = 127 cm (130 elements).

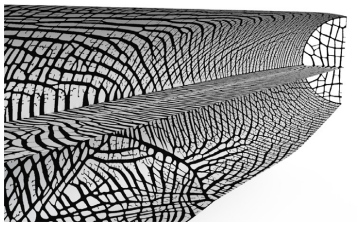


Fig. 15. Small local filter size - Close up of the tail section of the fuselage. The reinforcement structure of the floor can also be seen. Computed with a filter radius of r_{solid} = 7.5 cm (7.9 elements) and local volume filter radius r_{LV} = 127 cm (130 elements).

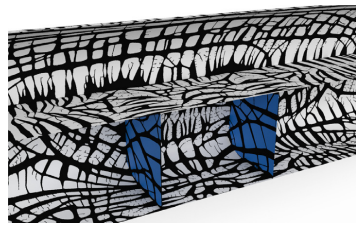


Fig. 17. Large local filter size - Close up of the central part of the fuselage. Computed with a filter radius of r_{solid} = 7.5 cm (7.9 elements) and local volume filter radius r_{LV} = 300 cm (260 elements).

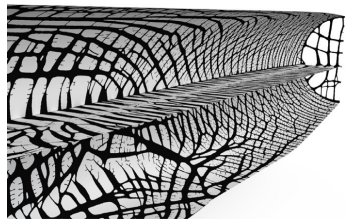


Fig. 18. Large local filter size - Close up of the tail section of the fuselage. The reinforcement structure of the floor can also be seen. Computed with a filter radius of r_{solid} = 7.5 cm (7.9 elements) and local volume filter radius r_{LV} = 300 cm (260 elements).

The outer shell, shown in yellow in Fig. 3 and floor panel shown in green, are both modeled as shells reinforced symmetrically from both sides. In both cases the central 0.13 cm is modeled as a passive domain, while two outer reinforcements, each with thickness 1.2 cm, are modeled using the SIMP approach with a single design variable. This corresponds to the innermost 5% of the shell being passive. The plate structures shown in blue in Fig. 3 are modeled without any passive domain and a thickness of 2.54 cm, which corresponds to a shell perforation design problem.

Three variations of the fuselage example are considered. The two first variations vary the local volume filter size, which is used for the local volume constraint. The fuselage is studied using a local volume filter size of r_{LV} = 127 cm and r_{LV} = 300 cm, which corresponds to 130 and 260 times the average element size respectively. The final variation studies the effects of including a thicker passive shell in the outer skin and on the floor plate. Here, a passive shell of 40%, instead of the usual 5%, is studied. The variation in passive thickness is performed using a local volume filter radius of r_{LV} = 127 cm, corresponding the first variation. The three cases are denoted as 'small local filter size', 'large local filter size', and 'thick passive shell' respectively, to help distinguish the designs.

A symmetry boundary condition is applied on the mid-plane of the fuselage, such that only half the fuselage is modeled. A Dirichlet boundary condition with zero displacement in the 'upwards' direction is

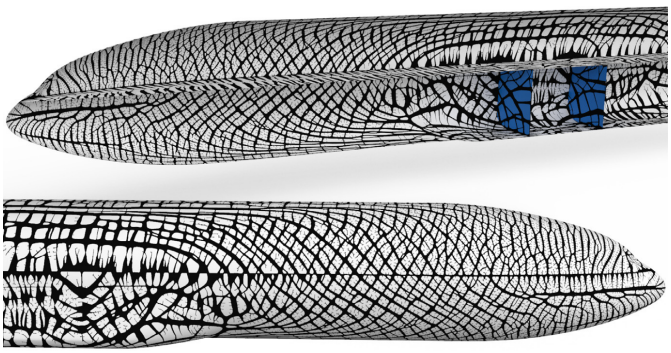


Fig. 16. Large local filter size - Overview of the resulting topology of the fuselage example. It should be noted that a white plate has been added to the components which have been modeled using a passive shell. Likewise, a blue background plate is added to the reinforcing plates near the wing, in order to improve the visualization. Computed with a filter radius of r_{solid} = 7.5 cm (7.9 elements) and local volume filter radius r_{LV} = 300 cm (260 elements).

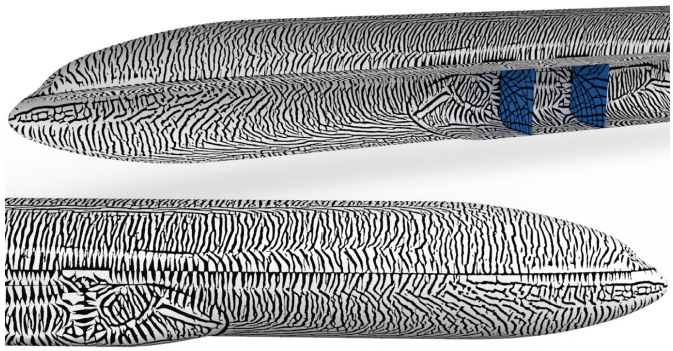


Fig. 19. Thick passive shell - Overview of the resulting topology of the fuselage example. It should be noted that a white plate has been added to the components which have been modeled using a passive shell. Likewise, a blue background plate is added to the reinforcing plates near the wing, in order to improve the visualization. Computed with a filter radius of $r_{\text{solid}} = 7.5\,\text{cm}$ (7.9 elements) and local volume filter radius $r_{\text{LV}} = 127\,\text{cm}$ (160 elements). The outer aircraft skin and floor are modeled with a passive central shell corresponding to 40% of the total shell thickness.

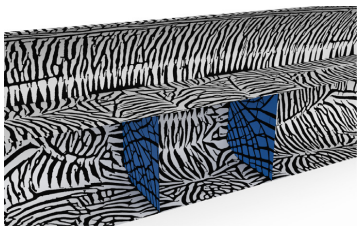


Fig. 20. Thick passive shell - Close up of the central part of the fuselage. Computed with a filter radius of $r_{\text{solid}} = 7.5\,\text{cm}$ (7.9 elements) and local volume filter radius $r_{\text{LV}} = 127\,\text{cm}$ (160 elements). The outer aircraft skin and floor are modeled with a passive central shell corresponding to 40% of the total shell thickness.

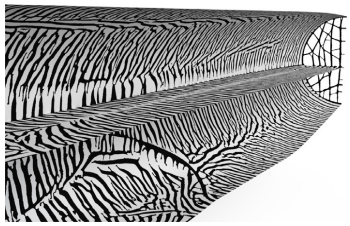


Fig. 21. Thick passive shell - Close up of the tail section of the fuselage. The reinforcement structure of the floor can also be seen. Computed with a filter radius of $r_{\text{solid}} = 7.5\,\text{cm}$ (7.9 elements) and local volume filter radius $r_{\text{LV}} = 127\,\text{cm}$ (160 elements). The outer aircraft skin and floor are modeled with a passive central shell corresponding to 40% of the total shell thickness.

applied at two edges at the intersection of the interior vertical stiffeners and the outer shell, and along the edge which connects a point from each curve. A single point on the top of the outer shell is given a Dirichlet boundary condition in the direction along the length of the fuselage, to avoid rigid body motion.

Two load cases are considered for the fuselage example. The first load case is an internal pressure of magnitude $35\,\text{N/cm}^2$ on the outer shell. In order to ensure equilibrium in the loads along the length of the fuselage, an additional force is distributed along the intersection of the outer shell and the vertical reinforcement near the tail. The second load case is a simplified gravity load. It is applied as a design independent body load of magnitude $15.6\,\text{N/cm}^3$ on both the outer shell and the floor panel.

The purpose of this model is to show the efficiency of the proposed method for large unstructured meshes. It should therefore be noted that this geometry does not represent an actual aircraft, due to the lack of windows and several internal reinforcing beams among other things. We remark that the applied load-cases and boundary conditions do not represent the physical loading of a fuselage, as such were not available to the authors. Furthermore, a realistic physical system would need to take additional effects into account, such as buckling, dynamics, thermal and electromagnetic responses. In relation to these considerations it is worth mentioning an additional benefit of using the local volume constraint. This constraint ensures that the final designs are free of long, slender and disconnected structural members, at least for the cases with a thin passive background shell. From the work of [49] it was found that such structures have significantly improved buckling resistance compared to designs obtained using the classical minimum compliance formulation. However, since this has not been proven or demonstrated by buckling analysis of the obtained shell structures, this interesting research question together with inclusion of global buckling constraints, is left for future investigations.

Figs. 13, 16 and 19 shows an overview of the resulting structures of the fuselage for all considered cases. For all three realizations it can be seen that rings are formed along the radial direction of the cylinder as a reinforcement against the internal pressure. Near the center of the fuselage, the rings are also connected by axial reinforcements, which carry the simplified gravity loading. These axial reinforcements slowly curve into the radial reinforcements in a smooth transition.

In the realizations with a thin passive support small load carrying arms appear in the unreinforced patches of the outer skin, as can be seen in Figs. 13 and 16. These arms appear to prevent the unreinforced patch to locally have a large deformation by adding some additional reinforcement. The arms are not observed in the case with a thick background plate, where the background bending stiffness is higher. This phenomenon is studied further in Appendix B. These two designs have compliance values of 818 kJ and 766 kJ, for the small and large local filter size respectively. It can be seen that the compliance value of the structure with larger allowed feature size is lower as expected, due to the increased design freedom (see Figs. 15 and 18).

In the case with a thicker passive domain, many of the supporting structures are not connected, as seen in Fig. 19. This effect is due to the high background stiffness of the passive shell, which will sufficiently

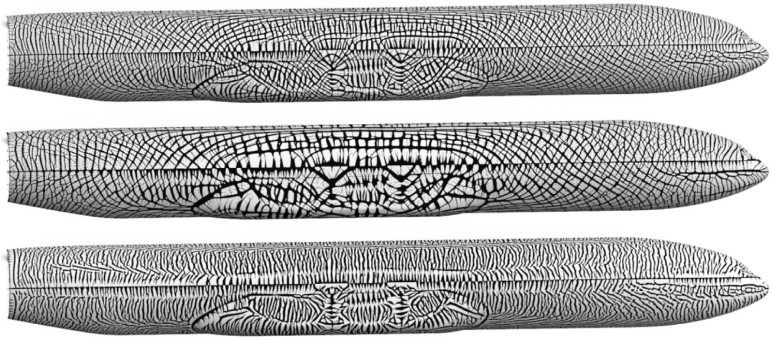


Fig. 22. Comparison of the various fuselage examples. From top to bottom: small local filter size, large local filter size, and thick passive shell.

carry the loads in areas with low strain energy density without the need for reinforcement. This design has a compliance value of 571 kJ, which is higher than both realizations with a thinner passive plate. This is expected, as more material is used compared to the two other designs. The full designs can be compared in Fig. 22, where the full fuselages are shown side by side (see Fig. 21).

It can be seen that all three realizations contain the same basic features. The smaller feature size results in more substructures, which form web like reinforcements. As the filter size is increased the optimization algorithm generates larger substructures. In Fig. 13 the first spyglass is a circle with radius corresponding to the used local volume filter. It can be seen that even the smallest features of the resulting structure are resolved by a large number of finite elements. It can also be observed that even the smallest used local volume filter is quite large compared to the overall structure. Due to the weighting factor which occurs in the PDE filtering, the distance between elements has a large effect on the resulting local volume fraction. Therefore, a large filter must be employed in order to achieve the desired local volume constraint.

The internal reinforcements near the wing connection are shown in Figs. 14, 17 and 20. They form a webbing support, which helps carrying the loads across the various sections. In the intersection of the support with the other shells it can be seen that the supporting material is placed as an extension of the reinforcements.

The floor panels are stiffened with cross beams, as shown in Figs. 14, 17 and 20 for the small local filter size. A line runs along the center of the plate with no reinforcing material, due to low bending moments at this point in the structure.

We remark that the 'small local filter size' case has a global volume fraction of 0.4524, which is considerably lower than the constraint. Likewise, the 'large local filter size' case has a volume fraction of 0.4517 and the 'thick passive shell' case has a volume fraction of 0.4316. This is due to the application of the volume constraint on the dilated field, and the local volume filter, which overestimates the volume in the p-norm aggregation used for the local volume constraint, as discussed in [23]. This could in principle be circumvented by an adaptive constraint technique, but has not been further pursued here.

6. Conclusion

This paper presents a design approach for generating optimized reinforcement or perforation of shell structures. The approach is based on a solid-shell element formulation and a multigrid preconditioned Krylov iterative method, which allows to efficiently solve the series of state equations associated with the optimization process. The multigrid preconditioner employs geometric multigrid restriction for the fine levels, and an algebraic multigrid method to obtain the solution of the coarse space problem. The approach overcomes the ill-conditioning problems arising partly due to shell formulations and partly due to high stiffness contrast in the element-based design parameterization. Using the proposed design method a series of academic optimization problems are solved, each repeatedly solving finite element problems using up to 11.6 million shell elements with +69.8 million degrees of freedom. This paves the way for solving large unstructured systems with shell elements for real life applications in future work.

The approach makes use of local-density control which ensures distributed material and hence a certain robustness towards unpredicted loads. For reinforcements problems, like the airplane fuselage problem considered, skin stiffness itself as well as applied minimum length-scale eliminate the need for locally connected reinforcements. This may seem counterintuitive but satisfies the applied pressure loading and optimization setting. Future studies includes buckling constraints, dynamics, thermal effects, electromagnetic effects, local load fluctuations or even finer design resolutions (and correspondingly smaller length scales imposed) which should help eliminate the aforementioned artifacts.

7. Reproducibility

The used meshes, geometry files, and final designs can be made available upon reasonable request.

CRediT authorship contribution statement

Erik A. Träff: Software, Methodology, Validation, Investigation, Writing - original draft, Visualization. **Ole Sigmund:** Methodology, Investigation, Writing - review & editing, Supervision, Project administration. **Niels Aage:** Software, Methodology, Investigation, Resources, Writing - review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Laureen L. Beghini and Glaucio H. Paulino for their communications regarding dimensions and loads for the original Lotte tower example. Furthermore, the authors would like to thank Erik Lund, for valuable scientific discussions.

The authors acknowledge the support of the Villum Foundation, Denmark through the Villum Investigator Project InnoTop.

Appendix A. Shell element formulation

This appendix provides a detailed summary of the used shell element formulation in order to facilitate reproduction of the proposed framework. For the original formulation, the reader is referred to [27].

A.1. Transformation matrix

In order to perform the numerical integration a transformation matrix is constructed. With onset in the normal vector to the shell surface at node I, here denoted $\mathbf{V}_3^I$, two additional vectors are computed using the method proposed by [30], yielding

$$\mathbf{V}_1^I = \mathbf{e_2} \times \mathbf{V}_3^I, \quad \mathbf{V}_2^I = \mathbf{V}_3^I \times \mathbf{V}_1^I. \tag{A.1}$$

or as

$$\mathbf{V}_2^I = \mathbf{V}_3^I \times \mathbf{e_1}, \quad \mathbf{V}_1^I = \mathbf{V}_2^I \times \mathbf{V}_3^I. \tag{A.2}$$

for the case where $\mathbf{V}_3$ and $\mathbf{e_2}$ are parallel. The node director basis inside a given element $\mathbf{V}_i$ is found by interpolating the basis vectors defined at each node $\mathbf{V}_i^I$ using the quadrilateral bilinear shape functions.

The transformation matrix from coordinate system A to coordinate system B is constructed using direction cosines as

$$t_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j, \quad \mathbf{v}_B = \begin{Bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{Bmatrix} \mathbf{v}_A. \tag{A.3}$$

which allows for the construction of the transformation matrix

$$\mathbf{T} = \begin{bmatrix} (t_{11})^2 & (t_{12})^2 & (t_{13})^2 & t_{11}t_{12} & t_{11}t_{13} & t_{12}t_{13} \\ (t_{21})^2 & (t_{22})^2 & (t_{23})^2 & t_{21}t_{22} & t_{21}t_{23} & t_{22}t_{23} \\ (t_{31})^2 & (t_{32})^2 & (t_{33})^2 & t_{31}t_{32} & t_{31}t_{33} & t_{32}t_{33} \\ 2t_{11}t_{21} & 2t_{12}t_{22} & 2t_{13}t_{23} & t_{11}t_{22}+t_{12}t_{21} & t_{11}t_{23}+t_{13}t_{21} & t_{12}t_{23}+t_{13}t_{22} \\ 2t_{11}t_{31} & 2t_{12}t_{32} & 2t_{13}t_{33} & t_{11}t_{32}+t_{12}t_{31} & t_{11}t_{33}+t_{13}t_{31} & t_{12}t_{33}+t_{13}t_{32} \\ 2t_{21}t_{31} & 2t_{22}t_{32} & 2t_{23}t_{33} & t_{21}t_{32}+t_{22}t_{31} & t_{21}t_{33}+t_{23}t_{31} & t_{22}t_{33}+t_{23}t_{32} \end{bmatrix} \tag{A.4}$$

which allows the strain and constitutive laws to be transformed as follows

$$\sigma_B = \mathbf{T}\sigma_A, \quad \mathbf{C}_B = \mathbf{T}^{\mathsf{T}}\mathbf{C}_A\mathbf{T}. \tag{A.5}$$

A.2. Isoparametric formulation

In the continuum shell element formulation the displacement field and physical coordinates are interpolated using standard trilinear hexahedral shapefunctions [30].

The Jacobian matrix used to transform from the isoparametric reference space to the usual orthonormal basis [30] is defined as follows.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} = \sum_{I=1}^{8} \begin{bmatrix} N_{I,\xi}x_I & N_{I,\eta}x_I & N_{I,\zeta}x_I \\ N_{I,\xi}y_I & N_{I,\eta}y_I & N_{I,\zeta}y_I \\ N_{I,\xi}z_I & N_{I,\eta}z_I & N_{I,\zeta}z_I \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 \end{bmatrix}, \tag{A.6}$$

$$\mathbf{J}^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix} = \begin{bmatrix} \mathbf{G^1}^{\mathsf{T}} \\ \mathbf{G^2}^{\mathsf{T}} \\ \mathbf{G^3}^{\mathsf{T}} \end{bmatrix}. \tag{A.7}$$

The covariant, $\mathbf{G}_I$, and contravariant, $\mathbf{G^I}^{\mathsf{T}}$, bases are found directly from the Jacobian and its inverse [27,30].

Table A.2
MITC tying points.

	ξ_L	η_L	ζ_L
A	−1	0	0
B	0	−1	0
C	1	0	0
D	0	1	0

Table A.3
ANS tying points.

	ξ	η	ζ
A_1	−1	−1	0
A_2	1	−1	0
A_3	1	1	0
A_4	−1	1	0

The design dependent constitutive matrix for an isotropic material using the SIMP model is expressed as follows

$$\mathbf{C} = \frac{\rho^p E_0}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ & 1-\nu & \nu & 0 & 0 & 0 \\ & & 1-\nu & 0 & 0 & 0 \\ & & & \frac{1-2\nu}{2} & 0 & 0 \\ & \text{Sym.} & & & \frac{1-2\nu}{2} & 0 \\ & & & & & \frac{1-2\nu}{2} \end{bmatrix} \tag{A.8}$$

where ρ is the design variable and $p = 1$ is the penalty parameter. The constitutive matrix is formulated in the contravariant basism, i.e. Eq. (A.4) is applied with $t_{ij} = \mathbf{V}_i \cdot \{\mathbf{G}^j\}$ to obtain $\mathbf{T}^{\mathbf{V}}$.

$$\bar{\mathbf{C}} = \mathbf{T}^{\mathbf{V}^{\mathsf{T}}}\mathbf{C}\mathbf{T}^{\mathbf{V}} \tag{A.9}$$

A.3. Strain displacement matrix

The bi-linear strain interpolation matrix $\mathbf{B}$ interpolates deformations at element nodes to strains in the covariant basis at an internal point defined by some given $\xi, \eta, \zeta \in [-1, 1]$. The matrix uses the regular interpolations for the in plane components $\bar{\epsilon}_{11}$, $\bar{\epsilon}_{22}$, and $\bar{\gamma}_{12}$ strains, while alternative interpolations are applied in the remaining three strains to prevent locking.

To prevent out-of-plane shear locking, i.e. in $\bar{\gamma}_{23}$ and $\bar{\gamma}_{31}$, the Mixed Integer Tensorial Components (MITC) is employed [50] based on the four tying points A, B, C, and D shown in Table A.2.

Additionally, to prohibit locking in the shell normal direction, the Assumed Normal Strain (ANS) interpolation is used for $\bar{\epsilon}_{33}$ [51,52], using four new tying points A_1, A_2, A_3, and A_4 as depicted in Table A.3.

This leads to following strain–displacement relation for node I

$$\mathbf{B}_I = \begin{bmatrix} \mathbf{G}_1^{\mathsf{T}} N_{I,\xi} \\ \mathbf{G}_2^{\mathsf{T}} N_{I,\eta} \\ \sum_{L=1}^{4} \frac{1}{4}(1+\xi_L\xi)(1+\eta_L\eta)\mathbf{G}_3^{\mathbf{L}^{\mathsf{T}}} N_{I,\zeta}^L \\ \mathbf{G}_1^{\mathsf{T}} N_{I,\eta} + \mathbf{G}_2^{\mathsf{T}} N_{I,xi} \\ \frac{1}{2}\left[(1-\eta)(\mathbf{G_3}^{\mathsf{T}} N_{I,\xi}^B + \mathbf{G_1}^{\mathbf{B}^{\mathsf{T}}} N_{I,\zeta}^B) + (1+\eta)(\mathbf{G_3}^{\mathbf{D}^{\mathsf{T}}} N_{I,1}^D + \mathbf{G_1}^{\mathbf{D}^{\mathsf{T}}} N_{I,\zeta}^D)\right] \\ \frac{1}{2}\left[(1-\xi)(\mathbf{G_3}^{\mathbf{A}^{\mathsf{T}}} N_{I,\eta}^A + \mathbf{G_2}^{\mathbf{A}^{\mathsf{T}}} N_{I,\zeta}^A) + (1+\eta)(\mathbf{G_3}^{\mathbf{C}^{\mathsf{T}}} N_{I,2}^C + \mathbf{G_2}^{\mathbf{C}^{\mathsf{T}}} N_{I,\zeta}^C)\right] \end{bmatrix} \tag{A.10}$$

Collecting the contribution from each of the nodal points, yields the complete strain–displacement matrix, i.e.

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \dots & \mathbf{B}_8 \end{bmatrix}, \tag{A.11}$$

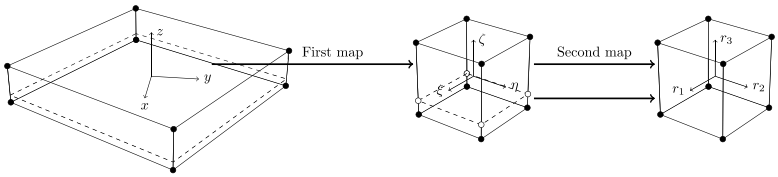


Fig. A.23. Illustration of the two isoparametric spaces.

A.4. Stiffness matrix integration and loads

The local stiffness matrix is obtained by standard Gaussian quadrature, i.e.

$$\begin{aligned}\mathbf{K}_e &= \int_{\Omega_e} \mathbf{B}^{\mathsf{T}}\bar{\mathbf{C}}\mathbf{B}\,\mathrm{d}V \\ &\approx \sum_{GP} |J|\mathbf{B}^{\mathsf{T}}\bar{\mathbf{C}}\mathbf{B}.\end{aligned} \tag{A.12}$$

If multiple material layers are used a new isoparametric space is introduced for each layer, as depicted in Fig. A.23. The resulting numerical integration scheme becomes

$$\mathbf{K}_e \approx \sum_{L=1}^{nlay} \sum_{GP} \mathbf{B}^{\mathsf{T}}\mathbf{C}_{\mathbf{L}}\mathbf{B}|J^{\xi}||J^{r}|, \tag{A.13}$$

The two external force integrals can be approximated by a Gaussian quadrature in the isoparametric space. The body force can integrated by the same quadrature rule s the stiffness matrix

$$\begin{aligned}\mathbf{f}_{\mathbf{vol}} &= \int_{\Omega_e} \mathbf{N}_e\hat{\mathbf{b}}\,\mathrm{d}V \\ &\approx \sum_{GP} |J|\mathbf{N}_e\hat{\mathbf{b}}.\end{aligned} \tag{A.14}$$

The surface forces, i.e. pressure loads and tractions, are obtained by integrating over the corresponding surface, determined by the two isoparametric coordinates $\xi_1, \xi_2 \in \{\xi, \eta, \zeta\}$. Let J_{ξ_1} denote the column of the Jacobian $\mathbf{J}$ corresponding to ξ_1. The surface integral is then computed by Gaussian quadrature as

$$\begin{aligned}\mathbf{f}_{\mathbf{surf}} &= \int_{\partial\Omega_e} \mathbf{N}_e\hat{\mathbf{t}}\,\mathrm{d}A \\ &\approx \sum_{GP} \|J_{\xi_1} \times J_{\xi_2}\|_2\mathbf{N}_e\hat{\mathbf{t}}.\end{aligned} \tag{A.15}$$

Now the resulting system of equations can be assembled using the regular

$$\mathbf{Ku} = \mathbf{f}_{\mathbf{vol}} + \mathbf{f}_{\mathbf{surf}}. \tag{A.16}$$

We remark that the resulting system of equations is poorly conditioned and hence, provides a challenge for iterative solvers.

A.5. Conditioning

To improve the performance of the proposed iterative solver, the Scaled Thickness Conditioning (STC) presented by [28] is included to reduce the condition number of the system matrix for thin continuum shells. The method uses a scaling parameter C, which for thin shells has the following optimal value

$$C^{\text{opt}} \approx \frac{l_1 + l_2}{2h}, \tag{A.17}$$

where l_1 and l_2 denote the element side lengths. The nodal scaling matrices are computed for nodes lying in the element mid-plane corresponding to the ANS nodes shown in Table A.3. For consistency, the nodes are denoted $\lambda \in \{A_1, A_2, A_3, A_4\}$ and the nodal scaling matrices are given as

$$\mathbf{s}_{\mathbf{1}}^{\lambda} = \frac{1}{\eta_\lambda}\begin{bmatrix} \frac{C+1}{2C} & 0 & 0 \\ 0 & \frac{C+1}{2C} & 0 \\ 0 & 0 & \frac{C+1}{2C} \end{bmatrix}, \quad \mathbf{s}_{\mathbf{2}}^{\lambda} = \frac{1}{\eta_\lambda}\begin{bmatrix} \frac{C-1}{2C} & 0 & 0 \\ 0 & \frac{C-1}{2C} & 0 \\ 0 & 0 & \frac{C-1}{2C} \end{bmatrix}, \tag{A.18}$$

where η_λ denotes the number of elements attached to node λ. Using the nodal scaling matrices, the element scaling matrices are constructed as follows

$$\mathbf{s}_e = \begin{bmatrix} \mathbf{s}_{\mathbf{2}}^{\mathbf{A_1}} & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_1}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{s}_{\mathbf{1}}^{\mathbf{A_1}} & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_1}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_2}} & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_2}} & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_3}} & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_3}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_3}} & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_3}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_4}} & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_4}} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{s}_{\mathbf{2}}^{\mathbf{A_4}} & \mathbf{s}_{\mathbf{1}}^{\mathbf{A_4}} \end{bmatrix}. \tag{A.19}$$

The nodal scaling matrices $\mathbf{s}_e$ are assembled to a symmetric global scaling matrix $\mathbf{S}$ by the regular finite element assembly. The scaling matrix is then applied to obtain the scaled stiffness matrix and force vector, i.e.

$$\mathbf{K}^C = \mathbf{SKS}, \quad \mathbf{f}^C = \mathbf{Sf}, \quad \mathbf{u} = \mathbf{Su}^C. \tag{A.20}$$

The resulting scaled linear system of equations now reads

$$\mathbf{Ku} = \mathbf{f} \Leftrightarrow \mathbf{K}^C\mathbf{u}^C = \mathbf{f}^C. \tag{A.21}$$

Note, that the matrix $\mathbf{S}$ is never assembled in order to reduce the memory usage. Instead, the scaled stiffness and forces are obtained during assembly by performing the corresponding local products.

Appendix B. A small study on the emergent 'arm' supports

The occurrence of non-connected reinforcements, dubbed 'arms', in the fuselage example merited further study. Intuition states that closed cells provide a better reinforcement, as the reinforcing material supports itself better and thus provides a stiffer reinforcement. In order to study whether the 'arms' provide some benefit two reinforced structures, shown in Figs. B.24 and B.25, are studied. Both cases are clamped plates of size 20×20 subjected to a uniform pressure load, where a fourth of the domain is modeled using symmetry conditions.

The reinforcement is placed on both sides of the base plate. The total thickness with reinforcements is set to a constant of 1, while the fraction of base thickness to reinforcement thickness is swept. Both plates are reinforced with material corresponding to a volume fraction of $V = 0.36$, such that the compliance values can be compared directly. The plates are resolved with 100×100 elements.

The resulting compliance values are shown in Table B.4. From there it can clearly be seen that the cross connected reinforcement performs better when the reinforced plate is thick compared to the

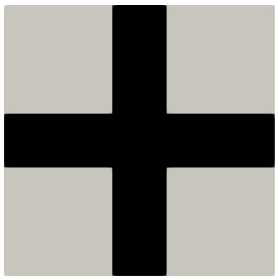


Fig. B.24. Plate with overlapping reinforcements shaped as a cross.

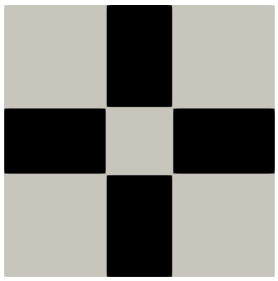


Fig. B.25. Plate with reinforcement in the form of 'arms', note that the four reinforcements are not connected.

Table B.4
Compliance values for both reinforcement configurations for a series of base plate thicknesses. It can be seen that the cross performs best when the stiffness contrast between the reinforced an non-reinforced areas is low, while the arms perform better when this contrasts is high.

Base plate thickness	Cross	Arms
40%	2.05229	2.442617
20%	3.629288	4.303002
10%	13.88358	12.2136
5%	94.69309	72.91165
1%	5821.474	4370.23
0.1%	11 634.7	8735.052
0.01%	11 708.35	8818.975

reinforcement. As the thickness decreases the 'arm' like structure becomes better performing. Therefore it can be concluded that the 'arm' reinforcements which occur in the fuselage designs are a part of the desired solution, and not some artifact due to enforced lengthscale and volume constraint.

References

[1] M.P. Bendsoe, O. Sigmund, Topology Optimization: Theory, Methods, and Applications, Springer Science & Business Media, 2003.

[2] U.T. Ringertz, Numerical methods for optimization of nonlinear shell structures, Struct. Optim. 4 (3) (1992) 193–198, http://dx.doi.org/10.1007/BF01742744, URL https://doi.org/10.1007/BF01742744.

[3] M.P. Bendsøe, Optimal shape design as a material distribution problem, J.-Struct.-Optim. 1 (1989) 193–202.

[4] M. Zhou, G.I.N. Rozvany, The COC algorithm, part II: Topological, geometry and generalized shape optimization, Comput. Methods Appl. Mech. Engrg. 89 (1–3) (1991) 309–336.

[5] R.J. Yang, C.J. Chen, C.H. Lee, Bead pattern optimization, Struct. Optim. 12 (4) (1996) 217–221, http://dx.doi.org/10.1007/BF01197359, URL https://doi.org/10.1007/BF01197359.

[6] J.H. Luo, H.C. Gea, Optimal bead orientation of 3d shell/plate structures, Finite Elem. Anal. Des. 31 (1) (1998) 55–71, http://dx.doi.org/10.1016/S0168-874X(98)00048-1, URL http://www.sciencedirect.com/science/article/pii/S0168874X98000481.

[7] Y.C. Lam, S. Santhikumar, Automated rib location and optimization for plate structures, Struct. Multidiscip. Optim. 25 (1) (2003) 35–45, http://dx.doi.org/10.1007/s00158-002-0270-7.

[8] J. Moita, J.I. Barbosa, C.M. Soares, C.M. Soares, Sensitivity analysis and optimal design of geometrically non-linear laminated plates and shells, Comput. Struct. 76 (1) (2000) 407–420, http://dx.doi.org/10.1016/S0045-7949(99)00164-9, URL http://www.sciencedirect.com/science/article/pii/S0045794999001649.

[9] K. Maute, E. Ramm, Adaptive topology optimization of shell structures, in: 6th Symposium on Multidisciplinary Analysis and Optimization, American Institute of Aeronautics and Astronautics, 1996, http://dx.doi.org/10.2514/6.1996-4114, URL http://arc.aiaa.org/doi/10.2514/6.1996-4114.

[10] R. Ansola, J. Canales, J.A. Tarrago, J. Rasmussen, On simultaneous shape and material layout optimization of shell structures, Struct. Multidiscip. Optim. 24 (3) (2002) 175–184, http://dx.doi.org/10.1007/s00158-002-0227-x.

[11] J. Stegmann, E. Lund, Discrete material optimization of general composite shell structures, Internat. J. Numer. Methods Engrg. 62 (14) (2005) 2009–2027, http://dx.doi.org/10.1002/nme.1259.

[12] S.N. Sørensen, E. Lund, Topology and thickness optimization of laminated composites including manufacturing constraints, Struct. Multidiscip. Optim. 48 (2) (2013) 1–17, http://dx.doi.org/10.1007/s00158-013-0904-y.

[13] C.F. Hvejsel, E. Lund, Material interpolation schemes for unified topology and multi-material optimization, Struct. Multidiscip. Optim. 43 (6) (2011) 811–825, http://dx.doi.org/10.1007/s00158-011-0625-z.

[14] M. Rais-Rohani, J. Lokits, Reinforcement layout and sizing optimization of composite submarine sail structures, Struct. Multidiscip. Optim. 34 (1) (2007) 75–90, http://dx.doi.org/10.1007/s00158-006-0066-2.

[15] S. Townsend, H.A. Kim, A level set topology optimization method for the buckling of shell structures, Struct. Multidiscip. Optim. 60 (5) (2019) 1783–1800, http://dx.doi.org/10.1007/s00158-019-02374-9.

[16] R. Kussmaul, J.G. Jónasson, M. Zogg, P. Ermanni, A novel computational framework for structural optimization with patched laminates, Struct. Multidiscip. Optim. 60 (5) (2019) 2073–2091, http://dx.doi.org/10.1007/s00158-019-02311-w.

[17] A.E. Albanesi, I. Peralta, F. Bre, B.A. Storti, V.D. Fachinotti, An optimization method based on the evolutionary and topology approaches to reduce the mass of composite wind turbine blades, Struct. Multidiscip. Optim. (2020) 1–25, http://dx.doi.org/10.1007/s00158-020-02518-2.

[18] W.A. Wall, M. Gee, E. Ramm, The challenge of a three-dimensional shell formulation-the conditioning problem, in: Proceedings of ECCM, Vol. 99, 2000.

[19] G.J. Kennedy, J.R. Martins, A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures, Finite Elem. Anal. Des. 87 (2014) 56–73.

[20] J. Fish, L. Pan, V. Belsky, S. Gomaa, Unstructured multigrid method for shells, Internat. J. Numer. Methods Engrg. 39 (7) (1996) 1181–1197.

[21] A. Evgrafov, C.J. Rupp, K. Maute, M.L. Dunn, Large-scale parallel topology optimization using a dual-primal substructuring solver, Struct. Multidiscip. Optim. 36 (2008) 329–345, http://dx.doi.org/10.1007/s00158-007-0190-7.

[22] N. Aage, E. Andreassen, B.S. Lazarov, O. Sigmund, Giga-voxel computational morphogenesis for structural design, Nature 550 (7674) (2017) 84–86.

[23] J. Wu, N. Aage, R. Westermann, O. Sigmund, Infill optimization for additive manufacturing—approaching bone-like porous structures, IEEE Trans. Vis. Comput. Graphics 24 (2) (2017) 1127–1140.

[24] S. Kambampati, C. Jauregui, K. Museth, H.A. Kim, Large-scale level set topology optimization for elasticity and heat conduction, Struct. Multidiscip. Optim. 2007 (2019) http://dx.doi.org/10.1007/s00158-019-02440-2, URL http://link.springer.com/10.1007/s00158-019-02440-2.

[25] H. Liu, B. Zhu, W. Matusik, M. Csail, Y. Hu, Narrow-band topology optimization on a sparsely populated grid, ACM Trans. Graph. 37 (6) (2018) 14, http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn.

[26] M. Baandrup, O. Sigmund, H. Polk, N. Aage, Closing the gap towards super-long suspension bridges using computational morphogenesis, Nature Commun. 11 (2735) (2020).

[27] S. Klinkel, F. Gruttmann, W. Wagner, A continuum based three-dimensional shell element for laminated structures, Comput. Struct. 71 (1) (1999) 43–62, http://dx.doi.org/10.1016/S0045-7949(98)00222-3, URL http://www.sciencedirect.com/science/article/pii/S0045794998002223.

[28] T. Klöppel, M.W. Gee, W.A. Wall, A scaled thickness conditioning for solid- and solid-shell discretizations of thin-walled structures, Comput. Methods Appl. Mech. Engrg. 200 (9) (2011) 1301–1310, http://dx.doi.org/10.1016/j.cma.2010.11.001, URL http://www.sciencedirect.com/science/article/pii/S0045782510003051.

[29] R.H. Macneal, R.L. Harder, A proposed standard set of problems to test finite element accuracy, Finite Elem. Anal. Des. 1 (1) (1985) 3–20, http://dx.doi.org/10.1016/0168-874X(85)90003-4.

[30] R.D. Cook, D.S. Malkus, M.E. Plesha, R.J. Witt, Concepts and Applications of Finite Element Analysis, fourth ed., Wiley, 2001.

[31] O. Amir, N. Aage, B.S. Lazarov, On multigrid-CG for efficient topology optimization, Struct. Multidiscip. Optim. 49 (5) (2014) 815–829, http://dx.doi.org/10.1007/s00158-013-1015-5, URL http://link.springer.com/10.1007/s.

[32] N. Aage, E. Andreassen, B.S. Lazarov, Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework, Struct. Multidiscip. Optim. 51 (3) (2015) 565–572, http://dx.doi.org/10.1007/s00158-014-1157-0.

[33] O. Axelsson, P.S. Vassilevski, A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning, SIAM J. Matrix Anal. Appl. 12 (4) (1991) 625–644.

[34] G.H. Golub, C.F. Van Loan, Matrix Computations, Vol. 3, JHU press, 2012.

[35] U.M. Yang, Parallel algebraic multigrid methods—high performance preconditioners, in: Numerical Solution of Partial Differential Equations on Parallel Computers, Springer, 2006, pp. 209–236.

[36] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page, 2019, http://www.mcs.anl.gov/petsc, URL http://www.mcs.anl.gov/petsc.

[37] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual. Tech. Rep. ANL-95/11 - Revision 3.11, Argonne National Laboratory, 2019, URL http://www.mcs.anl.gov/petsc.

[38] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.

[39] L.L. Stromberg, A. Beghini, W.F. Baker, G.H. Paulino, Application of layout and topology optimization using pattern gradation for the conceptual design of buildings, Struct. Multidiscip. Optim. 43 (2) (2011) 165–180.

[40] O. Sigmund, Manufacturing tolerant topology optimization, Acta Mech. Sinica 25 (2) (2009) 227–239.

[41] F. Wang, B.S. Lazarov, O. Sigmund, On projection methods, convergence and robust formulations in topology optimization, Struct. Multidiscip. Optim. 43 (6) (2011) 767–784.

[42] B.S. Lazarov, F. Wang, O. Sigmund, Length scale and manufacturability in density-based topology optimization, Arch. Appl. Mech. 86 (1–2) (2016) 189–218.

[43] B.S. Lazarov, O. Sigmund, Filters in topology optimization based on helmholtz-type differential equations, Internat. J. Numer. Methods Engrg. 86 (6) (2011) 765–781.

[44] K. Svanberg, The method of moving asymptotes—a new method for structural optimization, Internat. J. Numer. Methods Engrg. 24 (2) (1987) 359–373, http://dx.doi.org/10.1002/nme.1620240207, URL http://dx.doi.org/10.1002/nme.1620240207http://doi.wiley.com/10.1002/nme.1620240207.

[45] N. Aage, B.S. Lazarov, Parallel framework for topology optimization using the method of moving asymptotes, Struct. Multidiscip. Optim. 47 (4) (2013) 493–505, http://dx.doi.org/10.1007/s00158-012-0869-2, URL http://link.springer.com/10.1007/s00158-012-0869-2.

[46] Cubit 13.2 user documentation, 2019, https://cubit.s{and}ia.gov/public/13.2/help_manual/WebHelp/cubit_users_manual.html.

[47] G.A. da Silva, N. Aage, A.T. Beck, O. Sigmund, Three-dimensional manufacturing tolerant topology optimization with hundreds of millions of local stress constraints, Internat. J. Numer. Methods Engrg. (2020).

[48] Nasa common research model, 2019, https://commonresearchmodel.larc.nasa.gov/.

[49] A. Clausen, N. Aage, O. Sigmund, Exploiting additive manufacturing infill in topology optimization for improved buckling load, Engineering 2 (2) (2016) 250–257.

[50] K.J. Bathe, E.N. Dvorkin, A formulation of general shell elements—the use of mixed interpolation of tensorial components, Internat. J. Numer. Methods Engrg. 22 (3) (1986) 697–722, http://dx.doi.org/10.1002/nme.1620220312, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620220312, https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620220312.

[51] K. Park, G. Stanley, A curved c0 shell element based on assumed natural-coordinate strains, J. Appl. Mech. 53 (2) (1986) 278–290.

[52] C. Militello, C.A. Felippa, A variational justification of the assumed natural strain formulation of finite elements—i. variational principles, Comput. Struct. 34 (3) (1990) 431–438.

[P2]

An Advection–Diffusion Based Filter for Machinable Designs in Topology Optimization.

L. C. Høghøj and E. A. Träff.

Computer Methods in Applied Mechanics and Engineering 391 (March 2022): 114488.

doi:10.1016/j.cma.2021.114488.

[Published]

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 391 (2022) 114488
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

An advection–diffusion based filter for machinable designs in topology optimization

Lukas C. Høghøj*, Erik A. Träff

Department of Mechanical Engineering, Section for Solid Mechanics, Technical University of Denmark, Kgs. Lyngby, Denmark

Received 11 February 2021; received in revised form 10 November 2021; accepted 16 December 2021
Available online xxxx

Abstract

This paper introduces a simple formulation for topology optimization problems ensuring manufacturability by machining. The method distinguishes itself from existing methods by using the advection–diffusion equation with Robin boundary conditions to perform a filtering of the design variables. Furthermore, the approach is easy to implement on unstructured meshes and in a distributed memory setting. Finally, the proposed approach can be performed with few to no continuation steps in any system parameters. Applications are demonstrated with topology optimization on unstructured meshes with up to 64 million elements and up to 29 milling tool directions.

Keywords: Manufacturability; Machining; Milling; Filtering; Topology optimization

1. Introduction

Topology optimization (TO) is a widely adopted method for structural optimization [1,2]. The main advantage of TO is its ability to generate structures of arbitrary topology regardless of the prescribed initial conditions, i.e. little to no prior knowledge of the optimal structure is required. Due to this property, topology optimization is often used in the initial conceptualization of new designs. If a manufacturing method, such as milling, is imposed on the designer when conceptualizing a part, the main advantage of topology optimization can become a disadvantage. The freedom for TO to generate arbitrary topologies can result in structures with features, which are impossible to manufacture with traditional machining techniques. This can be characterized as a mismatch between the constraints placed on the designer and the constraints used in the TO formulation.

Extending the topology optimization approach to ensure manufacturability is an ongoing field of research. A wide variety of methods for topology optimization which ensures manufacturability by additive manufacturing have been proposed during recent years [3–7]. The additive manufacturing approaches usually constrain the allowed overhang of the structure in order to ensure self-support during the manufacturing process. Other types of manufacturing are also considered in the literature, such as Wang et al. [8], which introduces a method for ensuring topologies that can be extruded, by utilizing the so-called PDE-filter [9] with high anisotropy.

* Corresponding author.
E-mail address: luch@mek.dtu.dk (L.C. Høghøj).

https://doi.org/10.1016/j.cma.2021.114488

Several approaches for manufacturable TO using milling have been suggested. Gersborg and Andreasen [10] propose a method to consider an explicitly castable or millable design by using one design variable for each row or column of a structured grid. This approach is contrasted by Guest and Zhu [11], which proposes a similar method, but retains all design variables and uses cumulative summation along rows or columns as a filter to ensure that a design can be cast or milled. The approach by cumulative summation is extended to arbitrary directions by Langelaar [12], which maps the densities to a structured grid aligned with the milling direction in order to perform the summation. Finally, Hur et al. [13] recently proposed a milling filter, which uses a modified variant of the advection–diffusion equation to perform the cumulative summation for the level-set formulation, not quite unlike the proposed method. In Hur et al. [13] Dirichlet boundary conditions are used for the flow problem, which introduce the need for domain padding in order to correctly capture non-zero physical densities on the boundary of the design domain.

The approaches which use a cumulative sum provide a high degree of control allowing tool shapes to be taken into account [12]. If the sum operation is precomputed, the resulting matrix will contain many non-zero values and result in very high memory usage, which hinders application on large-scale systems. A matrix free implementation of the rotation mapping and cumulative sum is possible which significantly lowers the memory usage. However, such a matrix-free implementation is nontrivial to write for distributed memory systems. This paper presents an approach to perform the milling tool emulation, by solving the advection–diffusion equation with Robin boundary conditions. The proposed approach is conceptually simple and scales well, facilitating the solution of high resolution 3D problems. Like the cumulative summation approach, this method can be employed without any continuation scheme, and with little to no tuning for problem specific parameters. For now, however, the approach is limited to milling considerations without explicit tool shapes.

This paper is organized as follows; Section 2 introduces the formulation necessary for the advection–diffusion filtering step; Section 3 introduces the optimization problem used for numerical examples; Section 4 discusses the computational efficiency of the presented methodology; Section 5 presents two and three dimensional machinable results, with machining in one or multiple directions and Section 6 provides a discussion of the presented methodology and a conclusion.

2. Formulation

The milling filter is based on the approach presented by Langelaar [12], with the notable difference that the cumulative sums have been replaced by solving the advection–diffusion equation, with a dominating advective term. The filter relates the design variables to a 'physical' density field, which guarantees that the resulting void regions are reachable by a tool direction from outside the design domain. This relation is performed through a series of filters, of which most are already common in many topology optimization approaches. A brief overview of the steps used to compose the complete milling filter is presented in Fig. 1. The initial design variable is filtered in step 1, yielding the regularized design field, $\tilde{\rho}$, as discussed in Section 2.1. In step 2, the shadowing operations are performed, resulting in the shadowed intermediate design fields, $\check{\tilde{\rho}}_s$, as discussed in Section 2.2. Note that a shadowing step is performed for each prescribed tool direction. The shadowed intermediate design fields are then agglomerated in step 3, resulting in the intermediate agglomerated variable $\check{\tilde{\rho}}$, as discussed in Section 2.3. Finally, in step 4, the agglomerated variable is projected, as discussed in Section 2.4. The sensitivity analysis of the filter is included in Section 3.2 for completeness.

2.1. Step 1: Smoothing of design field

A density filter is initially applied to the design field. This can be performed either by a classical convolution approach [14], or by employing the so-called PDE-filter which solves a modified Poisson equation to perform density filtering [9,15]. The reader is referred to the cited articles for details on the respective filters and their sensitivity analysis.

The density filter is introduced to regularize the design field. This is especially useful when evaluating a black and white field, where the gradients will be zero in most of the void region (due to the SIMP interpolation) and in parts of the solid region (due to the agglomeration of shadowing steps). By including the density filter, non-zero gradient values will be smoothed into the domain with zero sensitivities.

In this work, the convolution approach is used for the 2D examples in Section 5.1, due to the widespread use of this approach for 2D codes, such as [16,17]. The PDE-filter approach [9,15] is used in the 3D examples, due to the increased performance and reduced memory footprint compared to the convolution approach.

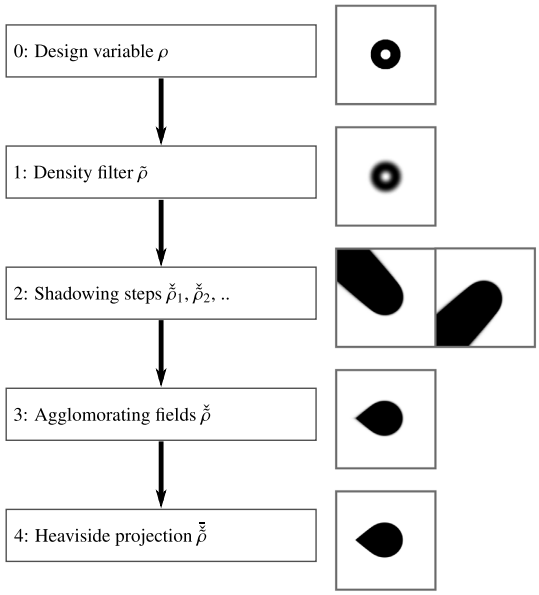


Fig. 1. Flowchart and visualization of the composition of filters to realize the milling filter. Two tool directions are used for the milling filter, resulting in two distinct shadowed fields.

2.2. Step 2: Shadowing by advection–diffusion

The advection–diffusion equation describes the transport of a scalar field, it is known from e.g. heat transfer problems. The steady state advection–diffusion equation is given as:

$$Pe\, u_i \frac{\partial T}{\partial x_i} - \frac{\partial^2 T}{\partial x_i^2} = q \tag{1}$$

where T is the transported field, u_i the prescribed advective field, q the normalized source term and Pe the Peclet number which is the ratio between the advective and diffusive transport.

In the presented shadowing methodology, a regularized design field $\tilde{\rho}$ is used as a source term, and the obtained transported field $\check{\tilde{\rho}}_s$ is shadowed in the direction of the constant advection term u_i^s of unity norm $\|u_i^s\| = 1$, corresponding to the machining direction, where s refers to the shadowing angle index, since multiple shadowing angles may be applied. For numerical reasons, the equation is normalized by the Peclet number. Thus, the equation solved using the filter notation becomes

$$u_i^s \frac{\partial \check{\tilde{\rho}}_s}{\partial x_i} - \frac{1}{Pe} \frac{\partial^2 \check{\tilde{\rho}}_s}{\partial x_i^2} = s\tilde{\rho} \quad \text{in } \Omega \tag{2}$$

where s is a factor scaling the source term on an element basis. The factor s is chosen such that the solution can go from 0 (void) to 1 (solid) over a single element and is further discussed in Section 2.2.2. The scaling is applied on the source rather than on the solution, as unstructured meshes are considered.

Solutions to Eq. (1) are known to become numerically unstable in cases with a high Peclet number. However, while paying a price in solution accuracy, the Finite Volume formulation using an upwind difference scheme on the advection term is known to be numerically stable for high Peclet numbers. The derivations of the used finite volume schemes are presented in Appendix A. Alternate stable discretization schemes with better numerical properties exist, e.g. [18], but these are also more cumbersome to implement.

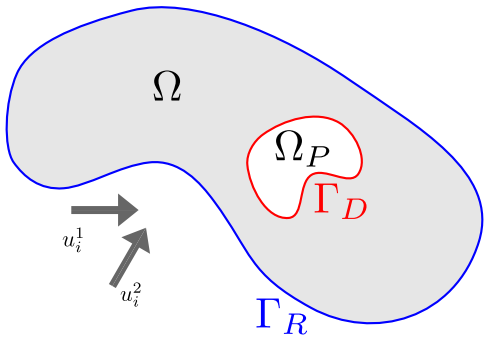


Fig. 2. Overview of the employed boundary conditions on the optimization domain Ω with shadowing directions θ_1 and θ_2. The passive solid domain Ω_P has Dirichlet-type boundary conditions along its boundary Γ_D to the optimization domain. Free boundaries of the optimization domain, Γ_R, have Robin-type boundary conditions imposed.

2.2.1. Boundary conditions

The advection–diffusion equation is only solved in the optimization domain. Hence, boundary conditions are imposed on free boundaries, as well as on the boundaries between the optimization- and passive solid domains — as illustrated in Fig. 2.

The employed boundary conditions on the free domain boundaries, Γ_R, should allow material to be placed adjacent to the free boundary. While the problem would not be well posed if only using Neumann boundary conditions, homogeneous Dirichlet boundary conditions do not allow the appearance of material at the boundary. Robin type boundary conditions are found to allow material at the boundary:

$$\check{\tilde{\rho}}_s + \mathbf{n}\frac{1}{s}\nabla\check{\tilde{\rho}}_s = 0 \quad \text{on} \quad \Gamma_R \tag{3}$$

where $\mathbf{n}$ is the outward pointing surface normal at the boundary.

On the domain boundaries adjacent to solid passive domains, Γ_D, Dirichlet boundary conditions are introduced to obtain the shadow of the passive domain in the corresponding milling direction. This ensures manufacturability of the full design, including the passive domains.

$$\check{\tilde{\rho}}_s = 1 \quad \text{on} \quad \Gamma_D \tag{4}$$

The implementation presented in the present work is based on unstructured meshes, hence passive void elements are not required. If one wants to use the presented methodology with the use of passive void elements, the authors suggest to ensure that the embedded design domain is machinable with the used tool directions.

Further details on the boundary conditions and their implementations are given in Appendix A.

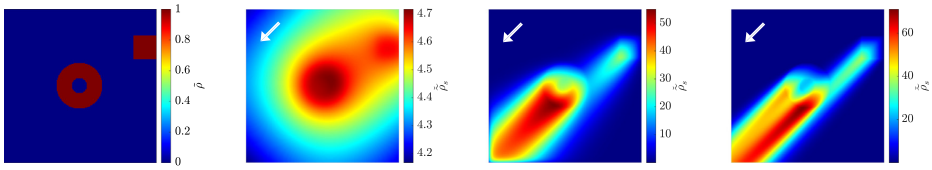
2.2.2. Choice of Peclet number and scaling factor

The s parameter, that is used for scaling the source term in the advection–diffusion equation, Eq. (2), is set such that a $\tilde{\rho} = 1$ value permits the shadowed variable to go from 0 to 1 over one cell face (normal to the shadowing direction). Assuming a very large Peclet number, and $|u_i| = 1$, this parameter should be set as:

$$s = \frac{1}{h_e} \tag{5}$$

where h_e is the average element side length.

The choice of Peclet number is critical to the effect of the advection–diffusion filtering step on the obtained results. As straight walls are machinable, the conic (diffusive) effects of the filter are to be minimized. The desired effect of the shadowing is hence to obtain features parallel to the specified direction, u_i, with as little diffusion as possible. This can be achieved by using a high Peclet number. When setting $Pe \gg 1$, the problem becomes advection dominated. This is also seen in Fig. 3, where the field from Fig. 3(a) is shadowed in a diagonal direction



(a) $\tilde{\rho}$ (b) $Pe = 0.1$ (c) $Pe = 100$ (d) $Pe = 10^4$

Fig. 3. Shadowing step performed on the field (a) with different Peclet numbers. Note the different color-bars for the respective cases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at three different Peclet numbers. With $Pe = 0.1$, as seen in Fig. 3(b), the diffusive effect is clearly dominating. When raising the Peclet number to $Pe = 100$, as demonstrated in Fig. 3(c), the regime is advection dominated, however diffusion still has a significant effect. In the case with $Pe = 10^4$, seen in Fig. 3(d), information is seen to almost exclusively propagate downstream. However, inside the shadowed region, some diffusive effects are still observed, as the propagated value decreases slightly in the downstream direction. This diffusive effect will always be present when solving the equations with the finite volume method, as some numerical diffusion occurs regardless of the chosen Peclet number. As the values inside these regions are usually $\check{\tilde{\rho}}_s \gg 1$, this is not an issue as they are thresholded using the smooth Heaviside projection, introduced in Section 2.4. The influence of this artifact can be reduced by setting the Heaviside projection threshold η to some low value close to zero, forcing a more conservative placement of material downstream.

It should be noted that the Peclet number is defined based on a characteristic length scale of the considered domain. This implies that when larger domains are considered, a lower Peclet number will suffice to achieve similar results. For reference, Fig. 3 is computed on a domain of unit side-lengths. A rule of thumb to choose the Peclet number based on a characteristic domain length l_c is

$$Pe_{\text{thumb}} = \frac{10^4}{l_c}. \tag{6}$$

The constant 10^4 is by no means a fixed rule, as acceptable results have also been found using constant factors of 10^3, 10^5, and 10^6. We do however advise ensuring that the used Peclet number satisfies $Pe\, l_c > 10^3$.

2.3. Step 3: Field agglomeration

The resulting fields of the shadowing for each tool direction $\check{\tilde{\rho}}_s$ are agglomerated to a single field using the p-mean, which provides a differentiable approximation to the min operator.

$$\check{\tilde{\rho}}^e = \left(\frac{1}{n_s} \sum_{s=1}^{n_s} (\check{\tilde{\rho}}_s^e)^p \right)^{\frac{1}{p}} \approx \min_s \check{\tilde{\rho}}_s^e \tag{7}$$

Here $p < 0$ denotes an agglomeration parameter. The p-mean becomes a more accurate approximation of the min operator, but also more non-linear, as $p \to -\infty$. In practice $p = -4$ is commonly used throughout this article, as it was found that the field projection step reduces the effect of a low accuracy approximation of the min operator. When a large number of milling directions are used it becomes necessary to use a lower value of p, in order to ensure a sufficiently good approximation of the min operator. Other field agglomeration approaches are possible as discussed in detail in [19]. For instance, Langelaar [12] suggested using the KS functional for the agglomeration. For large problems, however, where $\check{\tilde{\rho}}_s$ can take very large values, due to the high number of elements along one axis, this resulted in numerical issues with the KS function, and the p-mean was found to be more robust in these cases.

2.4. Step 4: Field projection

The aggregated fields are projected using the smoothened Heaviside filter [20] in order to bound the final density field between 0 and 1:

$$\bar{\bar{\rho}}^e = \frac{\tanh(\beta\eta) + \tanh(\beta(\check{\rho}^e - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))} \tag{8}$$

where β is the projection sharpness and η the projection threshold. A desirable side effect is that a black-and-white design is also obtained. Throughout this article the value $\eta = 0.5$ is used, while β is chosen to either 8 or 10 depending on the required projection sharpness.

3. Optimization formulation

The general optimization problem is formulated on the design variables ρ. $\bar{\bar{\rho}}$ is used to denote the 'physical' density field, and is obtained by performing all milling filter steps described in Section 2 on the design field ρ. The posed optimization problem is minimization of the compliance with a global volume constraint:

$$\begin{aligned} \min_{\rho \in \mathbb{R}^N} \quad & c(\rho) = \mathbf{u}^\top \mathbf{K}(\bar{\bar{\rho}})\mathbf{u} \\ \text{s.t.} \quad & g(\rho) = \frac{V(\bar{\bar{\rho}})}{V^*} - 1 \leq 0 \\ & \mathbf{K}(\bar{\bar{\rho}})\mathbf{u} - \mathbf{p} = 0 \\ & 0 \leq \rho_i \leq 1 \quad i = 1 \dots N \end{aligned} \tag{9}$$

where $\mathbf{K}(\bar{\bar{\rho}})$ is the stiffness matrix, $\mathbf{p}$ the load vector and $\mathbf{u}$ the resulting displacement vector. The volume constraint is enforced based on the volume of the current design, $V(\bar{\bar{\rho}})$, and the maximum allowable volume, V^*.

The Solid Isotropic Material Penalization (SIMP) interpolation scheme is used to map the element projected design variable $\bar{\bar{\rho}}^e$ to the corresponding Young's modulus [14]:

$$E(\bar{\bar{\rho}}^e) = E_{min} + (\bar{\bar{\rho}}^e)^p \, (E_{max} - E_{min}) \tag{10}$$

Where E_{min} and E_{max} are the lower and upper values of the Young's modulus, respectively and p is the SIMP penalization parameter.

3.1. Parameters for the method of moving asymptotes

The optimization problem is solved using the Method of Moving Asymptotes [21] implemented by [22] with non standard parameters. These parameters are necessary as the milling filter projection does not work for low projection sharpnesses, as this results in $\bar{\bar{\rho}}$ not being bounded correctly. This requires the usage of a constant and high β value. When optimizing with a constant projection sharpness, the problem becomes highly sensitive and the asymptotes in the MMA algorithm hence need to be tightened, as discussed by Guest et al. [23].

Therefore, to eliminate oscillatory behavior the initial asymptotes are tightened by setting the initial asymptote spacing, `asyinit`,

$$s_0 = \frac{0.5}{2\beta + 1}. \tag{11}$$

Furthermore the parameter to widen the asymptotes, `asyincr`, is set to 1.05 and the parameter to tighten them, `asydecr`, to 0.65 instead of 1.2 and 0.7, respectively, in the standard distribution of MMA. The outer mover limit was set to 0.1 per design iteration.

A scaling parameter is applied on the objective function, such that its value is 10 in the first optimization iteration. When the scaled objective function gets below 0.1 the scaling parameter is updated by a factor of 10. Likewise the volume constraint is used in a scaled formulation, as shown in Eq. (9). This ensures a good numerical conditioning on the optimization problem.

3.2. Sensitivity analysis

The sensitivities of the objective- and constraint functions are obtained with respect to the final projected element variable, $\bar{\bar{\tilde{\rho}}}^e$. The sensitivities of a function f with respect to the projected variable, $\frac{\partial f}{\partial \bar{\bar{\tilde{\rho}}}}$ are projected back to the design variable, ρ, by use of the chainrule:

$$\frac{df}{d\rho} = \frac{df}{d\bar{\bar{\tilde{\rho}}}} \frac{\partial \bar{\bar{\tilde{\rho}}}}{\partial \check{\bar{\tilde{\rho}}}} \sum_{s=1}^{n_s} \left[\frac{\partial \check{\bar{\tilde{\rho}}}}{\partial \check{\tilde{\rho}}_s} \frac{\partial \check{\tilde{\rho}}_s}{\partial \bar{\tilde{\rho}}} \right] \frac{\partial \tilde{\rho}}{\partial \rho}. \tag{12}$$

where the therm $\frac{\partial \check{\tilde{\rho}}_s}{\partial \bar{\tilde{\rho}}}$, represents the chainrule term of the shadowing step. To correct the filter for the advection–diffusion equations the discretized milling filtering step is considered in tensor notation:

$$A^s_{i,j} \check{\tilde{\rho}}_{s,j} = \tilde{\rho}_i \tag{13}$$

differentiating the expression with respect to the regularized field $\tilde{\rho}_i$ and multiplying with the sensitivities $\frac{\partial f}{\partial \check{\tilde{\rho}}_{s,j}}$ with respect to the shadow yields

$$A^s_{i,j} \left. \frac{\partial f}{\partial \tilde{\rho}_i} \right|_s = \frac{\partial f}{\partial \check{\tilde{\rho}}_{s,j}} \tag{14}$$

which in matrix notation yields to solving the transposed filtering equation:

$$\mathbf{A}^{s\intercal} \left. \frac{\partial f}{\partial \tilde{\rho}} \right|_s = \frac{\partial f}{\partial \check{\tilde{\rho}}_s} \tag{15}$$

where the partial derivatives $\left. \frac{\partial f}{\partial \tilde{\rho}} \right|_s$ need to be summed for all shadowing steps to obtain the full sensitivity with respect to the regularized field, as seen in Eq. (12). More details on the sensitivity analysis and chain-rule terms are outlined in Appendix B.

4. Computational efficiency

The computational cost of introducing the proposed milling filter depends directly on both the desired number of milling directions and the number of constraints used in the optimization formulation. For every milling direction an advection–diffusion equation needs to be solved at every design iteration. Likewise, an adjoint system of equations needs to be solved for every tool direction for the objective function and for every constraint every design iteration. When also accounting for the used density filter this results in a total of $(n_{\text{constraints}}+2)(n_{\text{tools}}+1)$ linear system solves for filtering, where $n_{\text{constraints}}$ denotes the number of constraints and n_{tools} denotes the number of tool directions.

The worst case presented in this article uses one constraint and 29 tool directions, resulting in 90 auxiliary linear systems solutions every design iteration. While many auxiliary systems might need to be solved, the time required to solve the finite volume problems is significantly lower than the time required to solve the linear elasticity state equation, since they are scalar problems and are hence much cheaper than the vectorial state problem. Furthermore, the system matrix of the advection–diffusion problem does not change between design iterations, amortizing the computational cost of constructing preconditioners, as these can be stored throughout the optimization process. The cost of solving the advection–diffusion equation depends on a large set of parameters, where some of the most significant are; the used discretization of the equations, the number of elements in the mesh, and the used method to solve the resulting system of equations.

The used method for solving the resulting system of equations is also of great importance for the efficiency. If the system is sufficiently small, e.g. less than one million elements, a direct solution technique is the most viable strategy. As the operator does not change with the design iterations, only one LU factorization is required for every tool direction for the entirety of the optimization problem. This is the approach used in the 2D examples from Section 5.1.

If the number of elements grows too large, iterative solvers are required. For the presented 3D examples the Flexible Generalized Minimal RESidual [24] method was used, with additive Schwartz preconditioning, which in turn uses incomplete LU to approximate the local solutions. All of these solvers are implemented by the Portable, Extensible Toolkit for Scientific Computation [25]. More advanced preconditioners, such as the multigrid method

Table 1
Summary of used optimization parameters.

	2D cantilever	3D cantilever	GE Bracket
Projection sharpness β	8	8	10
Projection threshold η	0.5	0.5	0.5
SIMP penalization p	3, 5	3	3
Filter radius r_{min}	0.03	0.0085	0.7 mm
Peclet number Pe	10^4	10^4	500
Heat source factor	1	100	1
p-mean penalization	-3	1, -4, -6	-4
Young's module E_{max}	1	1	113.8 GPa
Young's module ratio E_{min}/E_{max}	10^{-9}	$10^{-4} \rightarrow 10^{-7}$	$10^{-3} \rightarrow 10^{-6}$
Poisson's ratio ν	0.3	0.3	0.342
Initial design variable ρ_{init}	0.005, 0.02	0.002	0.004
Volume fraction V^*	0.5	0.15	0.137135
Number of design iterations	-	250	200

employed in solving the linear elastic state equation, can also be implemented in order to improve the computational efficiency of solving the advection–diffusion equation [26,27]. This was not deemed necessary during our numerical examples, as the solution time of the auxiliary problems never exceeded reasonable limits.

The discretization technique for the advection–diffusion problem is a finite volume scheme with an upwind difference scheme, which is known to be numerically stable for large Peclet numbers. Unfortunately, this means that the adjoint problem corresponds to a downwind difference scheme, which is not so stable. It is observed that solving the adjoint problem requires up to 6 times the iterations before the convergence criteria are reached, compared to the forward problem for the advection–diffusion equation. This could be mitigated by explicitly constructing the adjoint operator using an upwind scheme, although this might introduce a small error due to the difference in the discretization schemes. Alternately, another discretization scheme might have better properties for both the forward and adjoint problems, e.g. [18].

Using a finite volume scheme for solving the advection–diffusion PDE means that the design field representation corresponds to one degree of freedom per design element. Hence no mapping is required between elements and nodes.

5. Numerical examples

The introduced methodology is first demonstrated on a two dimensional cantilever beam example, which was also used by [12]. The same cantilever beam is then optimized in three dimensions, showcasing how the methodology works on large scale. Finally the GE engine bracket [28] is optimized to illustrate how the methodology can be used in an industrial example. The three dimensional examples are implemented in a preexisting in-house unstructured topology optimization code [29]. The optimization parameters for all three cases are given in Table 1. Note that the homogeneous initial value of the design vector is also stated as an optimization parameter, as the choice of this parameter is important as the resulting initial design should neither be pure solid nor pure void. This is due to the sensitivities vanishing in SIMP at pure void and in the Heaviside projection if the agglomerated field is $\breve{\tilde{\rho}} \gg 1$.

As 3D results can be difficult to fully understand from the figures provided in the following sections, STL files (thresholded at $\breve{\tilde{\rho}} > 0.5$) of all results are available as supplementary material and have been uploaded to https://doi.org/10.11583/DTU.16930198 [30].

5.1. Two dimensional cantilever examples

In the following, the optimization results on a two dimensional cantilever beam are discussed. The design domain of dimensions 2×1 is clamped on the left hand side and point loaded in the lower right corner, as shown in Fig. 4.

The domain is discretized by 200×100 rectangular elements. In addition to the parameters seen in Table 1, the milling directions are selected to a variety of combinations. The SIMP penalization is set to $p = 5$ for the cases with one or two milling directions and to $p = 3$ for other cases. The increased SIMP penalization is used in the

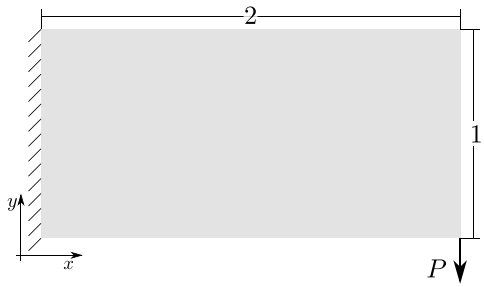


Fig. 4. Problem setup of the 2D cantilever beam example. Material can be placed inside the design domain of dimensions 2 × 1, depicted in gray.

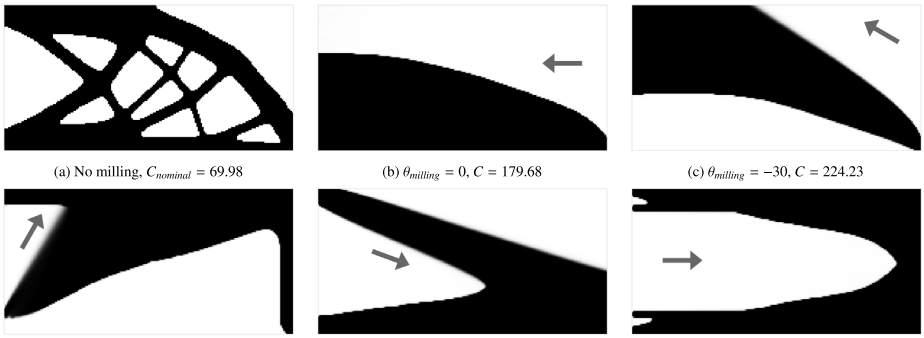


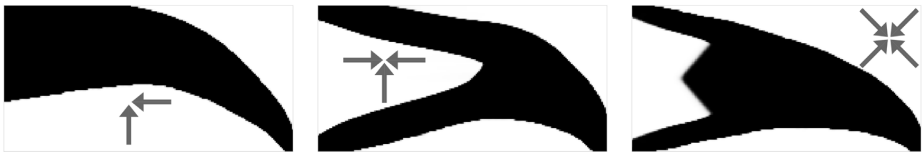
Fig. 5. Reference design and single tool direction designs for the 2D case. The final projected variable, $\tilde{\bar{\rho}}$ is shown.

cases with few tool directions to avoid designs with intermediate densities since these designs have a tendency to perform relatively well in terms of compliance, and therefore need to be penalized further. Furthermore, the initial design variable value is set to $\rho_{init} = 0.005$ for cases with a single milling direction and $\rho_{init} = 0.02$ if multiple directions are considered.

A reference example is optimized using the robust formulation [20]. For the reference, the β value is continuated, and the SIMP penalization power is set to $p = 1$. The projection thresholds are set to $\eta_{dilated} = 0.2$, $\eta_{nominal} = 0.5$ and $\eta_{eroded} = 0.8$, for the dilated, nominal and eroded fields, respectively. This choice of threshold values should ensure a minimum length scale of $0.9r_{min} = 2.7$ elements. The volume fraction on the dilated field is updated regularly, such that the nominal volume fraction matches the desired one, shown in Table 1. The obtained reference example is seen in Fig. 5(a). The objective values and number of design iterations corresponding to the designs are seen in Table 2. It should be noted that this reference design is constrained by a minimal length-scale of the solid members, while the upcoming designs are not.

Results optimized with a single milling direction are shown in Fig. 5. It is observed that all of the resulting designs are distinctly shaped by their corresponding tool direction. From Table 2 it can be seen that all of the obtained structures have a higher compliance than the reference design. The best performing design Fig. 5(e) has a compliance, which is 25% higher than the reference design. From this, it is seen that constraining the design to be filtered from a single tool direction severely limits the design freedom of the optimization algorithm.

Designs optimized with multiple milling directions are shown in Fig. 6. Having multiple milling directions impacts the obtained design, as material can be removed from multiple directions. This is most notably seen when comparing Figs. 5(b), 6(a) and 6(b). In the first case, Fig. 5(b), material can only be removed from the right hand



(a) $\theta_{milling} = \{-90,\ 0\}$, $C = 220.46$ (b) $\theta_{milling} = \{-90,\ 0,\ 180\}$, $C = 86.92$ (c) $\theta_{milling} = \{45,\ 135,\ 225,\ 315\}$, $C = 115.14$

Fig. 6. Resulting designs when using multiple tool directions for the 2D case. The final projected variable, $\bar{\bar{\rho}}$ is shown.

Table 2
Comparison of compliances of the two dimensional results.

Figure	$\theta_{milling}$	C	C/C_{ref}	Num. It.
5(a)		69.98	1.00	650
5(b)	0	179.68	2.57	501
5(c)	-30	224.23	3.20	500
5(d)	240	211.04	3.01	243
5(e)	160	87.49	1.25	195
5(f)	180	287.93	4.11	194
6(a)	$\{-90,\ 0\}$	220.46	3.15	500
6(b)	$\{-90,\ 0,\ 180\}$	86.92	1.24	491
6(c)	$\{45,\ 135,\ 225,\ 315\}$	115.14	1.65	536

side, resulting in material being placed in the lower left triangle part of the design domain. In the second case, seen in Fig. 6(a), material can also be removed from below, resulting in material being placed near the upper design domain boundary. In Table 2, it is seen that the first case, with a single direction performs better than the latter one, which indicates that a local minimum with an inferior performance has been obtained. The same local minimum was also observed by Langelaar [12]. The solutions find different local minima, due to the tool directions heavily affecting the initial density layout which is found using a homogeneous design variable distribution. This could potentially be avoided with an other starting guess.

With a third milling direction, as seen in Fig. 6(b), the obtained design can become attached to both extremities of the supported side. In the performance comparison, Table 2, it is also seen that this design performs better than the ones presented in Figs. 5(b) and 6(a) and only 24% worse than the reference design from Fig. 5(a).

The design obtained when optimizing with the four diagonal tool directions is seen in Fig. 6(c). The design has some features similar to the one from Fig. 6(b). However, the groove going into the structure from the support plane cannot be deeper with the selected tool orientations. The selected tool directions are also responsible for the triangular shape, seen inside the groove. This triangular shape does not bear any load, however it cannot be removed with the selected tool directions. The design performs 32% worse than the one seen in Fig. 6(b), which is most likely due to the high amount of non-load bearing material.

It is noted that the performance of the designs with $\theta_{milling} = 160$ and $\theta_{milling} = \{-90,\ 0,\ 180\}$ have very similar compliances and are also similar in a qualitative manner. It is noted that the design with $\theta_{milling} = 160$, seen in Fig. 5(e), is feasible with both sets of milling directions, this is not the case for the design obtained with $\theta_{milling} = \{-90,\ 0,\ 180\}$, seen in Fig. 6(b). This might explain the slightly lower compliance of the design seen in Fig. 6(b).

A large qualitative similarity is observed between the designs obtained by Langelaar [12] and the ones presented in Figs. 5 and 6 — with the exception of the reference design (Fig. 5(a)). The differences in the reference designs are probably due to different formulations being used for obtaining the reference design by [12] and the presented work.

5.2. Three dimensional cantilever examples

The three dimensional cantilever beam has a similar design domain as the two dimensional one from Fig. 4, where the out of plane direction is modeled with the thickness 1. The applied load is a line load at the corresponding 3

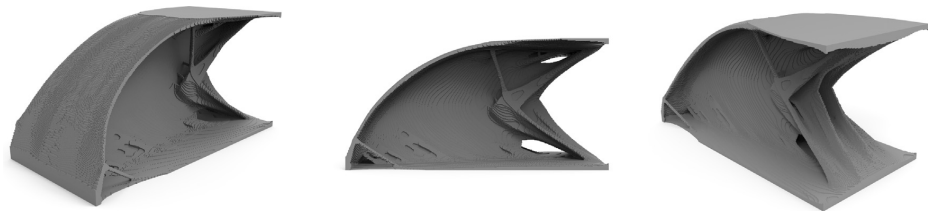


Fig. 7. Reference cantilever beam obtained with no milling and a robust formulation (nominal design is shown). $C_{nominal} = 1.15 \times 10^7$.

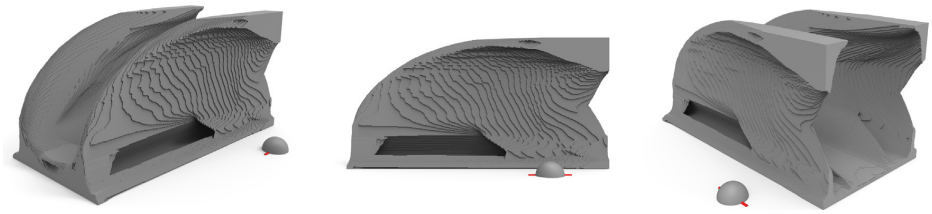


Fig. 8. Cantilever beam with a single milling direction from the front and back. $C = 1.37 \times 10^7$.

dimensional position. The domain is discretized with a mesh containing 31.25 million hexahedral elements. The applied filter radius corresponds to approximately 1.5 elements. The optimization settings are shown in Table 1. The results are computed on 10 nodes at the DTU Sophia cluster, which has 2 AMD EPYC 7351 16-core processors and 125 GB memory at every node. The solution time is between 55 and 450 s per design iteration, depending on the number of milling directions.

A benchmark cantilever beam without imposed manufacturability is optimized using the robust formulation [20]. In this case, the β-value for the Heaviside projection sharpness is ramped up to a final value of 59.4, and the benchmark optimization process is run for 700 iterations. The threshold values are set to $\eta = 0.2$ in the dilated-, $\eta = 0.5$ in the nominal- and $\eta = 0.8$ in the eroded field — resulting in a minimum length scale of 0.00765. The reference design obtained with the robust formulation is seen in Fig. 7. The final robust objective, used as a reference is evaluated on the nominal field as a postprocessing step to the optimization.

In the early design iterations of optimizations considering machining, the structure is not connected to the support and load, due to a milling tool coming from the support plane — or loaded line. This renders the linear elasticity equation very difficult and slow to solve due to the ill conditioning. As a remedy, a continuation scheme is applied on the Young's module contrast, which is reduced by a factor of 10 after 20, 40 and 60 iterations. The optimization is hence started with $E_{min} = 1 \times 10^{-4} E_{max}$ and reaches $E_{min} = 1 \times 10^{-7} E_{max}$ at iteration 60.

Several milling direction cases are considered. In the cases with a single milling direction, the p norm power is set to $p = 1$ as no element-wise minimum is required. In cases with less than 10 milling directions, it is set to $p = -4$ and to $p = -6$ for cases with more than 10 directions. The p norm power is changed with the varying number of tool directions due to the changing characteristics of the p norm. A lower value of p results in a less accurate approximation of the min operator, but also results in a less non-linear operator, which is desired during the optimization.

The cantilever beam is optimized using milling directions normal to the supported plane (in both directions). The obtained design is seen in Fig. 8. The obtained structure consists of two vertical plates that curve down towards the loaded line. A plate at the bottom of the domain connects the two vertical ones.

In Fig. 9, the design obtained with a single milling direction from top is shown. The design consists of two vertical plates, however they are not connected at the bottom of the domain. A third vertical plate connects the two other ones along the loaded line. This third plate is higher than the plates to which it is connected. This could be to add some bending stiffness along the loaded line. Note that numeric diffusion creates a non feasible disconnection between the part and the support plane at approximately one third of the support height. The disconnection is

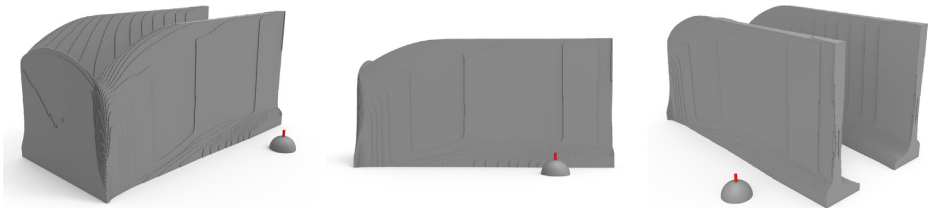


Fig. 9. Cantilever beam with a single milling direction from the top. $C = 1.53 \times 10^7$.

Fig. 10. Cantilever beam with a single milling direction from the side. $C = 1.37 \times 10^7$.

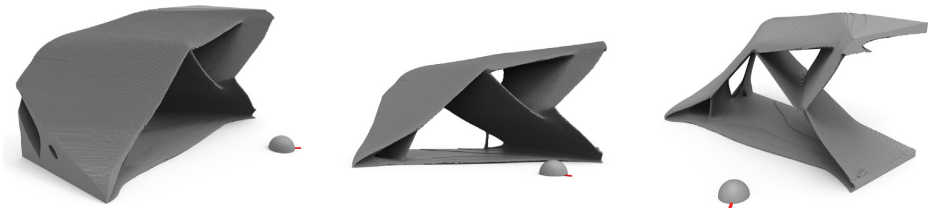


Fig. 11. Cantilever beam with a single milling direction with a 45 degree angle in the plane. $C = 1.62 \times 10^7$.

due to numerical diffusion in the discretization of the advection diffusion equation, which is discussed in detail in Section 2.2.2 along with potential solutions to this problem.

A cantilever beam obtained with milling only from one side, orthogonal to both the load direction and the normal of the clamped surface, is shown in Fig. 10. The design cross section resembles the two dimensional cantilever beam design with no milling constraints seen in Fig. 5(a). However, this three dimensional design is not a pure extruded version of the two dimensional design, as less structure is present in the side where the milling tool comes from. This probably reflects that a design, where the load can be transferred to a reduced part of the domain performs better.

A beam optimized with a similar direction, which has been rotated 45 degrees towards the supporting plane, is seen in Fig. 11. The resulting cross section of the structure is seen to resemble the one from Fig. 10. However, the skewed milling direction affects the design drastically, both qualitatively and in terms of compliance value. It can be observed that the angle forces material to be placed at locations that are unfavorable, as none of the other designs make use of them (most notably the upper left side over the loaded line).

The three previously shown cases with directions normal to the domain boundaries are combined to a single case, with six milling directions, where the milling directions follow the cartesian coordinate system, in positive and negative directions. The design obtained with six milling directions is shown in Fig. 12. The obtained design resembles the one obtained with no milling constraint, Fig. 7. However, the hollow interior of the side walls from the reference is infeasible with the milling filter, and has hence been replaced by holes in the structure connecting the top and bottom.

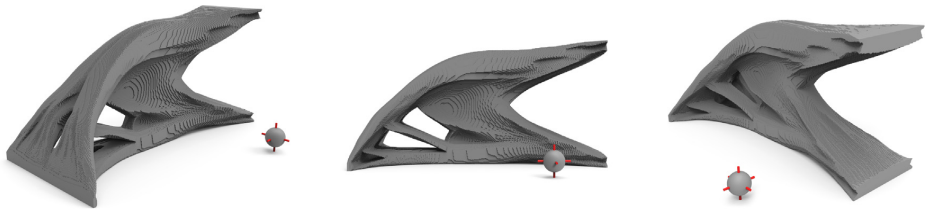


Fig. 12. Cantilever beam with six milling directions, normal to the bounding box surfaces. $C = 1.33 \times 10^7$.

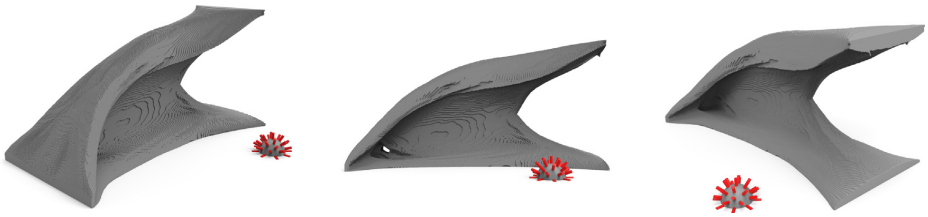


Fig. 13. Cantilever beam with 29 milling directions, coming from direction from the northern hemisphere. $C = 1.32 \times 10^7$.

Table 3
Comparison of performance of the 3D cantilever beam designs, obtained with different milling directions and comparison with reference design obtained without milling.

Figure	Num. tools	C	C/C_{ref}
7	0	1.15×10^7	1.0
8	2	1.37×10^7	1.192
9	1	1.53×10^7	1.331
10	1	1.37×10^7	1.194
11	1	1.62×10^7	1.408
12	6	1.33×10^7	1.159
13	29	1.32×10^7	1.148

A cantilever beam is also optimized using a set of 29 milling directions, all coming from the northern hemisphere, as described by [12]. The design obtained with this combination of milling directions is seen in Fig. 13. The design is very similar to the one using 6 directions seen in Fig. 12. However, most of the holes through the structure in the middle have been filled, and the thickness of that structure is varied instead. Some of the overhang near the only remaining hole next to the loaded line are only realizable due to some of the milling directions with the skewed angles.

The performance of the obtained designs is compared in Table 3. As expected, all designs obtained with the machinability constraints perform worse than the benchmark using the robust formulation. Furthermore, it is also observed that adding more milling directions improves the performance of the design, as this also increases the design freedom of the optimization process. In the work of Hur et al. [13], the orientations of the tools are also determined by the optimization. This can be considered an alternative to representing the full span of admissible tool directions, as the tool directions obtained by the optimization need to be admissible for manufacturing. There is currently no direct comparison of the resulting structures from the two approaches for topology optimization considering multi-axis machining.

As in the two dimensional results, discussed in Section 5.1, it is again seen that the chosen milling directions have a large influence on the performance of the obtained designs. If one wishes to only consider only a single milling direction, the direction should be chosen carefully, as it can have a large influence on the performance of

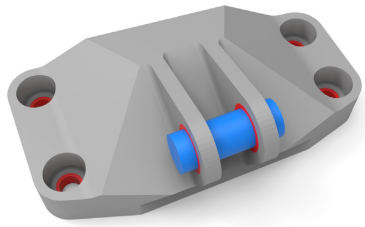


Fig. 14. Geometry of GE Jet Engine Bracket example. The pin with increased stiffness is shown in blue, while the passive rings which are kept solid are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the optimized design. A surprising discrepancy in performance is observed between the designs obtained with the milling directions normal to the supported plane from Fig. 8 and the one obtained with the milling direction from above from Fig. 9. The design with the milling direction from above performs $\approx 11\%$ worse, even though the obtained designs are somewhat similar, qualitatively.

5.3. Three dimensional GE bracket examples

To demonstrate the real-world possibilities of the milling filter, the GE jet engine bracket [31] is used as an industrial example, which is shown in Fig. 14. The original bracket geometry, material properties, and load cases are adapted from [28,31], with slight simplifications. The pinned boundary interfaces are modeled by clamping the inner surface of the bolt interface. The rigid pin is included in the model using 10 times the Young's modulus of the used material. The rigid pin is modeled as a part which is fused to the passive ring, as correct modeling of contact during structural optimization is beyond the scope of this article. All six interfaces in the bracket model are assigned a passive solid ring, in order to ensure that an interface exists. The compliance of the structure is minimized subject to a constraint of volume fraction $V^* = 0.137$ in the design domain, corresponding to a total bracket weight of 300 g for the design and the passive solid rings.

The minimum Young's modulus E_{min} is updated using a simple continuation scheme, as also described in Section 5.2. The optimization begins with $E_{min} = 1 \times 10^{-3} E_{max}$ which is decreased by an order of magnitude every 15 iterations until reaching $E_{min} = 1 \times 10^{-6} E_{max}$ at iteration 45.

The large size of the computational domain ($160 \times 110 \times 90$ mm) allows to use a lower Peclet number than in the previous examples. This is due to the Peclet number being a function of the characteristic domain length.

It should be noted that the bracket design domain is non-convex due to the two flanges which support the bolt. This non-convexity has some important implications for the solution of the advection–diffusion equation on the bracket domain. The boundaries will not transport information across the gaps in the design domain, treating every domain surface as an outer surface from where a tool might remove material. This will have an effect, as placing material in one flange, will not force the formulation to place material in the other flange due to the milling filter. As the flanges represent a small part of the domain, this error is limited, although present.

As discussed in Section 2.2, the surfaces which separate the passive rings from the design domain are modeled using a Dirichlet boundary condition in the advection–diffusion equation in order to ensure that the passive rings also introduce material downstream into the design domain. Furthermore, it should be noted that while the stiffened pin, shown in blue in Fig. 14, is included in the model, it is omitted from the visualization.

The bracket examples are solved using a mesh of 64 million hexahedral elements. The results are computed on 20 nodes at the DTU Sophia cluster. The computations take between 17 and 31 h, where solving the milling filters and their adjoint problems usually take between 25 and 45 s for each tool direction, compared to the linear elasticity solver which usually takes between 70 and 100 s for each load case. .

A reference example is included for the bracket in order to compare the resulting structures and compliance values. The reference example is computed using the robust formulation [20] with β-continuation, using a PDE-based filter with consistent boundary conditions [32], which emulate a padded domain. The robust formulation is

Fig. 15. Reference design of the bracket example with Robust formulation and consistent boundary condition for PDE-filter. $C = 4.21 \times 10^7$ J.

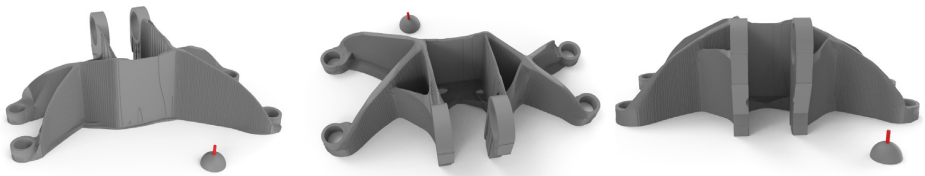


Fig. 16. Design of the bracket example with one milling direction in the z-axis. $C = 9.28 \times 10^7$ J.

Table 4
Comparison of compliance of the GE jet engine bracket examples.

Figure	Num. tools	C	C/C_{ref}
15	0	4.21×10^7 J	1
16	1	9.28×10^7 J	2.20
17	5	4.89×10^7 J	1.16
18	5	4.91×10^7 J	1.17
19	4	5.12×10^7 J	1.22
20	4	6.09×10^7 J	1.45
22	1	9.52×10^7 J	2.26

used due to the included Heaviside filter, which results in discrete 0–1 designs, like the proposed milling filter. This allows for a more accurate comparison of compliance values, as neither method is forced to include intermediate densities in the final design. Though, it should be noted that the robust formulation imposes a length-scale on the final design, which is not present in the milling filter. The resulting reference structure is shown in Fig. 15.

The first example, which is performed using only one milling direction from the top is shown in Fig. 16. It can be seen that the volume underneath the passive rings, which contain the rigid pin, is set to solid due to the Dirichlet boundary condition on the shadowing step — with the exception of the undercut under the two rings, which is not a part of the design domain. This prescribes the use of an already small global volume fraction, leaving little material for the remaining structure. All four supporting bolts are connected by thin legs, which connect in a cross shaped structure. From the resulting compliance values, shown in Table 4, it is clear that the single direction performs poorly, resulting in approximately twice the compliance value of the reference case. This is somewhat expected, given the restriction of available material and severely restricted design freedom.

Two additional bracket examples each using five mutually orthogonal milling directions are visualized by the red bars along with the resulting structures in Figs. 17 and 18. The structures both differ from the reference by being more compact almost truss-like, as opposed to the shell-like reference structure. The compliance value of both examples with five directions is very close to the reference value, considering the design restrictions, deviating with only 16% and 17%. From this, and similar cantilever examples with many tool directions, it can be seen that structures generated, for the considered linear elastic compliance minimization problems with milling constraints using many tool directions, perform well compared to reference designs, when taking into account the severe limitations in design freedom.

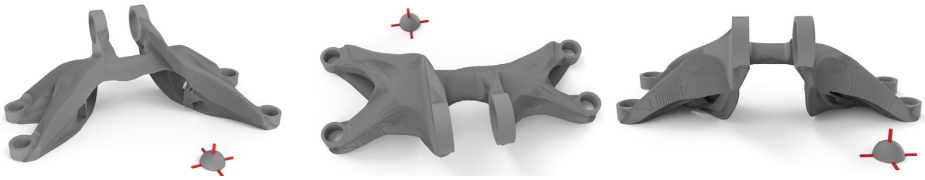


Fig. 17. Design with five milling directions. $C = 4.89 \times 10^7$ J.

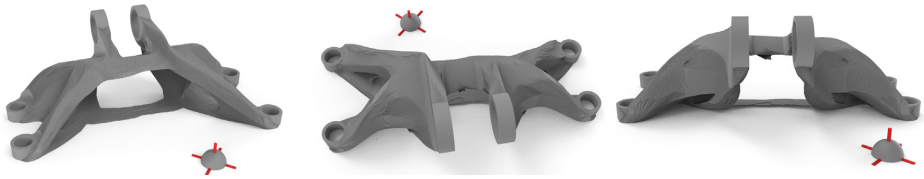


Fig. 18. Design with five milling directions. $C = 4.91 \times 10^7$ J.

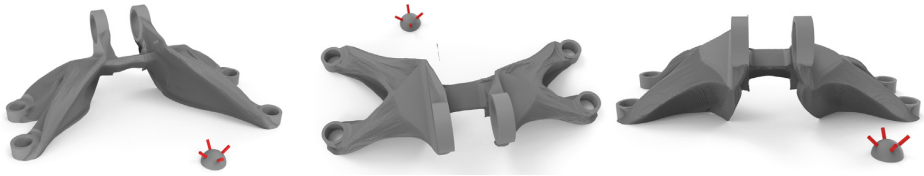


Fig. 19. Design with four milling directions at an 45 degree angle. $C = 5.12 \times 10^7$ J.

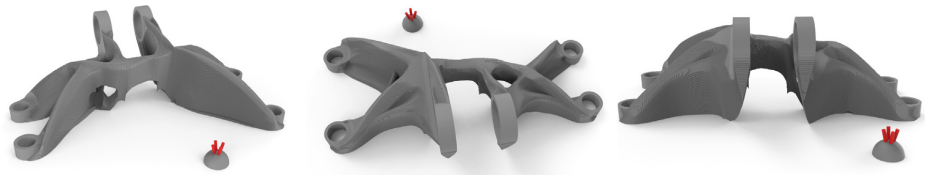


Fig. 20. Design with four milling directions at an 14.4 degree angle. $C = 6.09 \times 10^7$ J.

Additional numerical examples are also given with more constrained tool directions, to evaluate the milling approach itself, as structures under very limiting manufacturing constraints are generated. Fig. 19 shows the bracket with four non cartesian tool directions. Again, an agreeable structure with only 22% increase in compliance compared to the reference is obtained. If closely examined, the structure shows two 'spikes' in the center of the structure. These spikes do not carry any load, but cannot be removed by any of the given tool directions without also removing some vital load-carrying structural member. Therefore they can be compared to the phenomenon seen in Fig. 6(c), where non-load carrying material is also present.

An alternate version of the four directions is shown in Fig. 20, where all four directions have a 14.4 degree angle to the vertical axis and are aligned with one of the other axes. In this case, the resulting structure is forced to use a significant amount of material under the supporting rings in form of spikes, as can also be seen in the zoom into the corresponding area in Fig. 21. These spikes appear as there is no milling direction which allows the removal of this material. Similarly, any structural member which has material removed from under it needs to have a 'wedge'-like shape as the tools need to reach the void under the structure.

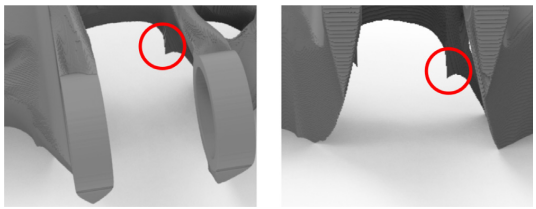


Fig. 21. Zoom in on the wedge-like features from the design obtained with the four milling directions at a 14.4 degree angle, seen in Fig. 20.

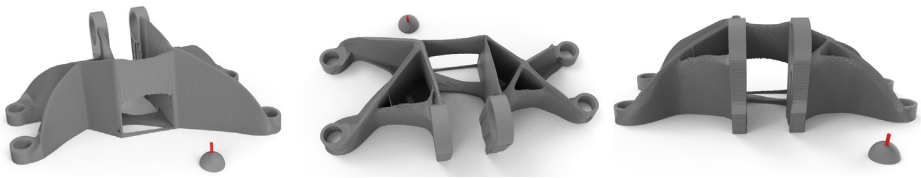


Fig. 22. Design with one milling directions at an 14.4 degree angle. $C = 9.52 \times 10^7$ J.

Finally, a last example with a single tool direction is shown in Fig. 22, but this time the tool direction is at a 14.4 degree angle with the vertical axis, like in the previous example, but only the direction coming from the side of the bolt flanges is considered. Again it can be seen that a thin-walled structure appears, but this time all of the walls have the expected angle. One curious artifact of this example is that the optimizer is able to circumvent the milling filter, and generate a hole in the bottom part of the central wall. This is understandable in a structural sense as the bottom part of the wall carries little load, but still a very undesirable artifact in the milling filter. This effect occurs due to an interesting interplay between the advection–diffusion filter and the Heaviside projection. As can be seen in Fig. 3(d), the advection–diffusion-filtered value can slowly decrease along the advection direction. This is partly due to the numerical diffusion due to the used upwind scheme, which is necessary for numerical stability. If the value gets below the Heaviside parameter η, which is 0.5 in this case, the Heaviside projection will project the design towards 0, rather than 1, allowing such a hole to appear. This can be solved by setting a lower η value, although the example is included here for completeness.

6. Conclusion

The article presents a formulation for performing topology optimization of manufacturable structures through milling by using a PDE-based alternative to a cumulative summation. The proposed method has been shown to generate manufacturability by machining in a number of numerical examples in two and three dimensions, resulting in topologies where all void regions are reachable through a tool direction from outside the domain. The proposed method has also shown itself to work for larger topology optimization problems, as shown by the numerical examples consisting of 64 million elements. From the large-scale examples it can also be seen that the increased number of elements allows thin shell members in the structures to be resolved, notably in the bracket. This would not be possible on lower resolutions.

The proposed method for performing the cumulative summation by solving the advection–diffusion equation, can also be considered a significant simplification of the mapping process [12], since it is not necessary to keep track of multiple meshes, and their relative orientation.

In the shown examples the proposed method accounts for a non-negligible fraction of the total computational effort of the optimization. However, there is no hard limit to the size of problems, for which this method can be applied. Furthermore, it should be possible to improve the wall clock time of the advection–diffusion solver [18,26,27] — which is left as future work.

Some disadvantages of the PDE-based formulation can also be noted from the numerical examples. The advection–diffusion equation is unable to transport material through regions which have not been meshed, such as between the two flanges of the bracket example. When a milling direction across flanges is chosen, no coupling between the flanges occurs.

The transport problem could in principle be solved by developing special boundary conditions, which couple the field values across the gap in the mesh. This would require a quite non-trivial effort in terms of mathematical and software development. An easier alternative, would be to use an auxiliary mesh of a convex bounding volume (e.g. the bounding box) of the domain, and solve the advection–diffusion problems on this mesh, somewhat similar to the approach used in [12]. Solving the advection–diffusion equation in the convex bounding volume is still expected to scale better with the number of elements, compared to performing a cumulative summation.

Another disadvantage is the lack of control in the tool-shape, which is possible when explicitly computing the cumulative summations [12]. A further challenge is the diffusivity term of the advection–diffusion equation, which is needed for numerical stability. It can be seen from the presented results that the diffusive terms can be kept very small.

While it is difficult to guarantee a tool shape based on the solution of the advection–diffusion equation, it could be possible to ensure a minimum member thickness of the features in the resulting design. This could be done by providing eroded, nominal, and dilated variations of the design field during Heaviside projection phase. These fields could then be used to provide a robust formulation based on [20]. This extension of the milling formulation is left to further work, as it is considered somewhat a separate issue from the advection–diffusion based approach.

The proposed method sometimes shows erosion of structures downstream due to numerical diffusion, as discussed in Section 2.2.2 and shown by example in Fig. 22. While this drawback is important to note for the proposed method, the effects can be alleviated somewhat by appropriate choice of Heaviside parameters in the following projection.

It is found that the chosen milling directions have a large impact on the compliance of the resulting structures when few milling directions are used. This is to be expected, as the milling filter poses a large restriction on the possible structures.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Assoc. Prof. Dr. Schousboe Andreasen, Assoc. Prof. Dr. Aage, and Prof. Dr. Sigmund for technical discussions, proof-reading, and supervising our PhDs.

The authors acknowledge the support of the Villum Foundation, Denmark for funding the aforementioned PhDs through the Villum Investigator Project InnoTop.

Appendix A. Finite volume implementation with upwind scheme

The advection–diffusion equation from Eq. (2) is integrated over a control volume. After applying the Gauss divergence theorem, the finite volume form of the equation over a control volume V with boundaries S is obtained [33]:

$$\int_S n_i u_i^s \check{\tilde{\rho}}_s dS - \int_S \frac{1}{Pe} n_i \nabla \check{\tilde{\rho}}_s dS = \int_V \tilde{\rho} dV \tag{A.1}$$

Finite difference schemes are used to describe the fluxes through the faces of the cell. The stencil for the respective face consists of the two cells adjacent to the face. The diffusive flux through the face is described using a Center Difference Scheme (CDS):

$$n_i \nabla \check{\tilde{\rho}}_s \Big|_f = \left| \frac{\mathbf{d}}{\mathbf{d} \cdot \mathbf{A}} |\mathbf{A}|^2 \right| \frac{\check{\tilde{\rho}}_s \Big|_N - \check{\tilde{\rho}}_s \Big|_P}{|\mathbf{d}|} \tag{A.2}$$

where $\mathbf{A}$ is the normal area vector and $\mathbf{d}$ the distance between the cell center and the neighbor cell center [34]. The subscript f refers to the face value, P to the cell and N to the neighbor adjacent to the face.

In order to guarantee numerical stability, an upwind difference scheme is used for the advection coefficient in the hence, the interpolated value at the face is the one from the upstream cell:

$$\begin{aligned} &\text{if} : (n_i u_i^s)_f > 0 : \ \check{\check{\rho}}_s\Big|_f = \check{\check{\rho}}_s\Big|_N \\ &\text{if} : (n_i u_i^s)_f < 0 : \ \check{\check{\rho}}_s\Big|_f = \check{\check{\rho}}_s\Big|_P \end{aligned} \tag{A.3}$$

where n_i is the outward pointing surface normal.

The Robin boundary conditions are introduced in Eq. (3), using a linear interpolation of the $\check{\check{\rho}}_s$ value on the cell boundary and CDS equation (A.2) on the corresponding gradient, the following boundary condition contribution is found for the boundary face:

$$\check{\check{\rho}}_s\Big|_N = -\check{\check{\rho}}_s\Big|_P \frac{\frac{1}{2} - n_i \frac{1}{s}\frac{\mathbf{A}_d}{|\mathbf{d}||\mathbf{A}|}}{\frac{1}{2} + n_i \frac{1}{s}\frac{\mathbf{A}_d}{|\mathbf{d}||\mathbf{A}|}} \tag{A.4}$$

Dirichlet boundary conditions are implemented adjacent to passive domains. Here, the surface contribution is given as:

$$\check{\check{\rho}}_s = 1 \quad x \in \partial\Omega \tag{A.5}$$

the right hand side in these cells is corrected with:

$$-2a_f \tag{A.6}$$

where a_f is the surface contribution of the boundary face.

Appendix B. Sensitivity analysis

The sensitivity of the full milling filter can be found by applying the chain rule on all operations, as seen in Eq. (12).

The derivative of the compliance w.r.t. the used density field can be found for each element as [14]

$$\frac{dc}{d\check{\check{\rho}}^e} = -\mathbf{u}^\top \frac{d\mathbf{K}}{d\check{\check{\rho}}^e}\mathbf{u} = -\mathbf{u}^{\mathbf{e}\top}\frac{d\mathbf{K}^\mathbf{e}}{d\check{\check{\rho}}^e}\mathbf{u}^\mathbf{e} \tag{B.1}$$

where the e superscripts denote the element density, deformations, and local stiffness matrix.

The partial derivative of the Heaviside projection can be found analytically for every entry of the density field as

$$\frac{\partial\check{\check{\rho}}^e}{\partial\check{\rho}^e} = \frac{\beta\left(1 - \tanh^2(\beta(\check{\rho}^e - \eta))\right)}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))} \tag{B.2}$$

The partial derivative of the p-norm agglomeration for a given field i can be found analytically for every element e as

$$\frac{\partial\check{\rho}^e}{\partial\check{\rho}_s^e} = \left(\frac{1}{n}\sum_{s=1}^{n_s}(\check{\rho}_s^e)^p\right)^{\frac{1}{p}-1}\frac{1}{n}(\check{\rho}_s^e)^{p-1} \tag{B.3}$$

The final partial derivative $\frac{\partial\tilde{\rho}}{\partial\rho}$ depends on the chosen density filtering strategy. In both cases applying the partial derivative corresponds to performing the density filtering on the accumulated sensitivity $\frac{dc}{d\tilde{\rho}}$.

References

[1] M.P. Bendsøe, O.O. Sigmund, Topology Optimization : Theory, Methods, and Applications, Springer, 2003, p. 370.
[2] O. Sigmund, K. Maute, Topology optimization approaches, Struct. Multidiscip. Optim. 48 (6) (2013) 1031–1055, http://dx.doi.org/10.1007/s00158-013-0978-6.
[3] A.T. Gaynor, J.K. Guest, Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design, Struct. Multidiscip. Optim. 54 (5) (2016) 1157–1172, http://dx.doi.org/10.1007/s00158-016-1551-x.
[4] M. Langelaar, Topology optimization of 3D self-supporting structures for additive manufacturing, Addit. Manuf. (2016) 11.

[5] Y. Mass, O. Amir, Topology optimization for additive manufacturing: Accounting for overhang limitations using a virtual skeleton, Addit. Manuf. 18 (2017) 58–73, http://dx.doi.org/10.1016/j.addma.2017.08.001.
[6] X. Qian, Undercut and overhang angle control in topology optimization: A density gradient based integral approach, Int. J. Numer. Methods Eng. 111 (3) (2017) 247–272, http://dx.doi.org/10.1002/nme.5461.
[7] K. Zhang, G. Cheng, L. Xu, Topology optimization considering overhang constraint in additive manufacturing, Comput. Struct. 212 (2019) 86–100, http://dx.doi.org/10.1016/j.compstruc.2018.10.011.
[8] B. Wang, Y. Zhou, K. Tian, G. Wang, Novel implementation of extrusion constraint in topology optimization by Helmholtz-type anisotropic filter, Struct. Multidiscip. Optim. 62 (4) (2020) 2091–2100, http://dx.doi.org/10.1007/s00158-020-02597-1.
[9] B.S. Lazarov, O. Sigmund, Filters in topology optimization based on Helmholtz-type differential equations, Int. J. Numer. Methods Eng. 86 (6) (2011) 765–781, http://dx.doi.org/10.1002/nme.3072.
[10] A.R. Gersborg, C.S. Andreasen, An explicit parameterization for casting constraints in gradient driven topology optimization, Struct. Multidiscip. Optim. 44 (6) (2011) 875–881, http://dx.doi.org/10.1007/s00158-011-0632-0.
[11] J.K. Guest, M. Zhu, Casting and milling restrictions in topology optimization via projection-based algorithms, in: Volume 3: 38th Design Automation Conference, Parts A And B, American Society of Mechanical Engineers, 2012, pp. 913–920, http://dx.doi.org/10.1115/DETC2012-71507.
[12] M. Langelaar, Topology optimization for multi-axis machining, Comput. Methods Appl. Mech. Eng. (2019) 27, http://dx.doi.org/10.1016/j.cma.2019.03.037.
[13] D.Y. Hur, Y. Sato, T. Yamada, K. Izui, S. Nishiwaki, Level-set based topology optimization considering milling directions via fictitious physical model, Mech. Eng. J. (2020) http://dx.doi.org/10.1299/mej.20-00226.
[14] M.P. Bendsøe, O. Sigmund, Topology Optimization: Theory, Methods, and Applications, Springer, 2003.
[15] B.S. Lazarov, F. Wang, O. Sigmund, Length scale and manufacturability in density-based topology optimization, Arch. Appl. Mech. 86 (1–2) (2016) 189–218, http://dx.doi.org/10.1007/s00419-015-1106-4.
[16] E. Andreassen, A. Clausen, M. Schevenels, B.S. Lazarov, O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code, Struct. Multidiscip. Optim. 43 (1) (2011) 1–16, http://dx.doi.org/10.1007/s00158-010-0594-7.
[17] F. Ferrari, O. Sigmund, A new generation 99 line matlab code for compliance topology optimization and its extension to 3D, Struct. Multidiscip. Optim. 62 (4) (2020) 2211–2228, http://dx.doi.org/10.1007/s00158-020-02629-w.
[18] B. Van't Hof, J. ten Thije Boonkkamp, R.M.M. Mattheij, Discretization of the stationary convection-diffusion-reaction equation, Numer. Methods Partial Differential Equations 14 (5) (1998) 607–625.
[19] G.J. Kennedy, J.E. Hicken, Improved constraint-aggregation methods, Comput. Methods Appl. Mech. Eng. 289 (2015) 332–354, http://dx.doi.org/10.1016/j.cma.2015.02.017.
[20] F. Wang, B.S. Lazarov, O. Sigmund, On projection methods, convergence and robust formulations in topology optimization, Struct. Multidiscip. Optim. 43 (6) (2011) 767–784, http://dx.doi.org/10.1007/s00158-010-0602-y.
[21] K. Svanberg, The method of moving asymptotes—a new method for structural optimization, Int. J. Numer. Methods Eng. 24 (2) (1987) 359–373, http://dx.doi.org/10.1002/nme.1620240207.
[22] N. Aage, B.S. Lazarov, Parallel framework for topology optimization using the method of moving asymptotes, Struct. Multidiscip. Optim. 47 (4) (2013) 493–505, http://dx.doi.org/10.1007/s00158-012-0869-2.
[23] J.K. Guest, A. Asadpoure, S.-H. Ha, Eliminating beta-continuation from Heaviside projection and density filter algorithms, Struct. Multidiscip. Optim. 44 (4) (2011) 443–453, http://dx.doi.org/10.1007/s00158-011-0676-1.
[24] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (2) (1993) 461–469, http://dx.doi.org/10.1137/0914028.
[25] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc web page, 2019, URL: https://www.mcs.anl.gov/petsc.
[26] O. Dubois, Optimized Schwarz methods with Robin conditions for the advection-diffusion equation, in: Domain Decomposition Methods in Science and Engineering XVI, Springer, 2007, pp. 181–188.
[27] N.M. Evstigneev, Numerical analysis of Krylov multigrid methods for stationary advection-diffusion equation, in: Journal Of Physics: Conference Series, 1391, (1) IOP Publishing, 2019, 012080.
[28] W. Carter, D. Erno, D. Abbott, C. Bruck, G. Wilson, J. Wolfe, G. Finkhousen, A. Tepper, R. Stevens, The GE aircraft engine bracket challenge: an experiment in crowdsourcing for mechanical design concepts, in: Solid Freeform Fabrication Symposium 2014, pp. 1402–1411.
[29] E.A. Träff, O. Sigmund, N. Aage, Topology optimization of ultra high resolution shell structures, Thin-Walled Struct. 160 (2021) 107349, http://dx.doi.org/10.1016/j.tws.2020.107349, URL: http://www.sciencedirect.com/science/article/pii/S026382312031212X.
[30] L.C. Høghøj, E.A. Träff, An advection-diffusion based filter for machinable designs in topology optimization - STL files, 2021, http://dx.doi.org/10.11583/DTU.16930198, URL: https://data.dtu.dk/articles/dataset/An_Advection-Diffusion_based_Filter_for_Machinable_Designs_in_Topology_Optimization_-_STL_files/16930198/0.
[31] GE jet engine bracket challenge webpage, 2014, https://grabcad.com/challenges/ge-jet-engine-bracket-challenge. (Accessed: 04 December 2020).
[32] M. Wallin, N. Ivarsson, O. Amir, D. Tortorelli, Consistent boundary conditions for PDE filter regularization in topology optimization, Struct. Multidiscip. Optim. (2020) http://dx.doi.org/10.1007/s00158-020-02556-w.
[33] J.H. Ferziger, M. Perić, Computational Methods For Fluid Dynamics, Springer Berlin Heidelberg, 2002, http://dx.doi.org/10.1007/978-3-642-56026-2.
[34] C.B. Dilgen, S.B. Dilgen, D.R. Fuhrman, O. Sigmund, B.S. Lazarov, Topology optimization of turbulent flows, Comput. Methods Appl. Mech. Eng. 331 (2018) 363–393, http://dx.doi.org/10.1016/j.cma.2017.11.029.

[P3]

Topology Optimization For Structural Mass Reduction of Direct Drive Electric Machines.

A. C. Hayes, E. A. Träff, C. V. Sørensen, S. V. Willems, N. Aage, O. Sigmund and G. L. Whiting.

Sustainable Energy Technologies and Assessments.

[In review]

Topology Optimization For Structural Mass Reduction of Direct Drive Electric Machines

Austin C. Hayes[a], Erik A. Träff[b], Christian Vestergaard Sørensen[b], Sebastian Vedsø Willems[b], Niels Aage[b], Ole Sigmund[b], Gregory L. Whiting[a]

[a] *University of Colorado Boulder Paul M. Rady Department of Mechanical Engineering, 1111 Engineering Drive, Boulder, 80309, CO, USA*
[b] *Department of Civil and Mechanical Engineering, Technical University of Denmark, Building 404, Koppels Alle, Kgs. Lyngby, 2800, Denmark*

Abstract

Topology optimization is well suited to computationally achieve highly optimized designs with respect to objective and constraint functions, and when coupled with additive manufacturing, enables the fabrication of complex structures that can offer improved generator performance and increased power density. Here, topology optimization was used for light-weighting high power direct drive generators. The rotor of a 5 MW wind turbine generator was analyzed to determine geometric avenues of mass reduction with varying safety factors. Three designs were created with safety factors between 1 and 2 which resulted in structural mass reductions ranging between 54%-67% compared to a baseline design leading to a 13%-25% increase in power density. These designs represent a ∼50% structural mass reduction compared to a previous structural optimization design using triply periodic minimal surfaces. All deflections were less than their critical limits and the topology optimized rotor designs exhibited 64% lower radial deflection than baseline. Deflections of the topology optimized designs were verified with commercial code and supported through experimental validation using digital image correlation with additively manufactured rotors. Computational resource use was reduced as compared to past parameter optimization approaches, and manufacturability studies demonstrated that topology optimization holds potential for significant mass reduction of direct drive generators.

Keywords: Topology Optimization, Direct Drive Generators, Additive Manufacturing, Wind Turbines, Mass Reduction
PACS: 0000, 1111
2000 MSC: 0000, 1111

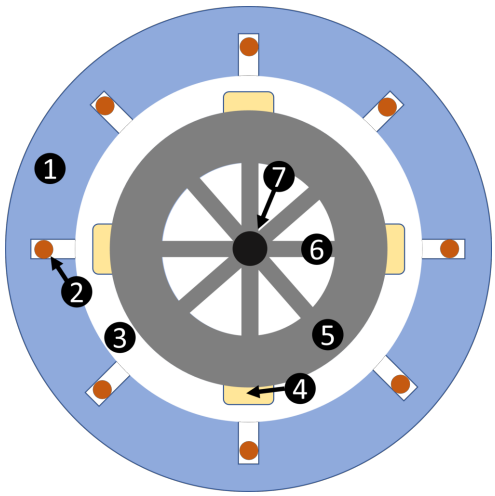
1. Introduction

Electric machines are an integral part of modern society. Electric generators create 98% of the worlds electricity supply [1] and are found in nearly all power generation technologies worldwide. AC generators are composed of a rotor and stator in which a typical configuration utilizes magnet poles on the rotating rotor and copper coils within slots on the stator. AC generators can be classified by phase count (typical 3 phases), rotor arrangement (outer or inner rotor), excitation scheme (AC, DC, brushless, permanent magnet), and generation method (induction, synchronous).

In 2020, power generation comprised 32% of all U.S. greenhouse gas emissions [2] and, in 2019, accounted for 44% of global CO_2 emissions [3]. To address this, low carbon power generation technologies can be used. These include wind, solar, geothermal and nuclear sources. With the exception of solar, each low carbon power generation source previously mentioned requires an electric generator to convert chemical or mechanical energy to electrical.

In 2020, predominantly land based wind turbines provided 7.2% of grid electricity in the United States with projections by the U.S. Energy Information Administration to reach 12.5% of the total electricity supply by 2050 [4]. Similar trends of increasing wind reliance can also be seen in Europe. For example, 47% of Danish electric power was produced by wind in 2019. In U.S. states with large wind resources, (Texas, Idaho, North Dakota), the levelized cost of electricity (LCOE) of wind is lower than competing fossil fuel technologies (petroleum, coal, natural gas). In 2023, the LCOE of a combined cycle gas power plant entering service is 33.21$\$/MWh$ as compared to the onshore wind

*Corresponding author: Gregory.Whiting@colorado.edu



1. Stator
2. Copper Coil
3. Airgap
4. Permanent Magnet
5. Rotor Backiron
6. Structural Support
7. Shaft

Figure 1: Permanent magnet direct drive generator schematic.

LCOE of 30.44 $\$/MWh$ [5]. Rising fossil fuel costs further contribute to more favorable LCOE assessment of low carbon power generation systems. Further advances in wind turbine technologies and generator design hold potential to continually drive down both onshore and offshore wind LCOE. Additionally, advances in grid stabilization with inverter inertia technology [6] and battery storage [7] address several challenges for wind energy, namely, variable generation and no mechanical grid inertia.

Synchronous generators are particularly well suited for wind turbines in order to accommodate variable speed generation. Inverters allow electrical rectification of the generator output to match grid conditions no matter the wind speed [8]. In order to increase efficiency and remove the need for external power, electrically excited rotor field windings can be replaced by permanent magnets. Furthermore, in 2012 researchers at the U.S. National Renewable Energy Lab (NREL) found that the gearbox was the single most expensive component to repair or replace. The mean time before gearbox failure was 5 years and increased to 9 years with optimized gearbox design for more accurate loading conditions [9]. Even a 9 year failure timeframe is undesirable for offshore wind where reliability is paramount; therefore, improved reliability can be achieved through a direct drive coupling system removing the gearbox altogether. A permanent magnet, direct drive generator schematic is seen in Fig. 1. In order to maintain the same power output at lower rotational speeds, the torque must increase. This increase in torque results in significantly higher generator diameters vastly increasing its size, mass, and volume. Furthermore, as much as 90% of the generator mass in a direct drive generator can be inactive mass used solely to support the rotor and stator and prevent closing of the airgap [10]. Therefore, substantial weight savings of the generator can be realized with structural mass optimization.

Analytical design tools developed in the 2000's furthered the understanding of electromagnetic forces and structural deflection of a generator under load resulting in designs such as a spoked arm and solid disk design in the rotor and stator [11, 12, 13]. These analytical equations were used with a genetic algorithm optimization approach in 2013 by Zavos *et al.* in order to select for optimal cost, mass, and deflection constraints, resulting in a 30% reduction in total generator mass through parameter optimization alone [14]. Attempts to account for loading rotor eccentricity and air gap closure resulted in new designs with more complex structures such as bridge chamfers and elliptical holes [15]. With increasing geometric complexity, researchers turned to finite element analysis for design iteration. Pa-

rameter optimization using triply periodic minimal surfaces for the rotor of a 5 MW permanent magnet direct drive (PMDD) generator resulted in a 34% reduction in structural mass [16] compared to the baseline design [17] (Fig. 2). Initial topology optimization attempts on the rotor and stator was performed using commercial code [18, 19], resulting in mass reduction. However, challenges to using commercial code included non-symmetric designs, lack of clearly defined structures, and lack of multiple length scale structures for improved stiffness. Non-symmetric results would result in centrifugal forces and eccentricity during operation and creation of large holes instead of lattice type supports suggests a non optimized numerical scheme. Fig. 2 shows the progression of rotor topologies using different optimization methods.

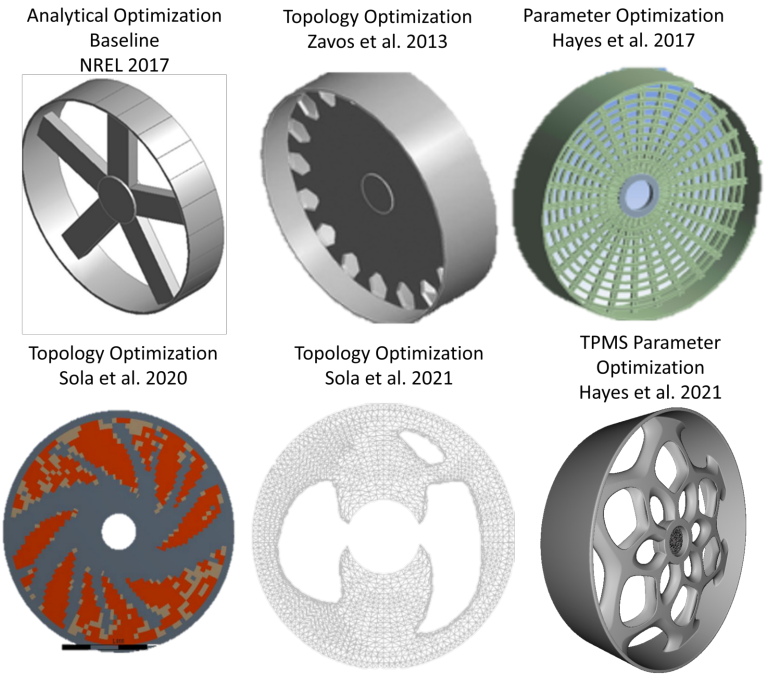
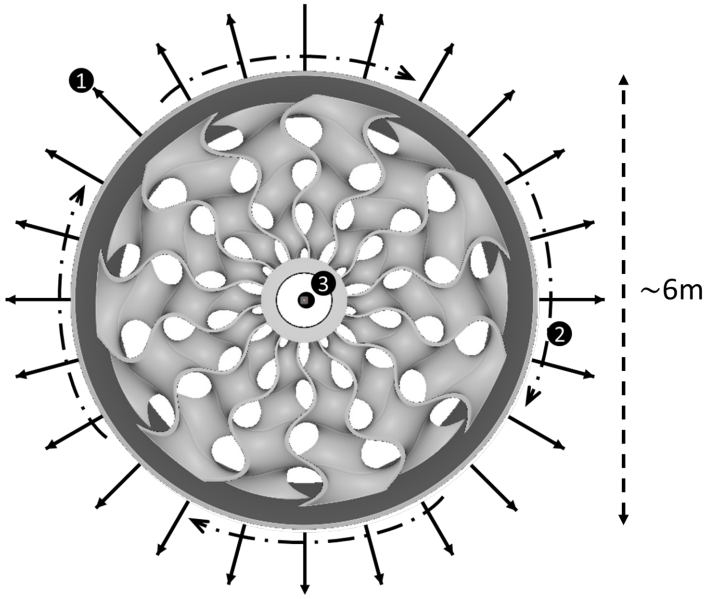


Figure 2: Rotor designs in literature include a 1D model genetic algorithm approach as the baseline design (NREL 2017), initial efforts at using topology optimization in commercial codes to reduce structure mass (Zavos et al. 2013, Sola et al. 2020, Sola et al. 2021), parameter optimization of more complex analytically defined geometries (Hayes et al. 2017), and the first use of parameter optimization of TPMS structures for lightweighting direct drive generators (Hayes et al. 2021)

Topology optimization allows for high design freedom by determining what elements should be empty or solid to satisfy a given objective function subject to constraints. The optimization technique can be used for almost all governing physics or multiphysics application however much research has focused on structural compliance problems [20, 21]. In these problems, material is added only where needed which allows for significant lightweighting [22]. Previous attempts at lightweighting direct drive generators with topology optimization utilized generic commercial code solvers. Several challenges existed with commercial code solutions including asymmetry, local minima, and lack of feature resolution. Obtaining a functional design for large direct drive generators using topology optimization requires small feature sizes and thus discretization with tens or even hundreds of million elements. Efficient computational infrastructure catered to the problem at hand is important in producing viable results. Past work by Baandrup *et al.* and Aage *et al.* have employed gigavoxel topology optimization on large structures resulting in a 28% mass reduction in girder weight for a suspension bridge [23] and 2-5% reduction in mass of a full NASA common research model aeroplane wing [24].

This study presents density based volume minimization for linear elasticity as a means of generating lightweight lattice designs for the rotor of a 5 MW direct drive wind turbine generator. Three rotor designs with safety factors between 1-2 were created using the topology optimization framework and deflections validated with a commercial code and digital image correlation of a printed part. Deflections were under the critical deformation limit in all directions and structural mass reductions between 54-67% were found, suggesting a strong use case for topology optimized rotor designs. This work exceeds the previous structural mass reduction scheme for a 5 MW direct drive generator by 50% and to the authors' knowledge is the current best structural mass savings approach to lightweighting direct drive generators.



Loading

1. Normal Maxwell Stress: 0.2 MPa
2. Shear Stress: 40 kPa
3. Gravitational Loading: 9.81 m/s^2

Critical Deflections

1. Radial < 0.65 mm
2. Torsional < 2.84 mm
3. Axial < 32.17 mm

Figure 3: Rotor loading conditions and critical deflections

2. Materials and Methods

2.1. Rotor Loading Criteria

The low speed, high torque PMDD generator has three main loading criteria seen in Fig. 3. First, the normal component of the Maxwell stress acts to close the airgap. Second, the tangential component of the Maxwell stress opposes the torque of the shaft resulting in power transformation. Thirdly, gravity acts as a body force on the entire structure seen mostly during transportation. Loading conditions in this study agree with those used in literature for

a 5 MW radial flux PMDD generator [12]. The normal component of the Maxwell stress was 0.2 MPA, the shear stress resulting from the tangential component of the Maxwell stress was 40 kPa, and gravietational acceleration was 9.81 m/s^2. The critical deflection criteria was a radial deflection less than 10% of the air gap diameter, torsional deflection less than 0.05° angle of twist, and an axial deflection less than 2% of the axial length. For the machine topology studied, this results in a critical radial, torsional, and axial deflection of 0.65 mm, 2.84 mm, and 32.17 mm respectively.

2.2. Topology Optimization Approach

The topology optimization problem (discussed in detail in the Theory section) was solved for torsional deflection safety factors of 1 and 2 (ie: for the nominal load and twice the nominal load with varying initial conditions). In the literature, typically a safety factor of 1 is used as this value represents the accepted limits in the design of the generator [12, 17]. In this study, the impact of doubling the safety factor on the final design was explored. This study suggests that mass reduction could still be made even when defining a more stringent set of deflection constraints. The three resulting designs can be found in Fig. 4 and were used for printing the experimental parts as well as validation with commercial code.

2.3. Mesh Independence and Commercial Code Validation

A mesh independence study was performed by evaluating the three topology optimized designs at three different mesh sizes (10, 20, and 30 mm elements) and comparing deflection results. A difference of less than 5% between the two finest meshes constitutes finite element convergence. The mesh file for each design was imported into commercial code package Abaqus for FEA validation. In this study, a simple static analysis was performed and the developed code compared against the commercial code FEA results.

2.4. Experimental Validation

In order to experimentally validate the topology optimized designs, a representative scaled model was printed in polylactic acid (PLA) using fused deposition modeling (FDM). The model was printed solid with 100% infill with a Raise 3D dual extrusion printer configured with PVA dissolvable supports (Raise 3D, Pro2 series, Irvine, CA, USA). Since low deformation existed in the 5 MW loading conditions, we assumed the material is well below its yield strength, solely elastic deformation occurs with no non-linearities, and thus the linear Hooke's law applies. Therefore, the topology optimized rotor designs can be simulated and verified with a different material than structural steel for ease of manufacturing while still providing validation of the FEA process. In this study, the 5 MW rotor design was scaled to 3% to a radius of 100 mm. Due to the difficulty of applying a continuous radially outward load and shear stress along the circumference of the rotor, for experimental validation, a 60 mm torque arm was added to the part. Additionally, a square shaft was added to the design to facilitate holding the part in a vice. The torque arm and shaft allowed for easy and accurate loading enabling the creation of a digital twin in Abaqus for FEA analysis. Since torsional deformation was found to be dominant in the 5 MW rotor designs, ability to resolve the torque arm deflections in the printed model was used to validate the ability to resolve the 5 MW rotor deflections. For experimental validation, a dead weight test was used to experimentally apply a highly accurate torque to the rotor. For this study, a 0.63628 kg deadweight was used leading to an applied force of 6.25 N to the torque arm. Digital image correlation (DIC) was employed using a 24.2 MP Rebel T7i camera (Canon, Melville, NY). A speckle pattern was first applied to the part with flat black spraypaint. Images were taken before and after hanging a deadweight from the torque arm and the opensource digital image correlation software DICe by Sandia National Labs used for analysis [25]. The digital twin was simulated in Abaqus in PLA with a density, elastic modulus and Poisson's ratio of 1200 kg/m^3, 2306 MPa, and 0.35 respectively.

3. Theory

The applied topology optimization formulation is a density based volume minimization for linear elasticity with multiple displacement constraints [26]. It also uses the robust formulation to ensure a minimal length-scale [27].

For the density based topology optimization, a finite element model of the rotor design domain is generated. Rings of solid material are prescribed on the inner and outer parts of the rotor, with a thickness of 64 mm and 66.37 mm respectively. The remainder of the rotor is also discretized using finite elements, but here every element is given a design, or density, variable, which indicates whether the element is solid or void. All of the element-wise design

variables are collected in a vector denoted x, while the subscript x_e is used to denote the design variable of a specific element e.

3.1. Filter and Regularization

In order to impose a minimum length-scale the modified robust formulation for linear elasticity is applied [27]. This approach allows for control of the final length-scale, without incurring the usual high computational cost of the robust formulation, by exploiting properties of the linear elasticity and volume constraint.

The first step in the robust approach consists of a convolution type filtering of the design variables. Besides being the basis for the robust approach, this is also needed in order to regularize the otherwise ill-posed optimization problem such that checkerboards and mesh dependent designs are avoided [26]. The filtered field $\tilde{x}$ is found by solving a modified Helmholtz equation, which uses the design variables as the forcing term [28]. The Robin boundary conditions are used on the free boundaries to remove the boundary effects on the domain [29]. The usual Neumann boundary condition is used for boundaries between the design domain and the inner and outer rings. The equation solved is

$$-r^2\nabla^2\tilde{\mathbf{x}} + \tilde{\mathbf{x}} = \mathbf{x} \text{ in } \Omega \tag{1}$$

Where Ω is used to denote the design domain. Here r denotes the term used to control the filter size in the equation, which should be set to $r = 2\sqrt{3}R$ to obtain the physical filtering radius R. The filter equations are solved using a first order finite volume discretization. This is chosen to avoid any interpolation, as the density values are constant on the element level in the finite element analysis.

The robust formulation builds on the filtered design field and considers three different design realisations, nominal, dilated, and eroded. In the original formulation of the robust approach, the objective and all constraints are evaluated on all design evaluations, and the worst-case is always chosen from the three. This ensures that the final design is robust with respect to dilation and erosion of the structure. The modified robust formulation used shows that more material will always lead to lower deformation and higher volume. Therefore in this work, the deformation is only evaluated on the eroded design, as this is known a-priori to be the worst-case. Likewise, the volume is only evaluated on the dilated design.

In order to create the different designs a differentiable approximation to the Heaviside function is used:

$$\bar{x}_e = \frac{\tanh\beta\eta + \tanh\beta(\tilde{x_e} - \eta)}{\tanh\beta\eta + \tanh\beta(1-\eta)}, \quad \forall e \in \{1, .., n\}. \tag{2}$$

Here η controls the point at where the design is thresholded, and β controls the sharpness of the threshold. The used values of η are 0.5 for the nominal design $\bar{\mathbf{x}}$, 0.8 for the eroded design $\hat{\mathbf{x}}$, and 0.2 for the dilated design $\hat{\hat{\mathbf{x}}}$. These settings ensure a minimal feature size of 64 mm when a filter radius of 40 mm is used. The value of β is increased slowly throughout the design iterations, in a continuation scheme. For this work, the scheme increases β slowly in the beginning, and faster towards the end. The full scheme is given in table A.1.

3.2. Stiffness interpolation

In this work the standard Solid Isotropic Material with Penalization method (SIMP) interpolation is applied to interpolate the Young's modulus as a function of the evolving density field $\bar{x}$ [26]. The stiffness interpolation for element e is thus

$$E_e = E_{\min} + \hat{\mathbf{x}}_e^p(E_0 - E_{\min}), \quad E_{\min} = 10^{-6}E_0 \tag{3}$$

Where E_0 is the Young's modulus of the solid material, and p is the penalisation variable, which controls how hard intermediate densities are penalised. For this work a penalisation of $p = 3$ is used.

3.3. Deformation constraints

In order to ensure that the outer surface of the rotor retains its original geometry, two deformation constraints are included in the optimization problem. From section 2.1, only the radial and torsional (1) and (2) from fig. 3) load-cases are considered. This is deemed reasonable as it was found that the axial deformation constraint was fulfilled by all designs. The two remaining load cases are treated individually, this is, to compute the radial deformation constraint,

only the normal Maxwell stress is applied to the structure, and to compute the torsional deformation constraint, only the shear stress is applied.

In order to compute the constraint value, firstly the node-based directional deflections are found for every node on the outer surface, by taking the inner product between the nodal deformation, and the radial or torsional unit direction vector for the corresponding node. These values collected in vectors are denoted D_{radial} and $D_{\text{torsional}}$ respectively.

Secondly, in order to reduce the number of constraints, the directional deformations are aggregated using the p-mean function, which is a differentiable approximation to the maximum function, using the aggregation parameter $p^* = 16$.

$$P_m(\mathbf{v}) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_i^{p^*} \right)^{\frac{1}{p^*}} \tag{4}$$

Hence a differentiable scalar approximation to the maximal deflection is obtained. While there is some error compared to the true maximal deformation, this is mitigated by updating the used constraint value for the optimization problem every 20 design iterations. The update scales the constraint value, in order to correct for the error in the p-mean approximation. Additionally, the deformation constraints are scaled by a safety factor s_f.

3.4. Optimization problem

Now the full optimization problem is presented, implicitly using the definitions from previous sections. The volume is minimised, subject to the linear elastic state equations, the deformation constraints, and a box constraint for each design variable.

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} && \frac{1}{\sum_{e=1}^{n_e} v_e} \sum_{e=1}^{n_e} v_e \hat{\hat{\mathbf{x}}}_e \\
&\text{subject to:} && \\
&\text{state equation} && \mathbf{K}(\hat{\mathbf{x}})\,\mathbf{u_i} = \mathbf{f}_i, \quad \forall i \in \{1,2\} \\
&\text{radial deformation} && P_m(D_{\text{radial}}(u_1)) \leq \frac{0.65\text{mm}}{s_f} \\
&\text{torsional deformation} && P_m(D_{\text{torsional}}(u_2)) \leq \frac{2.84\text{mm}}{s_f} \\
&\text{box constraint} && 0 \leq \mathbf{x}_e \leq 1, \quad \forall e \in \{1,..,n_e\}
\end{aligned} \tag{5}$$

The problem is solved using the method of moving asymptotes [30]. No explicit symmetry constraint was used for the optimization formulation. Six-way symmetric designs are obtained as the underlying finite element mesh was six-way symmetric, and the elasticity equations were solved with high precision.

4. Results and Discussion

4.1. Optimized Rotor Designs

Three topology optimized (TO) designs were created by varying the desired torsional safety factor from 1-2 and can be seen in Fig. 4. TO design 1 (Fig. 4a) and TO design 2 (Fig. 4b) were created with a safety factor of ∼2 but started using different initial conditions. Design 1 used a homogeneous density distribution of 0.25 whereas design 2 used 0.6. As topology optimization is a non-convex optimization problem, there is no guarantee that the found minima are global. This implies that different designs can be obtained by varying the initial design of a problem formulation, as seen in the difference between designs 1 and 2. The presented designs were selected from a set of trials, where a range of initial homogeneous density distributions were considered. Since topology optimization problems are inherently non-convex, different starting guesses may lead to different local optima. In this work, all obtained designs satisfy constraints and hence the different local minima provide the design engineer with different design options and the final design may thus be chosen due to other criteria like manufacturing costs, handability or aesthetics. A flowchart depicting the design process is given in Fig. 5.

TO designs 1 and 2 exhibited radial deflections of 0.26 mm and 0.19 mm respectively and torsional deflections of 1.41 mm and 1.43 mm respectively. TO design 3 (Fig. 4c) was created with a safety factor of ∼1 and an initial density distribution of 0.15 and resulted in a radial deflection of 0.20 mm and torsional deflection of 2.62 mm. All torsional

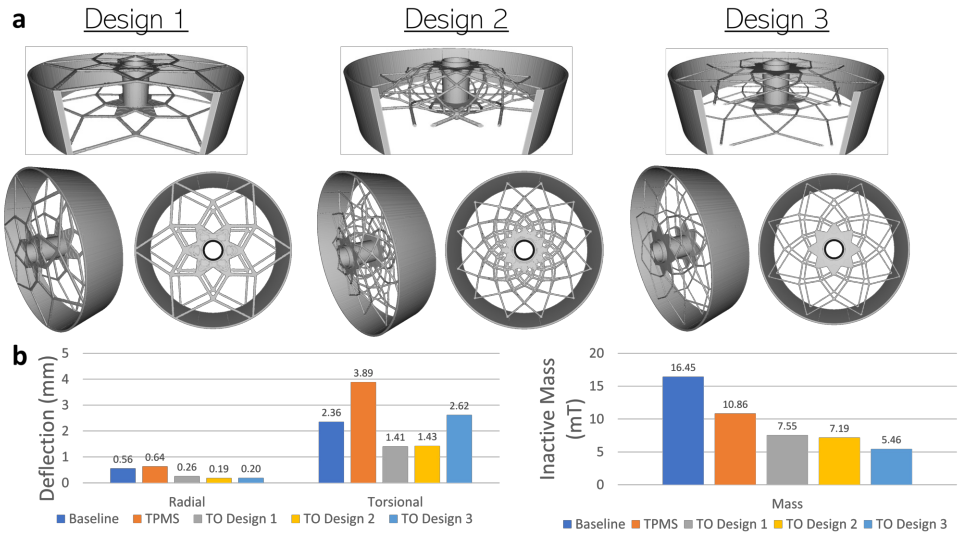


Figure 4: (a) TO rotor design perspective and cross sectional views (b) FEA radial and torsional deflections and inactive mass compared to baseline and previous designs

deflections were below the critical limit of 2.84 mm and radial deflections were well below the critical limit of 0.65 mm. The significant difference in torsional deflections between designs 1 and 2 and design 3 was due to the use of different safety factors. The TO designs had radial deflections 64% lower than the baseline spoked arm design [17]. This is important as it leads to greater airgap stability and lower eccentricity during rotor operation. All designs had axial deflections less than or equal to 0.813 mm which was less than 3% of the axial critical limit.

The rotor designs are shown in Fig. 4a-c. TO designs 1, 2, and 3 depicted radially outward cylindrical connected spokes with a thickened inner ring near the stress concentration at the shaft constraint. Design 1 and 3 depicted two planes of spokes along the axial axis close to the axial faces of the rotor backiron. Design 1 had spoke diameters of ~84 mm while design 3 had thinner but more numerous spokes of ~50 mm with greater interconnectivity. Design 2 showcased similar deflection and mass to design 1, yet is the only design with self connecting spokes in the axial direction showcasing a plane of symmetry in spoke geometry about the axial midplane. In this design, the spokes extended from the axial edges near the backiron faces in a convex and concave direction before reconnecting from either side with spokes that attach to the rotor backiron along the axial midplane. This was the most complex design out of the three yet resulted in fewer attachment points to the rotor backiron due to its self interconnectivity.

The inactive mass or structural mass is comprised of non-magnetically active regions of the rotor. As the backiron is magnetically active, the inactive mass comprises the structural support material which prevents airgap closure and allows connection to the drivetrain shaft. The inactive mass of the topology optimized designs was markedly lower than the baseline design. The lowest mass TO design was design 3 with a factor of safety in the torsional direction slightly above 1 and resulted in an inactive mass of 5.46 mT (1 mT = 1000 kg). This was 50% lower than the past best triply periodic minimal surface lattice design [16] and remarkably, 67% lower than the baseline design [17]. TO design 3 resulted in a total mass reduction of 11 mT of structural steel compared to baseline. The other two designs offer a higher factor of safety in the torsional direction with the sacrifice of added mass. TO designs 1 and 2 resulted in inactive masses of 7.55 mT and 7.19 mT respectively which were 54% and 56% less than the baseline spoked arm design respectively.

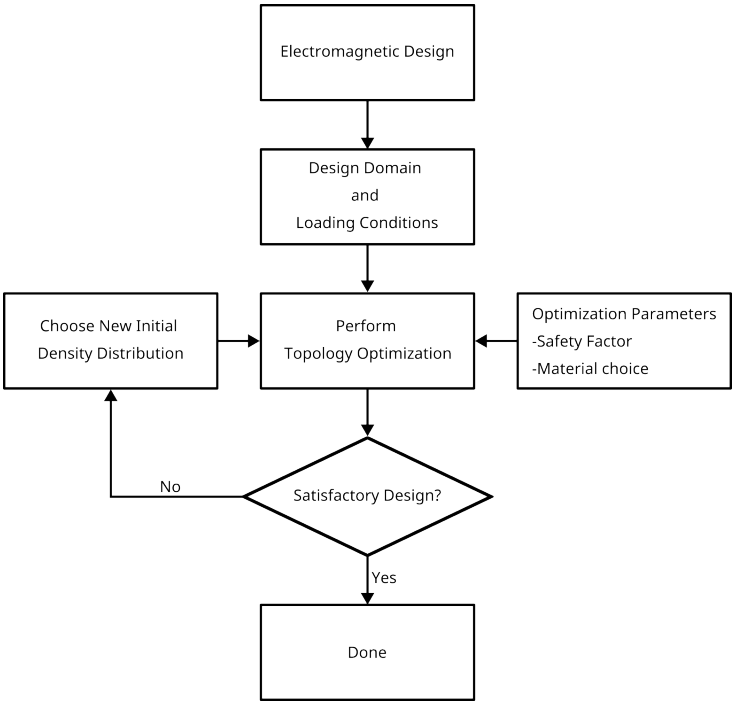


Figure 5: Optimization scheme flowchart

4.2. Mesh Independence and Commercial Code Validation

The voxel based topology optimized designs where subsequently converted into STL-files which were then used for generating body fitted meshes. Each topology optimized design mesh was refined three times in order to determine the influence of mesh size on the results. As design complexity increased or feature size decreased, a more refined mesh was needed to properly resolve the design. For TO designs 1 and 2, mesh refinement resulted in a less than 2.5% change in torsional and radial deflection suggesting proper mesh refinement at the initial level. Design 3 featured spoke sizes roughly half that of designs 1 and 2 resulting in greater mesh resolution necessary to resolve all features. Further mesh refinement for design 3 beyond this point resulted in a less than 1% change in deflections suggesting proper element size. Commercial code validation using Abaqus against the density representation in the Portable, Extensible Toolkit for Scientific Computation (PETSc) based code resulted in a deviation of less than 1.6% for all three designs for the radial deflection and torsional deviations for designs 1, 2, and 3 of 9.8%, 15.8%, and 8.6% respectively. This discrepancy is in part due to the jagged nature of the density based geometry representation in the PETSc based code. Nevertheless, mesh refinement and agreement with commercial code supports the developed code's FEA capabilities on the rotor structures.

4.3. Computational Resources

Each topology optimized design was computed using 13.2 million hexahedral finite elements on the DTU Sophia cluster [31] using 10 compute nodes, each with two 16-core AMD EPYC 7351 CPUs, for a total of 320 CPU cores. Each design ran between 14 and 20 hours, resulting in 4480 to 6400 core hours used for each design when utilizing an iterative solver setup similar to that presented in [32]. However, it should be noted that several iterations were needed

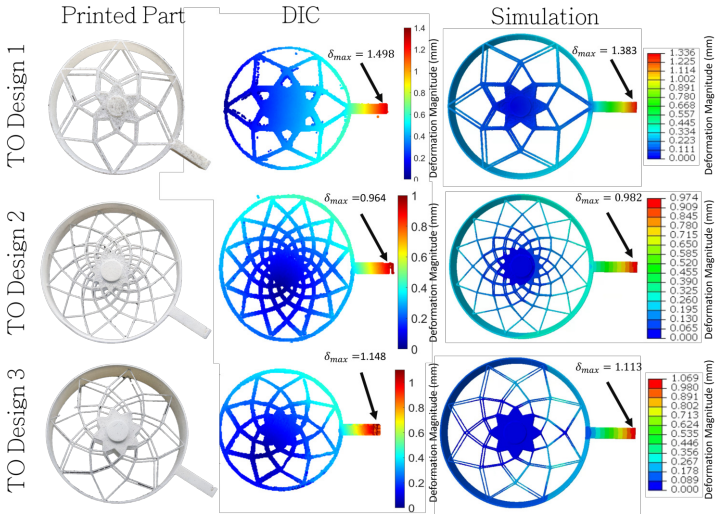


Figure 6: Experimental and simulation comparison for torque arm addition

to find the best parameters for computing the designs, where each iteration required several runs on the cluster. The previous lowest mass parameter optimization algorithm in literature for lightweighting permanent magnet direct drive rotors utilized a genetic algorithm with triply periodic minimal surfaces. This past work required ~ 8500 core hours per design [16]. However, it should be noted that the genetic algorithm and triply periodic minimal surface lattice generation technique required serial computing between generations due to the result dependence on concurrent computations resulting in limited opportunities for parallel computing speedup. Therefore, the true run time was much greater than the computation resource cost alone would suggest. Therefore, to the authors' knowledge, this study represents the largest geometric weight savings (~ 67% to baseline) with the least computational cost for the rotor of a PMDD generator.

4.4. Experimental Validation

The deformation magnitude between the printed torque arm designs and Abaqus FEA showed good agreement suggesting high ability of the FEA code in resolving the rotor geometries. Images of the printed parts, DIC results, and simulation results with the concurrent color maps can be seen in Fig. 6. The torque arm extension with design 1 had the highest error with 8.3% deviation between FEA and digital image correlation. Despite having the highest complexity of the three designs, design 2 depicted the lowest deviation within 1.8% of the FEA result. The average deviation between DIC and FEA analysis was 4.4%. The ability of the printer to resolve the small cylindrical spokes most likely accounts for the deviation in results. For example, a minimum feature size of 3 mm seen in design 1 and 2 results in 7.5 layer widths compared to 4 layer widths for the 1.6 mm feature sizes of design 3. This results in greater error between the actual printed geometry and the mesh file used for FEA analysis. Nevertheless, good agreement existed between the torque arm loading for both the printed PLA parts and digital twin FEA analysis suggesting good ability of the FEA tools in accurately capturing the deflection in the designs with varied material properties for the 5 MW design.

4.5. Manufacturability for Large Machines

Topology optimization coupled to additive manufacturing techniques excels at both designing and manufacturing highly complex parts. However, additive manufacturing techniques such as powder bed fusion, selective laser melting, or electron beam melting are typically high cost [33] and limited to parts less than 1 m [34]. This adds complexity to

manufacturing these topology optimized rotor designs as traditional additive manufacturing approaches prove difficult and expensive. One potential solution is hybrid additive manufacturing using powder binder jetting of a mold and traditional sand casting. This process has been explored in the past for large, complex generator support structures [35]. In this process, the sand mold of the support structure can be printed using large format binder jetting machines capable of printing sizes up to 4 m [36]. Since the mold is printed, it can create the complex spoke patterns of the topology optimized designs. Next, traditional casting can be used to create the metal topology optimized rotor support designs. Using traditional casting decreases cost while maintaining high complexity associated with additive manufacturing. Note that only the support structures need to be manufactured as rotor electromagnetically active regions can be produced by traditional stamping or lasercutting processes. These rotor support designs post-cast can then be welded at the spoke intersection points to the rotor backiron as traditionally done for rotor support structures. Each topology optimized design has different manufacturing challenges. Designs 1 and 3 have greater weld points due to two planes of spokes, yet they are symmetric allowing for significantly lower height molds since the support part can be printed as two separate parts resulting in higher production. Design 2, with less spokes connecting to the rotor backiron, would reduce the final welding needs but require larger molds with lower productivity due to its interconnective, axial-varying pattern.

4.6. Influence of Lightweighting on Power Density

Increased power density of electric machines can be accomplished by increasing the total power output of the machine or decreasing the total mass of the machine. Topology optimization as a means of lightweighting the rotor enabled up to a 25% increase in power density as compared to the baseline design. This increase in power density was solely due to geometric design and further improvements in power density could be made by applying the topology optimization method to the stator for complete generator structural optimization. Additionally, by reducing the mass of the generator, the nacelle mass can be decreased resulting in lower downstream costs as the tower can be designed to withstand reduced load. Furthermore, this topology optimization approach at reducing the structural mass of electric machines can be applied to motors. In this way, improved power density via structural mass topology optimization holds promise for electric machines across a variety of industries. In the power generation sector with objectives of greatest efficiency at lowest cost, improved power density by lower structural mass reduces the generator material cost and downstream costs associated with supporting the generator mass. In the electric motor propulsion sector, improved power density enables higher thrust to weight ratios for electric aviation and greater fuel economy due to lower vehicle mass for automotive applications.

5. Conclusions

Structural designs for the rotor of a direct drive wind turbine generator was performed using large scale topology optimization methods and high performance computing. Three designs were created using a safety factor of 1 and 2 and varying initial configurations. All designs showed a solid star disc structure near the shaft connection and cylindrical spokes radially outward. The topology optimized designs were within the radial, torsional, and axial critical limits. Large structural mass reduction was achieved through these designs with the lightest design reducing the rotor structural mass by 67% compared to the baseline design. This translates to an increase in power density of the 5 MW reference generator by up to 25%. These designs represent a 50% improvement over the current best lattice optimized support structures. This work provides a framework for creating custom topology optimized electric machine support structures, and can be extended to the stator for even greater mass reduction. Recommendations for future work include enabling a generalized rotor and stator structural optimization input allowing structural coupling to electromagnetic optimization. This would allow greater force densities to be realized with lower deflections. Additionally, extending the application to electric motors holds potential for improving their power density. Reduced top mass in the nacelle of wind turbines offers downstream cost savings due to lower structural support needs. The results of this study also holds promise for other electric machines, such as enabling increased power density (kW/kg) vital for electric propulsion industries.

6. CRediT Author Statement

Austin Hayes: Conceptualization, Methodology, Validation, Formal Analysis, Writing, Visualization **Erik Träff:** Methodology, Software, Formal Analysis, Investigation, Writing **Christian Vestergaard Sørensen:** Software, Investigation, Methodology **Sebastian Vedsø Willems:** Software, Investigation, Methodology **Niels Aage:** Software,

Resources, Supervision, Project administration **Ole Sigmund:** Software, Resources, Supervision, Project administration, Funding acquisition **Gregory Whiting:** Resources, Supervision, Project administration, Funding acquisition

7. Acknowledgement

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. (DGE 1650115), as well as startup funds provided by the University of Colorado Boulder. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors gratefully acknowledge the computational and data resources provided on the Sophia HPC Cluster at the Technical University of Denmark, DOI: 10.57940/FAFC-6M81.

Support from the Villum Foundation to ET, NAA and OS through the Villum Investigator project InnoTop is gratefully acknowledged.

8. Data Availability

Data will be provided upon request.

Appendix A. Continuation scheme for projection sharpness β

See table A.1 for the sharpness continuation scheme.

iteration	β
0	0.01
40	1
80	2
120	3
150	4
180	5
210	6
240	7
270	8
300	9.6
330	11.5
360	13.8
390	16.5
420	19.9
450	23.9
480	28.7
510	34.4
540	41.3
570	49.5
600	59.4

Table A.1: The continuation scheme for the projection sharpness. At the given iteration, the sharpness is set to the given value.

References

[1] International Energy Association, Key World Energy Statistics (2021).
[2] US EIA, Monthly Energy Review, Environment (2021).
[3] International Energy Association, Greenhouse Gas Emissions from Energy: Overview (2021).
[4] US EIA, U.S. Energy Information Administration Annual Energy Outlook 2020 (2020).
[5] US EIA, U.S. Energy Information Administration Annual Energy Outlook 2021 (2021).
[6] K. Y. Yap, J. M. Lim, Virtual inertia-based inverters for mitigating frequency instability in grid-connected renewable energy system: A review, Applied Sciences 9 (24) (2019) 5300.
[7] M. Hannan, S. Wali, P. Ker, M. A. Rahman, M. Mansor, V. Ramachandaramurthy, K. Muttaqi, T. Mahlia, Z. Dong, Battery energy-storage system: A review of technologies, optimization objectives, constraints, approaches, and outstanding issues, Journal of Energy Storage 42 (2021) 103023. doi:10.1016/j.est.2021.103023.
[8] S. Liao, Y. Chen, M. Huang, X. Fu, X. Zha, L. Wang, A. Luo, J. M. Guerrero, Clustering-based modeling and interaction analysis of multiple differently parameterized grid-side inverters in pmsg wind turbines, IEEE Transactions on Energy Conversion 36 (4) (2021) 3031–3043. doi:10.1109/TEC.2021.3071155.
[9] J. Keller, Y. Guo, Planetary load sharing in three-point mounted wind turbine gearboxes: A design and test comparison, Tech. rep. (2017).
[10] M. Mueller, A. McDonald, D. Macpherson, Structural analysis of low-speed axial-flux permanent-magnet machines, IEE Proceedings-Electric Power Applications 152 (6) (2005) 1417–1426.
[11] H. Polinder, F. van der Pijl, G.-J. de Vilder, P. Tavner, Comparison of direct-drive and geared generator concepts for wind turbines, IEEE Transactions on Energy Conversion 21 (3) (2006) 725–733. doi:10.1109/TEC.2006.875476.
[12] A. McDonald, Structural analysis of low speed, high torque electrical generators for direct drive renewable energy converters, PhD Thesis, University of Edinburgh (2008).
[13] D. je BANG, Design of transverse flux permanent magnet machines for large direct-drive wind turbines, PhD Thesis, Pukyong National University (2010).
[14] A. Zavvos, A. McDonald, M. Mueller, Optimisation tools for large permanent magnet generators for direct drive wind turbines, IET Renewable Power Generation 7 (2) (2013) 163–171.
[15] J. Ma, R. Qu, J. Li, S. Jia, Structural optimization of a permanent-magnet direct-drive generator considering eccentric electromagnetic force, IEEE transactions on magnetics 51 (3) (2015) 1–4.
[16] A. C. Hayes, G. L. Whiting, Reducing the structural mass of large direct drive wind turbine generators through triply periodic minimal surfaces enabled by hybrid additive manufacturing, Clean Technologies 3 (1) (2021) 227–242. doi:10.3390/cleantechnol3010013.
[17] L. Sethuraman and K. Dykes, Generatorse: A sizing tool for variable-speed wind turbine generators, Technical report NREL/TP -5000-66462, National Renewable Energy Laboratory, https://www.nrel.gov/docs/fy17osti/66462.pdf (September, 2017).
[18] P. Jaen-Sola, A. S. M. Donald, E. Oterkus, Design of direct-drive wind turbine electrical generator structures using topology optimization techniques, Journal of physics. Conference series 1618 (5) (2020) 52009.
[19] P. Jaen-Sola, E. Oterkus, A. S. McDonald, Parametric lightweight design of a direct-drive wind turbine electrical generator supporting structure for minimising dynamic response, Ships and Offshore Structures 16 (sup1) (2021) 266–274. doi:10.1080/17445302.2021.1927356.
[20] G. Dong, Y. Tang, D. Li, Y. F. Zhao, Design and optimization of solid lattice hybrid structures fabricated by additive manufacturing, Additive Manufacturing 33 (2020) 101116. doi:10.1016/j.addma.2020.101116.
[21] J. Wu, O. Sigmund, J. P. Groen, Topology optimization of multi-scale structures: a review, Structural and Multidisciplinary Optimization 63 (3) (2021) 1455–1480.
[22] J. Plocher, A. Panesar, Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures, Materials and Design 183 (2019) 108164. doi:10.1016/j.matdes.2019.108164.
[23] M. Baandrup, O. Sigmund, H. Polk, N. Aage, Closing the gap towards super-long suspension bridges using computational morphogenesis, Nature communications 11 (1) (2020) 1–7.
[24] N. Aage, E. Andreassen, B. S. Lazarov, O. Sigmund, Giga-voxel computational morphogenesis for structural design, Nature 550 (7674) (2017) 84–86.
[25] D. Turner, Digital Image Correlation Engine (DICe) Reference Manual. SAND2015-10606 (2015).
[26] M. P. Bendsoe, O. Sigmund, Topology optimization: theory, methods, and applications, Springer Science & Business Media, 2003.
[27] B. S. Lazarov, F. Wang, O. Sigmund, Length scale and manufacturability in density-based topology optimization, Archive of Applied Mechanics 86 (1-2) (2016) 189–218. doi:10.1007/s00419-015-1106-4.
[28] B. S. Lazarov, O. Sigmund, Filters in topology optimization based on helmholtz-type differential equations, International Journal for Numerical Methods in Engineering 86 (6) (2011) 765–781. doi:10.1002/nme.3072.
[29] M. Wallin, N. Ivarsson, O. Amir, D. Tortorelli, Consistent boundary conditions for pde filter regularization in topology optimization, Structural and Multidisciplinary Optimization 62 (3) (2020) 1299–1311. doi:10.1007/s00158-020-02556-w.
[30] K. Svanberg, The method of moving asymptotes—a new method for structural optimization, Int. J. Numer. Methods Eng. 24 (2) (1987) 359–373. doi:10.1002/nme.1620240207.
[31] DTU Computing Center, DTU Computing Center resources (2022). doi:10.48714/DTU.HPC.0001.
URL `https://doi.org/10.48714/DTU.HPC.0001`
[32] G. A. da Silva, N. Aage, A. T. Beck, O. Sigmund, Three-dimensional manufacturing tolerant topology optimization with hundreds of millions of local stress constraints, International Journal for Numerical Methods in Engineering 122 (2) (2021) 548–578.
[33] B. Westerweel, R. J. Basten, G.-J. van Houtum, Traditional or additive manufacturing? assessing component design options through lifecycle cost analysis, European Journal of Operational Research 270 (2) (2018) 570–585. doi:10.1016/j.ejor.2018.04.015.
[34] T. Lehmann, D. Rose, E. Ranjbar, M. Ghasri-Khouzani, M. Tavakoli, H. Henein, T. Wolfe, A. Jawad Qureshi, Large-scale metal additive manufacturing: a holistic review of the state of the art and challenges, International materials reviews 67 (4) (2021) 410–459.
[35] A. C. Hayes, G. L. Whiting, Powder-binder jetting large-scale, metal direct-drive generators: Selecting the powder, binder, and process

parameters ASME 2019 Power Conference (07 2019). doi:10.1115/POWER2019-1853.
[36] Voxeljet, 3D Printing Systems: Profitability Included, https://www.voxeljet.com. Accessed December 2017.

[P4]

Simple and Efficient GPU accelerated Topology Optimisation: codes and applications.

E. A. Träff, A. Rydahl, S. Karlsson, O. Sigmund and N. Aage.

Computer Methods in Applied Mechanics and Engineering.

[Accepted]

Simple and Efficient GPU accelerated Topology Optimisation: codes and applications

Erik A. Träff[a], Anton Rydahl[b], Sven Karlsson[b], Ole Sigmund[a], Niels Aage[a]

[a] *Department of Civil and Mechanical Engineering, Technical University of Denmark, Building 404, Koppels Alle, Kgs. Lyngby, 2800, Denmark*

[b] *Department of Applied Mathematics and Computer Science, Technical University of Denmark, Building 324, Richard Petersens Plads, Kgs. Lyngby, 2800, Denmark*

Abstract

This work presents topology optimisation implementations for linear elastic compliance minimisation in three dimensions, accelerated using Graphics Processing Units (GPUs). Three different open-source implementations are presented for linear problems. Two implementations use GPU acceleration, based on either OpenMP 4.5 or the Futhark language to implement the hardware acceleration. Both GPU implementations are based on high level GPU frameworks, and hence, avoid the need for expertise knowledge of e.g. CUDA or OpenCL. The third implementation is a vectorised and multi-threaded CPU code, which is included for reference purposes. It is shown that both GPU accelerated codes are able to solve large-scale topology optimisation problems with 65.5 million elements in approximately 2 hours using a single GPU, while the reference implementation takes approximately 3 hours and 10 minutes using 48 CPU cores. Furthermore, it is shown that it is possible to solve nonlinear topology optimisation problems using GPU acceleration, demonstrated by a nonlinear end-compliance optimisation with finite strains and a Neo-Hookean material model discretised by 1 million elements.

Keywords: Topology Optimisation, GPU acceleration, Structural Optimisation

1. Introduction

Topology optimisation is a rapidly maturing technology which enables automatic design of application tailored structures [5, 36]. At the time of writing, this design method has been shown to work for a wide variety of physics. Nevertheless, the computational cost of performing the optimisation is challenging for many practical use-cases. Large-scale topology optimisation problems usually take days to run on expensive compute clusters [2, 1], which are only readily available to few large companies and academics. Therefore, the research community seeks to improve the computational efficiency of topology optimisation in order to make the methods easily available to small and medium sized companies. One approach is to use graphics processing units (GPUs) to accelerate the computation.

Advantages of GPU accelerated computing. Modern GPUs offer a great computing power to price ratio, compared to the more general purpose central processing units (CPUs). Consider the Nvidia 3080Ti GPU, which promises a theoretical limit of 34 TFLOPS on 32-bit floating point numbers. This card was launched in 2021 at a recommended retail price of 1200 USD [30]. A similarly priced CPU, the Intel Xeon W-3335 launched in 2021 with recommended retail price 1300 USD, promises a theoretical limit of 1 TFLOPS on 32-bit floating point numbers (4 GHz $\times$ 16 cores $\times$ 16 SIMD-lanes) [18]. This order of magnitude improvement of computing speeds explains the current interest in using GPUs for computationally intensive tasks outside of graphics processing. It should be noted however, that many graphics cards are optimised for 32-bit floating point performance, as these are commonly used for graphics. The graphics cards with good 64-bit floating point performance are usually high-end cards designed specifically for acceleration of scientific and machine learning workloads. This is a relevant observation as topology optimisation implementations rely on 64-bit, as the high precision is required to avoid truncation errors which arise in the solvers, both for the physics and the optimisation problem [20].

Challenges of general purpose GPU programming. The GPU is able to attain large performance improvements over the CPU by exploiting the large amount of parallelism in graphics processing. The GPU can be thought of as a stream processor, which applies some function, a so-called kernel, to a large set of inputs. It is crucial that implementations for the GPU follow this structure. Most frameworks for general purpose GPU programming, e.g. CUDA or OpenCL, are based on the notion of kernels, in order to ensure that the resulting programs are structured correctly [31, 38]. While these frameworks are able to produce very efficient code for the GPU, it is challenging to port CPU programs to GPUs, as the reformulation of the problem to kernels is left to the programmer. One important aspect of the highly parallel GPU architecture is that problems with irregular data access, such as sparse matrix factorisation, are non-trivial to implement efficiently.

An alternate approach to the kernel based model is to use compiler directives to declare loops as kernel executions to be performed by the GPU. This approach is followed by OpenMP and OpenACC which both allow loop bodies to be compiled into kernels as specified by compiler directives [8, 11]. Hence, such approaches are interesting when seen from a mechanical engineering perspective.

A practical approach to general purpose GPU programming. An alternat approach to writing explicit kernels is to use a programming language with higher levels of abstraction, which can be compiled to GPU kernels. One such language is Futhark [14, 12, 15], which was chosen for this work due to the thoroughness of its language documentation. Similar languages include Dex [32], and Lift [37]. There are several advantages to using Futhark. Writing software at a higher level of abstraction is generally faster. While purely based on perception, it is our experience that using Futhark has significantly reduced the total development time of the presented GPU programs, compared to the corresponding reference program. Furthermore, the Futhark compiler includes an aggressive ahead-of-time optimisation, which results in efficient compute kernels with little effort compared to writing the GPU kernels by hand.

Prior work in topology optimisation using GPUs. The use of general purpose GPU programming for topology optimisation problems goes back more than a decade to Wadbro and Berggren [39] who developed a CUDA implementation to optimise two-dimensional heat conductors. Schmidt and Schulz [34] solved a linear elasticity problem using the conjugate gradient method and were the first to discus the details of writing a GPU kernel for topology optimisation. Further work exploits the structure of Cartesian grids to improve the performance of GPU implementations by introducing multigrid preconditioners with low memory overhead [41, 23, 21].

An interesting issue not handled in this work concerns the solution of topology optimisation problems on unstructured meshes. A key challenge for unstructured meshes, is to avoid concurrent writes to a single node, or so-called data-races. One way to avoid data-races is to use graph-colouring, as done in Zegard and Paulino [43], Martínez-Frutos et al. [22]. Recently, it has been considered to build an algebraic multigrid solver on the CPU and transferring it to the GPU, in order to efficiently solve unstructured problems [16].

Summary of this work. In this work we present two GPU accelerated topology optimisation implementations for the linear elastic minimum compliance problem [5], one using the Futhark language, and the other using OpenMP 4.5 to generate GPU kernels. Furthermore, we present a reference implementation for the CPU based on OpenMP for performance comparisons. All presented codes are made publicly available, and it is the authors' hope that the accessibility of such codes will make high resolution topology optimization a viable option for a larger group of researchers and practitioners than those currently exploiting these possibilities.

Our implementations assume a Cartesian grid and we exploit this structure of the problem to improve the performance. We show that both GPU accelerated implementations for the linear problem are able to solve topology optimisation problems with more than 50 million hexahedral elements and 100 design iterations in a matter of hours on a single Nvidia A100 GPU [28].

Finally, we demonstrate the extendibility of one of the GPU implementations by solving a nonlinear elasticity problem. To this end we use the Futhark language to solve a nonlinear end-compliance problem [7] using finite strains and a Neo-Hookean material formulation. As in the linear case, a Cartesian grid is used, and exploited for performance in the implementation. The nonlinear equilibrium is solved using a Newton-Krylov approach. The nonlinear problem has a greatly increased computational complexity, due to the need for numerically integrating all element contributions during every matrix-free application of the

tangent matrix. We show that it is indeed feasible to solve topology optimisation problems with nonlinear elasticity on the GPU, although it is notably slower than its linear counterpart. This implementation is also made publicly available.

2. Theory and Methods

This section is intended to provide the reader with a sufficient overview over the theoretical background of the considered topology optimisation problems.

2.1. Elasticity

Linear Elasticity. Finite element analysis of small strain linear elasticity is well-known and is based on the following strain energy density

$$\phi_{\text{L}} = \frac{\lambda}{2}(E_{kk})^2 + \mu E_{ij}E_{ij}, \tag{1}$$

Where λ and μ denote the Lamé parameters and E_{ij} is the strain tensor. This formulation is thoroughly covered in many textbooks. In this work a tri-linear hexahedral element formulation is used, which can be found in Cook et al. [10]. For the sake of brevity, the formulation is not reproduced in this article. The interested reader is referred to Cook et al. [10].

Nonlinear elasticity. When considering large deformations, Green-Lagrange strains and a Neo-Hookean material model are adopted. The energy expression for the Neo-Hookean formulation is [19]:

$$\phi_{\text{NL}} = \frac{\lambda}{2}(\ln J)^2 + \frac{\mu}{2}(C_{ii} - 3) - \mu \ln J, \quad J = |f_{ij}| \tag{2}$$

Here f_{ij} denotes the deformation gradient tensor, $C_{ij} = f_{ik}f_{kj}$ denotes the Cauchy-Green tensor, and λ and μ the Lamé parameters

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)} \tag{3}$$

In order to stabilise the void elements an interpolation between the nonlinear energy and the linear formulation is used [40]. The interpolation on the element level is given as:

$$\phi_e = E_e \left[\phi_{\text{NL}}(\gamma_e \mathbf{u}_e) + (1-\gamma_e)\phi_{\text{L}}(\mathbf{u}_e)\right] \tag{4}$$

Here ϕ_{L} is the energy corresponding to the linear formulation. Note that the Young's modulus from the Lamé parameters is moved out as a scaling parameter of the entire energy. γ_e is an element-wise interpolation parameter, based on the filtered element density $\tilde{x}_e$, given by

$$\gamma_e = \frac{\tanh(\beta\rho_0) + \tanh(\beta(\tilde{x}_e - \rho_0))}{\tanh(\beta\rho_0) + \tanh(\beta(1-\rho_0))} \tag{5}$$

where the parameters $\beta = 500$ and $\rho_0 = 0.1$ are used for this work. The details of the formulation are given in Appendix A.

2.2. Topology optimisation formulation

The chosen topology optimisation problem treated in this work is the classical problem of compliance minimisation with density filtering and a volume constraint. A detailed explanation of the formulations can be found in Bendsøe and Sigmund [5], Buhl et al. [7] for the linear and nonlinear case respectively. The problems are summarised here for completeness.

Density filter. Ensuring that non-physical phenomena, such as chequer-boarding are avoided, the density filter is applied [6]. The density filter applies a local weighted average to the elements

$$\tilde{x_i} = \frac{\sum_j x_j w_{ij}}{\sum_j w_{ij}}, \quad w_{ij} = \max[r - d_{ij}, 0] \tag{6}$$

Where x is the local element density, r is the filter radius, and d_{ij} is the distance between element centres of element i and j. $\tilde{x}$ denotes the filtered density. Note that w_{ij} only takes non-zero values when $d_{ij} < r$. The element-wise constant densities are preferred as this improves the efficiency of the linear implementation, as element contributions with a constant Young's module are faster to compute.

Stiffness interpolation. In order to avoid intermediate densities in the resulting design, the Solid Isotropic Material Penalisation (SIMP) is used [5]. The interpolation occurs between some solid material stiffness $E_{\max}$, and a low void stiffness $E_{\min}$ chosen so low that it does not have any practical effect on the solution to the elasticity problem. In this work it is chosen such that $E_{\min} = 10^{-6} E_{\max}$. The element-wise interpolation is

$$E_e = E_{\min} + (E_{\max} - E_{\min})\tilde{x_e}^p \tag{7}$$

The penalisation parameter p is chosen to reduce the Youngs Module at intermediate values. In this work a value of $p = 3$ is used throughout. For every finite element the local Young's Module is computed by the interpolation. The assembly of local finite element matrices is otherwise unaffected.

Linear optimisation problem. The linear optimisation problem is a compliance minimisation problem for linear elasticity. The problem is analogous to that solved in several Matlab implementations [35, 3], with the exception that it is extended to three dimensions. The formal definition of the problem is

$$\begin{aligned}
&\underset{\mathbf{x}\in\mathbb{R}^{n_e}}{\text{minimize}} && \mathbf{u}^\top \mathbf{f} \\
&\text{subject to:} && \\
&\text{filter} && \tilde{\mathbf{x}} = F(\mathbf{x}) \\
&\text{state equation} && \mathbf{r}(\mathbf{u}, \tilde{\mathbf{x}}) = \frac{\partial \Pi}{\partial \mathbf{u}} - \mathbf{f} = 0 \\
&\text{volume} && \frac{1}{\sum_{e=1}^{n_e} v_e} \sum_{e=1}^{n_e} v_e [\tilde{\mathbf{x}}]_e \leq V^* \\
&\text{box} && 0 \leq [\mathbf{x}]_e \leq 1, \quad \forall e \in \{1, 2, ..., n_e\}
\end{aligned} \tag{8}$$

Here F is used to denote the elementwise computation of the filter operation from eq. (6), and Π is used to denote the integral of ϕ in the considered domain and $\mathbf{f}$ is the nodal load vector. For the linear problem we have that $\mathbf{r}(\mathbf{u}, \tilde{\mathbf{x}}) = \mathbf{K}(\tilde{\mathbf{x}})\mathbf{u} - \mathbf{f}$, where $\mathbf{K}(\tilde{\mathbf{x}})$ denotes the assembled stiffness matrix using the given density values to interpolate the Young's module. Note that for the employed rectangular grids, where all elements have identical shape and size, simplifying the volume constraint in practise. The optimisation problem is solved using the Optimality Criteria method [5], similar to the 88-line Matlab code presented in Andreassen et al. [3].

Nonlinear optimisation problem. The nonlinear optimisation problem is in many aspects similar to the linear problem. Both volume constraint, filter technique, and stiffness interpolation are unchanged from the linear problem. The goal of the optimisation process is to minimise the end-compliance, i.e. the compliance of the final deformation [7]. Indeed eq. (8) is still valid for the nonlinear problem, with the small change of interpretation that $\mathbf{r}(\mathbf{u}, \tilde{\mathbf{x}})$ is no longer a linear system. Instead, a nonlinear problem is solved to find the end-displacement, and a corresponding linear adjoint problem is solved in order to compute the gradients.

In order to compute the sensitivities of the residual, the adjoint method is used to obtain an analytic expression for the gradients. The derivation of the adjoint expression for end-compliance can be found in Buhl et al. [7]. Given a system where equilibrium has been found, the Lagrange multiplier λ can be found by

$$\mathbf{K}_t(\mathbf{u}_{\text{equilibrium}})\lambda = \mathbf{f} \tag{9}$$

Where $\mathbf{K}_t(\mathbf{u}_{\text{equilibrium}})$ denotes the tangent matrix in the system of equilibrium. Using the Lagrange multiplier, the sensitivity with respect to an element can be found by

$$\frac{\partial(\mathbf{u}^\top \mathbf{f})}{\partial \rho_e} = \lambda^\top \frac{\partial \mathbf{r}}{\partial \rho_e} \tag{10}$$

The right hand derivative term can be found by taking the partial derivative of eq. (A.22).

2.3. Solving the linear systems

The multigrid preconditioned conjugate gradient method is used to solve the linear systems, arising from both linear elasticity, as well as the linearisations of the nonlinear elastic problem. The method is widely used in large-scale topology optimisation problems [2, 4], as its error smoothing properties allow for an efficient solution of a design iteration by reusing the previous solution as an initial guess. In this work a V-cycle multigrid is employed as the preconditioner for the conjugate gradient method.

Conjugate gradient. The preconditioned conjugate gradient method is used to solve the linear problems. The method is well established, and details can be found in e.g. Saad [33]. For the linear elastic problem the mixed precision scheme presented in Liu et al. [20] is used. 32-bit floating point numbers are used to store the auxiliary vectors, while the matrix operations and multigrid hierarchy are computed using 64-bit floating point numbers.

The conjugate gradient method requires that the system matrix solved is symmetric positive definite (SPD). While this is always the case for linear elasticity, it is possible to generate non-SPD matrices in the nonlinear case when instabilities occur in the solution. In practice this does not happen unless the magnitude of the applied load is too high compared to the load-carrying ability of the structure. To remedy this potential issue a load continuation scheme, as discussed in section 3, is introduced during the early design iterations.

Multigrid preconditioner. A V-cycle multigrid, shown in algorithm 1, is used as the preconditioner for the conjugate gradient method. The prolongation matrices P_l are used to project the vectors and matrices between grids. These prolongation matrices are constructed by using the element shape-functions to point-wise evaluate the coarse grid value at the nodes of the finer mesh. In practise the operators P_l and K_l are never assembled in matrix form, but are implemented through functions, with the exception of K_l for the coarsest grids.

For the OpenMP implementations, the used coarse space matrices are not the Galerkin projections shown in algorithm 1 lines 1-3. Instead, the linear elastic stiffness matrix is assembled on the coarse grid, with spatially varying densities within each tri-linear hexahedral element, similar to the approach used in Nguyen et al. [24]. In practice, this means that the coarse grid matrices are assembled by considering the larger coarse elements with a multiresolution density described by the finest level density distribution. The element formulation itself is unchanged from Nguyen et al. [24]. This approach gives rise to a slightly worse preconditioner, but a much simpler implementation. It was therefore chosen for the OpenMP codes, to reduce their inherent complexity.

Algorithm 1: Multigrid V-cycle

Data: Initial residual b_0, Stiffness matrix K_0
Result: smoothed displacement u_0

1 **for** l=1,...,L **do**
2 | $K_l \leftarrow P_l^\top K_{l-1} P_l$; `// construct coarse space matrices`
3 **end**
4 **for** l=0,1,...,L-1 **do**
5 | $u_l \leftarrow SSOR(0, b_l, K_l)$; `// smooth`
6 | $r_l \leftarrow b_l - K_l u_l$; `// compute residual`
7 | $b_{l+1} \leftarrow P_{l+1\top} r_l$; `// restrict residual`
8 **end**
9 Solve $K_L u_L = b_L$ as tight as possible ; `// solve coarse space`
10 **for** l=L-1,L-2,...,0 **do**
11 | $u_l \leftarrow u_l + P_l u_{l+1}$; `// prolong solution estimate`
12 | $u_l \leftarrow SSOR(u_l, b_l, K_l)$; `// smooth`
13 **end**

Smoothing. The used multigrid preconditioner uses nodal symmetric successive over-relaxation (SSOR)

iterations to smooth the residual at all levels. This smoother is similar to the one used in Wu et al. [42], although it is extended to retain symmetry properties.

Algorithm 2: Nodal symmetric successive over-relaxation

Data: displacement vector $\mathbf{u}$, force vector $\mathbf{f}$, stiffness matrix $\mathbf{K}$, damping parameter $\omega(= 0.6)$
Result: smoothed displacement $\hat{\mathbf{u}}$

$\mathbf{s} \leftarrow \mathbf{K}\mathbf{u}$; // apply stiffness matrix
$\hat{\mathbf{u}} \leftarrow \mathbf{u}$;
forall nodes n in mesh **do** // Update independently
  $M \leftarrow \mathbf{K}^n$; // nodal 3x3 matrix
  $r \leftarrow \mathbf{s}^n - Mu^n$;
  $\hat{\mathbf{u}}_1^n \leftarrow \frac{1}{M_{11}}(\mathbf{f}_1^n - r_1 - M_{12}\hat{\mathbf{u}}_2 - M_{13}\hat{\mathbf{u}}_3)$;
  $\hat{\mathbf{u}}_2^n \leftarrow \frac{1}{M_{22}}(\mathbf{f}_2^n - r_2 - M_{21}\hat{\mathbf{u}}_1 - M_{23}\hat{\mathbf{u}}_3)$;
  $\hat{\mathbf{u}}_3^n \leftarrow \frac{1}{M_{33}}(\mathbf{f}_3^n - r_3 - M_{31}\hat{\mathbf{u}}_1 - M_{32}\hat{\mathbf{u}}_2)$;
end
$\mathbf{u} \leftarrow \omega\hat{\mathbf{u}} + (1-\omega)\mathbf{u}$; // Damped update
$\hat{\mathbf{u}} \leftarrow \mathbf{u}$;
$\mathbf{s} \leftarrow \mathbf{K}\mathbf{u}$;
forall nodes n **do** // Update independently
  $M \leftarrow \mathbf{K}^n$;
  $r \leftarrow \mathbf{s}^n - Mu^n$;
  $\hat{\mathbf{u}}_3^n \leftarrow \frac{1}{M_{33}}(\mathbf{f}_3^n - r_3 - M_{31}\hat{\mathbf{u}}_1 - M_{32}\hat{\mathbf{u}}_2)$;
  $\hat{\mathbf{u}}_2^n \leftarrow \frac{1}{M_{22}}(\mathbf{f}_2^n - r_2 - M_{21}\hat{\mathbf{u}}_1 - M_{23}\hat{\mathbf{u}}_3)$;
  $\hat{\mathbf{u}}_1^n \leftarrow \frac{1}{M_{11}}(\mathbf{f}_1^n - r_1 - M_{12}\hat{\mathbf{u}}_2 - M_{13}\hat{\mathbf{u}}_3)$;
end
$\hat{\mathbf{u}} \leftarrow \omega\hat{\mathbf{u}} + (1-\omega)\mathbf{u}$;

Algorithm 2 shows the nodal SSOR smoothing as implemented for the multigrid. Here the superscript $(\cdot)^n$ is used to describe accessing the local vector of size 3, or 3×3 matrix, associated with node n. The nodal SSOR is applied successively for several smoothing sweeps, in this work two sweeps are always used unless otherwise stated. The damping parameter ω is chosen as a constant 0.6 in this work, as this is found to perform consistently well through trials. The OpenMP implementations use the simpler Jacobi iteration to smooth the residual on intermediate levels.

Coarse space correction. The coarse space correction is solved approximately using the Jacobi preconditioned conjugate gradient method. Ideally a direct solver should be employed, as the coarse problem is typically so small that a direct solver yields the best results. Unfortunately, a sparse matrix direct solver is currently not available in the Futhark ecosystem, and therefore an alternative approach was necessary.

Similar to the PETSc based topology optimisation framework from Aage et al. [2], the Futhark implementation uses yet another Krylov method to solve the coarse space correction. Specifically, the Futhark implementation uses the Jacobi preconditioned conjugate gradient due to memory efficiency, and due to ease of implementation. The conjugate gradient method is run as a coarse space correction for 800 iterations (4000 iterations in the nonlinear case), or until a relative tolerance of 10^{-10} is achieved. The OpenMP implementations, on the other hand, use a direct solution strategy for the coarsest level, using the Cholmod package to implement the factorisation and back substitution [9].

3. Implementation.

This section is intended to provide the reader with a sufficient overview over the three linear elastic implementations in question, their differences, complexities, and limitations. However, as none of the

presented code frameworks are shorter than 100 lines, cf. the multitude of Matlab codes, it is not possible to go through the frameworks line by line. Instead the most important and crucial parts are highlighted and explained in detail. The three implementations follow very similar structures and, unless otherwise noted, details given in this section are valid for all implementations. Two slower reference implementations written in Matlab are also present in the OpenMP repositories, which can be used to compare with the provided implementations. These Matlab implementations are not discussed further in this work, as they are significantly less efficient than the provided OpenMP-CPU implementation. The three implementations are denoted as follows.

Futhark The first GPU accelerated implementation is developed in the Futhark language [14]. The Futhark implementation for the linear problem is available at doi:10.5281/zenodo.7791871.

OpenMP-GPU The second implementation is written in the C language with OpenMP 4.5, and uses GPU acceleration through OpenMP. The OpenMP-GPU implementation is available at doi:10.5281/zenodo.7791868.

OpenMP-CPU Finally, a version of the OpenMP implementation which only uses OpenMP to parallelise the work on the CPU is presented, mostly as a reference code. This implementation is available at doi:10.5281/zenodo.7791870.

Remark, that the exact details of the solver setup differs between the Futhark implementation and the two OpenMP based implementations. This is because there is no direct factorisation implementation available for sparse matrices in Futhark, and the coarse space correction is therefore only computed approximately using the conjugate gradient algorithm [25]. In principle, it would be possible to split the futhark kernels in a way that allowed a direct solver to be employed on the coarse level, but this was not chosen, as it would introduce a lot of complexity to the implementation, and hence, it was deemed outside the scope of this work which attempts striking a balance between simplicity and performance. In contrast, the OpenMP implementation has easy access to any direct solver which has a C interface. It was chosen to use the best possible solver setup for each implementation, in order to avoid restricting the performance artificially. As the multigrid method generally performs better with a direct solution on the coarse space, it was chosen for the OpenMP implementations. The direct solver comes at the cost of transferring data between the CPU and GPU every time the preconditioner is applied. We nevertheless found that the total number of used iterations was reduced, resulting in an overall reduction of compute time. The second and final difference between the Futhark and OpenMP implementations is that the Futhark implementation uses nodal successive over-relaxation for smoothing the residual between multigrid levels, while the OpenMP implementations use Jacobi smoothing. SSOR was only implemented for Futhark, as the high-level language eased writing an additional smoother. While the SSOR implementation was found to perform better, it was not ported to the OpenMP based codes, due to a consideration of the improvement of run-time compared to the time required for implementation.

Matrix-free operator. In order to update all nodes in the finite element mesh independently, a nodal traversal approach is used. An informal way to describe the update $f = Ku$ is by

$$f_i^n = \sum_{e \in \mathcal{E}_n} E_e \sum_{j=1}^{24} K_{ij}^{e,n} u_j^e, \quad i \in \{1, 2, 3\} \tag{11}$$

where f_i^n denotes a component of the resulting force vector at node n, $\mathcal{E}_n$ denotes the set of neighbouring elements of the node n, E_e denotes the Young's modulus associated with element e, $K_{ij}^{e,n}$ denotes a 3×24 slice of the preintegrated stiffness matrix, where the three rows correspond to the node n, and u_j^e denote the local deformations associated with the nodes of element e.

While this approach requires additional work, as u_j^e and E_e are computed once for every neighbouring node, we can parallelise the updates across nodes, without the usual problem of potential data-races if the nodal values are updated element-by-element. In order to apply the matrix-free operator for a coarse space, such as the next-to finest mesh, the nodal values are prolonged to the fine mesh, and the fine mesh matrix-free operator is applied. The resulting values are then restricted to the original mesh. This is in contrast to most

standard FEM implementations. The reader is referred to Schmidt and Schulz [34], Wu et al. [41] which treat the subject of nodal updates more thoroughly.

Assembly of coarse operators. For the coarse levels in the multigrid preconditioner, the Galerkin projection of the matrices is constructed and stored in memory. The computation of the matrices is shown in algorithm 1 and an example for level 2 is given in eq. (12).

$$K_2 = P_2^\top P_1^\top K_0 P_1 P_2 \tag{12}$$

The matrices cannot be computed using sparse matrix-matrix products, as the matrices for the finest levels cannot be stored in memory. Instead, the coarse matrices are computed by repeatedly applying vectors from the standard basis of the coarse space. As an example, the first column of the assembled matrix at level 2 can be extracted by applying matrices of the left hand side of eq. (12) to $\mathbf{e}_1$, i.e. the basis vector with value 1 in the first entry and 0 in all other entries. By repeatedly prolonging the values using matrix-free representations of P_l, then applying the matrix-free fine level operator K_0, and finally restricting the resulting values back to the coarse mesh. The locality of the prolongation and restriction can be used to only compute fine values in a small neighbourhood around the node which is being expanded, which is needed for making this approach feasible. For the alternate coarse space matrix formulation used in the OpenMP implementations, a usual finite element assembly is used for the coarse space matrices as described in section 2.3.

3.1. Comparison of languages

In order to compare the complexity of the different implementations, we show the implementation of the density filter in both Futhark and C with OpenMP in Listing 1 and 2. The purpose of this comparison is to show the differences in programming styles, and ease of use. The interested reader may compare these implementations with the simpler `convolutionTexture` examples [27] and the optimised `convolutionSeparable` [26] example provided by NVIDIA, which both implement a separable convolution in two dimensions. As the density filter on a structured grid is also a separable convolution, albeit in three dimensions and with a slightly more complex filter kernel, these implementations give a good indication of how an equivalent CUDA implementation might look.

We note that there is a trade-off for performance in the choice of high-level languages. The simplicity comes at a loss of control of the exact layout of memory, thread groupings, and more. Therefore, a carefully written and tuned lower-level implementation will most probably outperform the presented implementations. Furthermore, high-level languages do not absolve the user from knowing some architectural details to achieve the best performance, such as e.g. having to set an appropriate stencil size in the OpenMP implementations to match available parallelism.

The Futhark implementation in listing 1 is based on a functional programming language, which uses higher order functions. The concept of this implementation is to compute the two sums from eq. (6) independently, before computing the filtered density value. Initially a series of helping methods and types are defined on lines 1-32. Specially noteworthy is the `sumOverNeighbourhood`, which is a so-called higher-order function that evaluates an input function for all neighbouring elements, and sums the results. This is a core operation in the density filter, which is used to compute both the sum of weights (line 40), and the sum of the scaled densities (line 41). The implementation of `sumOverNeighbourhood` makes use of pipes (written `|>`) which work much like Unix pipes, passing the output of the left expression as input to the function to the right. Another noteworthy detail is the partial application of functions, which is used to create new function, seen in lines 37, 40, and 41. Here the first input values of a function are passed, to create a new function that takes the remaining input.

```
1 type index = {x: i64, y: i64, z: i64}
2
3 -- Check if index is valid.
4 let isInsideDomain nelx nely nelz (idx :index) :bool =
5  idx.x >= 0 && idx.y >= 0 && idx.z >= 0 &&
6  idx.x < nelx && idx.y < nely && idx.z < nelz
7
```

```
8  -- Compute weight given radius, element, and neighbour.
9  let getFilterWeight rmin (ownIdx :index) (neighIdx :index) =
10  f32.max 0 (rmin - f32.sqrt( f32.i64 (
11   (ownIdx.x-neighIdx.x)*(ownIdx.x-neighIdx.x) +
12   (ownIdx.y-neighIdx.y)*(ownIdx.y-neighIdx.y) +
13   (ownIdx.z-neighIdx.z)*(ownIdx.z-neighIdx.z))))
14
15 -- Sums the output of computeValue in a local neighbourhood with radius boxRadius of
       element [i,j,k].
16 let sumOverNeighbourhood boxRadius nelx nely nelz i j k (computeValue :index -> f32) =
17  let boxSize = 2*boxRadius+1
18  in tabulate_3d boxSize boxSize boxSize (\ii jj kk ->
19   let neighIdx :index = {x=i+ii-boxRadius,y=j+jj-boxRadius,z=k+kk-boxRadius}
20    in
21     if isInsideDomain nelx nely nelz neighIdx
22      then computeValue neighIdx
23      else 0)
24  |> map (map f32.sum)
25  |> map f32.sum
26  |> f32.sum
27
28 -- given origin and neighbour indices, compute the weighted density contribution.
29 let getScaledDensity (x :[][][]f32) rmin ownIdx neighIdx =
30  let wgt = getFilterWeight rmin ownIdx neighIdx
31  let dens = #[unsafe] x[neighIdx.x,neighIdx.y,neighIdx.z]
32  in wgt*dens
33
34 -- For all elements, compute the filtered density.
35 entry forwardDensityFilter [nelx][nely][nelz] (rmin :f32) (x :[nelx][nely][nelz]f32) :[nelx
       ][nely][nelz]f32 =
36  let boxRadius = i64.max 0 (i64.f32 (f32.ceil (rmin-1)))
37  let sumOverThisNeighbourhood = sumOverNeighbourhood boxRadius nelx nely nelz
38  in tabulate_3d nelx nely nelz (\i j k->
39   let ownIdx :index = {x=i,y=j,z=k}
40   let weightSum = sumOverThisNeighbourhood i j k (getFilterWeight rmin ownIdx)
41   let scaledDensity = sumOverThisNeighbourhood i j k (getScaledDensity x rmin ownIdx)
42   in scaledDensity / weightSum)
```

Listing 1: Density filter implementation in Futhark

The C implementation using OpenMP in listing 2 is based on a slightly different algorithm for computing the density filter. Instead of adding 0 values for the out-of-bounds neighbours, as done in the Futhark implementation, their contributions are not computed, by adjusting the size of the inner loop. The C implementations have domain padding, changing the loop limits (line 10-12), and complicating the computation of indices slightly (lines 13,30). A `gridContext` data-structure is used to store grid dimensions, including information on the padding. The domain padding is added to allow an efficient implementation of the stiffness matrix operator, which is one of the most time-consuming methods. While the density filter would be faster if a style like the Futhark implementation was used, this was not prioritised, as the overall program time spent in the density filter is low.

```
1  void applyDensityFilter(const struct gridContext gc, const DTYPE rmin, const DTYPE *rho,
       DTYPE *out) {
2   const uint32_t nelx = gc.nelx;
3   const uint32_t nely = gc.nely;
4   const uint32_t nelz = gc.nelz;
5
6   const uint32_t elWrapy = gc.wrapy - 1;
7   const uint32_t elWrapz = gc.wrapz - 1;
8
9   #pragma omp target teams distribute parallel for collapse(3) default(none) firstprivate(
       nelx,nely,nelz,rmin,elWrapy,elWrapz) shared(out,rho)
10  for (unsigned int i1 = 1; i1 < nelx + 1; i1++)
11   for (unsigned int k1 = 1; k1 < nelz + 1; k1++)
12    for (unsigned int j1 = 1; j1 < nely + 1; j1++) {
```

```
13      const uint32_t e1 = i1 * elWrapy * elWrapz + k1 * elWrapy + j1;
14      double oute1 = 0.0;
15      double unityScale = 0.0;
16
17      // loop over neighbourhood
18      const uint32_t i2max = MIN(i1 + (ceil(rmin) + 1), nelx + 1);
19      const uint32_t i2min = MAX(i1 - (ceil(rmin) - 1), 1);
20
21      for (uint32_t i2 = i2min; i2 < i2max; i2++) {
22       const uint32_t k2max = MIN(k1 + (ceil(rmin) + 1), nelz + 1);
23       const uint32_t k2min = MAX(k1 - (ceil(rmin) - 1), 1);
24
25       for (uint32_t k2 = k2min; k2 < k2max; k2++) {
26        const uint32_t j2max = MIN(j1 + (ceil(rmin) + 1), nely + 1);
27        const uint32_t j2min = MAX(j1 - (ceil(rmin) - 1), 1);
28
29        for (uint32_t j2 = j2min; j2 < j2max; j2++) {
30         const uint32_t e2 = i2 * elWrapy * elWrapz + k2 * elWrapy + j2;
31         const double filterWeight = MAX(0.0, rmin - sqrt((i1 - i2) * (i1 - i2) + (j1 - j2) *
       (j1 - j2) + (k1 - k2) * (k1 - k2)));
32
33          oute1 += filterWeight * rho[e2];
34          unityScale += filterWeight;
35        }
36       }
37      }
38      out[e1] = oute1 / unityScale;
39     }
40 }
```

Listing 2: Density filter implementation in C with OpenMP

The OpenMP pragma which parallelises the density filter is shown in line 9. To get the corresponding multi-threaded CPU implementation, the keywords `target teams distribute` should be removed. This is a standard pragma that compiles the loop-body to a GPU kernel, with a `collapse` keyword to indicate that it is the iteration space of all three loops which must be distributed. The `firstprivate` clause is used to indicate that thread-local copies of the variable are created, initialised with the current value of the variable. The `shared` clause indicates that these variables are accessible by all threads, and that the programmer is responsible to avoid data-races.

While the Futhark implementation is about the same length as the C implementation, it has several advantages. By moving the summation out as a higher order function, the filtering method itself is written very concisely, and other similar methods, such as the filtering of the gradients using the chain rule, can use the same set of helping functions. While helping functions are also possible to implement in C, the implementation of local neighbourhood summation would quickly become complex and prone to mistakes, due to the use of function pointers. Furthermore, the strict type system in Futhark allows many types to be inferred by the compiler, and ensures that there is no unexpected behaviour due to implicit type conversions. As a partial summary, the authors infer that Futhark is in many ways simpler to work with than C/OpenMP, despite the additional need to learn the Futhark language in the first place.

When comparing these implementations to the CUDA convolution examples [26, 27], it can be seen that the CUDA examples are significantly longer, even though they solve a simpler problem. When trimming the CUDA programs for headers and such, it can be seen that the simple and optimised implementations are 130 and 175 lines respectively, compared to the around 40 lines for both Futhark and OpenMP in listings 1 and 2. The optimised kernel splits the computation into blocks, adds a halo, unrolls loops, and uses more of such transformations to improve the performance. While this is necessary to achieve the best performance on a GPU, it also decreases readability and ease of understanding of the implementation.

[17, 28]. [29].

Tuning the OpenMP-GPU Implementation. In general, GPUs excel at doing simple, identical operations on many pieces of data simultaneously. This is known as Single Program Multiple Data (SPMD). In principle, adding OpenMP directives for GPU acceleration to an existing C-code is a simple process.

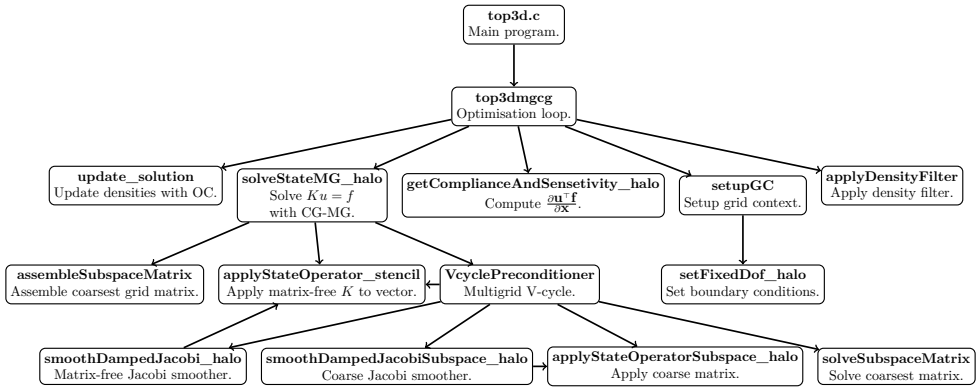


Figure 1: Program structure of the OpenMP-GPU implementation. Each node represents a significant function.

However, it is required to refactor some loops to increase the level of parallelism and to reduce the number of registers used in the loop bodies to achieve efficient kernels.

As copying data between the CPU and the GPU introduces overhead, it is important to think about where data resides and when it is transferred. Determining which parts of the program should run on the host and the accelerator, therefore, has to be decided by inspecting profiling results for various problem sizes. Due to the overhead of transferring data, it may be beneficial to run some inefficient kernels on the GPU if that can bring down the number of transfers.

As an example, the density filter kernel in listing 2 is rather complex and does not achieve high streaming-multiprocessor throughput due to non-uniform access patterns. However, by running it on the GPU we can limit the number of data transfers. Further, it is important to verify that the `map` clause has the intended behaviour. When doing a reduction on the GPU it may be needed to use the `always,tofrom` map modifier for the reduction variable. Otherwise, the reduction variable may be uninitialised on the GPU and the result may not be copied back to the CPU. That depends on whether or not the reduction variable has already been mapped. It is also important to ensure that the environment variable `OMP_TARGET_OFFLOAD` is set to `MANDATORY`. Otherwise, bugs can be introduced if a kernel may be executed on the host instead of the accelerator while the result is transferred back from the accelerator.

3.2. OpenMP implementation structure.

The two OpenMP based implementations are structured similarly and are therefore both covered in this section. The program execution begins in `main.c`, which reads parameters from the command line, and calls the optimisation routine. The main parts of the implementation are described in this section. A sketch of the most important functions, and their calling sequence is presented in fig. 1, to help navigate the implementation.

System definitions. The header `definitions.h` is used to set several important system-wide parameters, which are used by the entire program. Most importantly, the stencil size and floating point types are defined here. The stencil size indicates the amount of nodes in the finite element mesh are updated simultaneously. For the CPU version best performance is achieved by selecting the number of double precision floating points which fit in the vector instructions of the machine, which are 4 for AVX2, or 8 for AVX512. The default value is set to 8, which ensures acceptable performance on all current CPUs. For the GPU version best performance is usually achieved with 64 or 128, depending on the used GPU. Again, the default value is set to the higher value of 128 to ensure acceptable performance everywhere.

`definitions.h` also defines the `gridContext` struct, which is used to store all necessary grid data, such as local element matrices, material parameters, and boundary conditions. Additional methods to use the

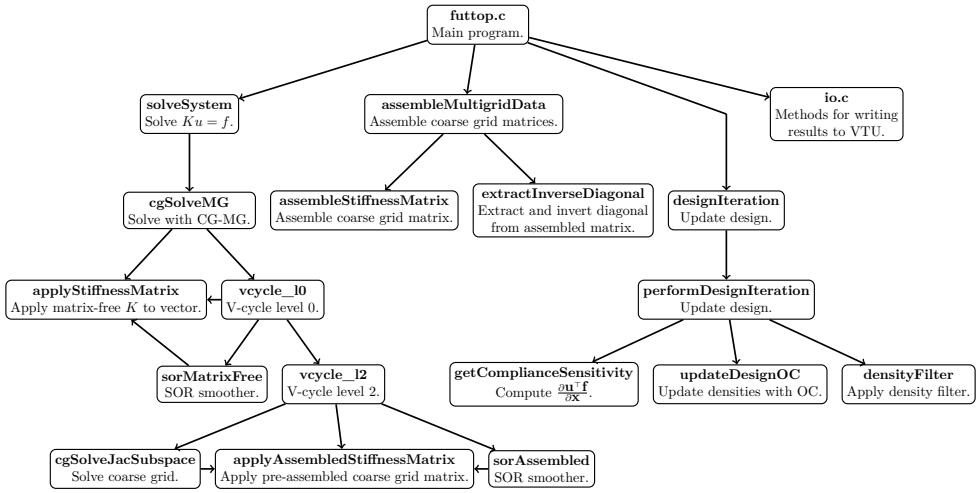


Figure 2: Program structure of the Futhark implementation. Each node represents a significant function.

struct are defined in `grid_utilities.h`, such as the initialisation utility **setupGC**. This is also where the boundary conditions are defined in the function **setFixedDof_halo**, and where they can be modified.

Optimisation. `stencil_optimization.c` is the central file for the optimisation process, where the main optimisation loop is defined in the **top3dmgcg** method. Optimisation specific methods, such as filtering (**applyDensityFilter**), computation of sensitivities (**getComplianceAndSensitivity_halo**), and the optimality criteria update (**update_solution**) are also implemented here.

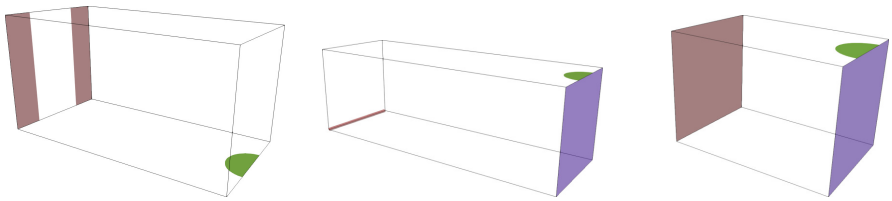
Multigrid solver. The multigrid solver is implemented in the `multigrid_solver.c` file, which includes the conjugate gradient method for the fine level (**solveStateMG_halo**), and the multigrid preconditioner itself (**VcyclePreconditioner**). Grid operations, such as the matrix-free stiffness matrix product for the fine level (**applyStateOperator_stencil**), matrix-free stiffness matrix product for the coarse level (**applyStateOperatorSubspace_halo**), and prolongations between grids are defined in `stencil_methods.c`, using utility methods from `stencil_utility.h`. The Cholmod [9] direct solver for the coarse grid (**solveSubspaceMatrix**) is called in `coarse_solver.c`, and the assembly of the coarse grid matrix is defined in `coarse_assembly.c`.

Utilities. Additional utility methods are also included in separate files. `local_matrix.c` defines the integraiton of the local matrices, which is performed once before starting the program. `write_vtk.c` defines methods to write the result to a vtk file. A small benchmark suite is also present in `benchmark.cpp`, using the Google Benchmark library.

3.3. Futhark implementation structure.

The Futhark implementation is structured differently from the OpenMP implementation, due to the functional nature of the Futhark language. This structure is sketched in fig. 2, to help navigate the implementation. The execution begins in the `futtop.c` file, which contains the main part of the program. The GPU kernels which are called from the C code are defined in the `libmultigrid.fut` source file, which imports all necessary Futhark sources.

Optimization. The main design loop is implemented in `futtop.c`, which calls four different GPU kernels, all defined in `libmultigrid.fut`. These kernels apply the density filter, assemble the coarse space matrices (**assembleMultigridData**), solve the linear system (**solveSystem**), and update the design variables (**designIteration**). The choice was made to split the functionality into multiple kernels, as this



(a) Cantilever. (b) MBB beam. (c) Double clamped beam.

Figure 3: Domains for the presented examples in this article. Purple is used to denote a surface with a symmetry condition normal to the surface. Green is used to indicate the position of the traction loads. Brown is used the indicate the position of the diriclet boundary conditions. For (a) and (c) red indicates a clamped surface. For (b) brown indicates that the displacement is 0 in the vertical direction, furthermore one point on the brown surface in (b) is prescribed a zero displacement in the y direction.

improves the compilation time of the Futhark compiler. Updating the design variables with the optimallity criteria method is implemented in the **updateDesignOC** method, defined in `optimization.fut`. The forward and backward filtering are implemented in `densityFilter.fut`.

Multigrid solver. The preconditioned conjugate gradient solver **cgSolveMG** is defined in `solver.fut`, while the multigrid preconditioner **vcycle_l0** is defined in `multigrid.fut`. Similarly, `projection.fut` implements the projection between multigrid levels, `sor.fut` implements the smoothing operations **sorMatrixFree** and **sorAssembled**, `assembly.fut` implements the assembly of coarse matrices, and so on. Of special interest is **applyStiffnessMatrix** defined in `applyStiffnessMatrix.fut`, which implements the matrix-free stiffness matrix product, and `boundaryConditions.fut` which define the boundary conditions for the problem, which are then called by **applyStiffnessMatrix**.

Utilities. The program makes use of several utility methods, which helps keep the code concise. These are defined in `utilities.fut` for general array utilities, and `indexUtilities.fut` for indices and element numbering. Pre-computed matrices are also stored in the separate source files `keConstants.fut` and `assemblyWeights.fut`. Methods to write VTK files are implemented in `io.c`.

4. Numerical experiments

In order to validate the correctness and performance of the GPU implementations, several numerical examples are presented here. Three examples are considered for both linear and nonlinear elasticity, a cantilever (aspect ratio $2 \times 1 \times 1$), the MBB beam (aspect ratio $3 \times 1 \times 1$), and a double clamped beam example (aspect ratio $1.5 \times 1 \times 1$), as shown in fig. 3. For all presented examples the following material parameters are used $E = 1\,\text{Pa}$ and $\nu = 0.3$.

4.1. Linear elasticity

This section is intended to demonstrate that the presented algorithms produce physically sound designs, and more importantly, to discus the efficiency and scaling of the presented implementations.

Cantilever. The first linear elastic example is the cantilever example. The cantilever domain is shown in fig. 3a. A constant traction load is applied on a small circular surface, with radius corresponding to a fifth of the domain width, as shown by green in fig. 3a.

A resulting design for the Futhark code is presented in fig. 4 for a $640 \times 320 \times 320$ mesh with approximately 65.5 million elements. The design was optimised using a volume fraction of $V^* = 0.1$, filter radius $r = 2.5$ elements, and 100 design iterations. The resulting designs for the other two codes are not shown, as they are indistinguishable. The presented result was computed using the Futhark implementation on an A100 GPU in 7380 seconds ($\sim$ 2 hours). In comparison, the OpenMP implementations are estimated to take approximately 2 hours (GPU) and 3.15 hours (CPU) , by extrapolating the computation time of the first 20 design iterations presented in fig. 6.

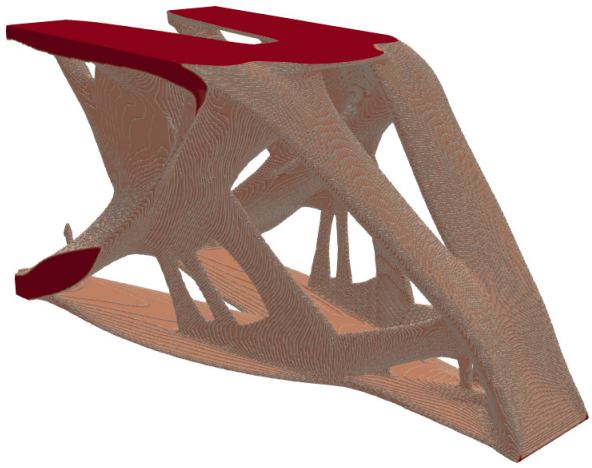


Figure 4: Linear cantilever example with 65.5 million elements, volume fraction of $V^* = 0.1$, and filter radius of 2.5 elements. Computed using the Futhark implementation on a Nvidia A100 GPU to run 100 design iterations, in 7380 seconds, approximately 2 hours. The shown result is thresholded such that elements with filtered density lower than 0.5 are removed.

It can be seen that the resulting structure has the expected V-shape near the clamped boundary, as the bending stiffness is increased by placing material near the top and bottom of the domain. The structure has a bottom plate, and a top plate, which curve into two supporting bars for the load.

MBB beam. The classic MBB beam example is also considered. The domain is shown in fig. 3a, where the brown line denotes a rolling boundary condition, which allows displacement along the longer aspect ratio, but restricts displacement in to two remaining directions.

Figure 5 shows a resulting MBB design using the linear formulation. The result is computed using the Futhark implementation, the results from the two other implementations are not shown, as they are identical. This example contains 41 million elements and was computed in 4 hours using the Futhark implementation and a Nvidia A100 GPU as accelerator. Compared to the cantilever, the MBB example uses more time to compute for fewer elements. This is due to the fact that the loading and larger domain aspect ratios makes it harder for the iterative solvers, and more iterations are required to solve the system [1].

The resulting structure makes intuitive sense, i.e. it can be seen to have two plates on the top and bottom of the domain with stiffening structure in between. The two round supporting corners are connected by beams, as the rolling support only supports in the vertical direction.

4.2. Performance Analysis of Linear Elasticity

Having established that all the presented implementations are capable of solving the design problem to satisfaction, it is time to consider the scaling and efficiency. To achieve this, the cantilever example is used. The wall-clock time of computing the first 20 design iterations is compared across implementations, for a variety of mesh sizes. The Gnu Compiler Collection (version 11.2) was used to compile the OpenMP-CPU code, the NVIDIA HPC SDK (version 22.5) was used for OpenMP-GPU, and the Futhark compiler (version 0.21.11) with GCC was used for the Futhark code. The used compiler flags can be found in the Makefile of the respective code repositories. It should be noted that the performance of the OpenMP-CPU implementation is sensitive to compiler optimisations, most notably enabling manipulation of floating point expressions assuming associativity greatly improves performance. The OpenCL backend was used for the Futhark compiler. Both the CUDA and OpenCL backends were tested, and it was found that the OpenCL backend resulted in faster GPU-kernels for this specific program. The multicore backend was also tested for the

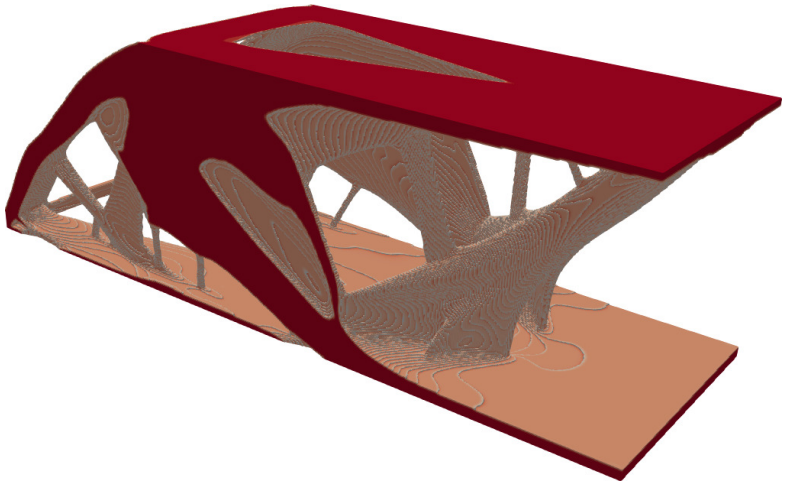


Figure 5: Linear MBB beam example with 41 million elements, a volume fraction of $V^* = 0.1$, and a filter radius of 2.5 elements. The shown result is thresholded such that elemenst with filtered density lower than 0.5 are removed.

Futhark program, which generates a multithreaded CPU program. However, this backend was found to perform significantly worse than the OpenMP-CPU code. This is somewhat expected, as the main focus of Futhark is compilation to GPU-kernels. For the remainder of this work we consider only the OpenCL backend of Futhark.

Two different test machines are used for the performance benchmarks. For the tests of CPU based implementations a machine with two Intel Xeon 8160 CPUs is used. The same machine also has a Nvidia 1080Ti GPU, which is used for GPU acceleration. The other machine used for testing has a Nvidia A100 GPU (80gb), and an Intel Xeon 6226R CPU, which has 16 cores and 700 Gb of RAM. One A100 GPU and the two Intel Xeon 8160 processors have a similar recommended retail price at launch (12,500 USD vs 2× 4700 USD), although the CPUs were launched in 2017, and the A100 in 2020 [17, 28]. This is also reflected in their double precision theoretical peak performances, which are 19.5 TFLOPS for the A100 [28], and 3.2 TFLOPS for the two Intel Xeon 8160 processors (2×24 cores $\times$ 32 FLOP/core $\times$ 2.1 GHz). This is an important observation when comparing the performance across various hardware components, in order to achieve a better comparison. Overall, the A100 GPU setup is both slightly more expensive, and is significantly newer than the CPU setup. Ideally, the release date and launch price would match better between the used CPU and GPU configurations, but unfortunately we are limited to the resources available to us. Finally, the CPU benchmark machine is also equipped with a Nvidia 1080Ti GPU, with a theoretical peak performance of 0.35 TFLOPS for double precision, which was launched in 2017 at 699 USD recommended retail price [29]. The 1080Ti is used to show the performance of the GPU implementations on consumer-grade GPU hardware, which is optimised for single precision floating point operations. As such, the 1080Ti is expected to be a bad match for these implementations, which perform key operations in double precision. However, there is a a tendency for compute capabilities of high-end GPUs to become available in the consumer grade market after some generations, e.g. tensor cores are becoming available in the consumer cards produced by NVIDIA.

While computing times are in general a stochastic parameter, it was found that the variations in wall-clock time were much smaller than the differences between implementations, therefore results for a single sample are presented.

Two studies are performed, one with varying filter radius and the second with a physically constant filter radius. For the variable filter example, a filter radius of 1.5 elements is used for all tests, meaning that

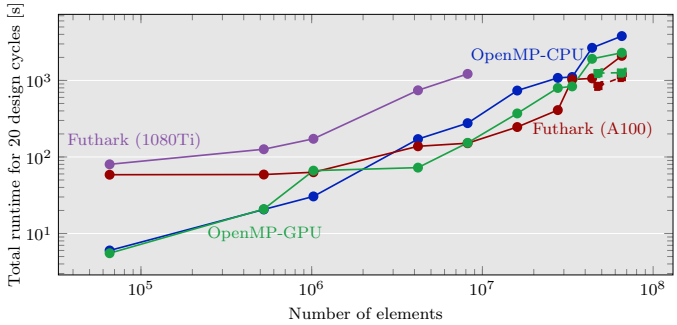


Figure 6: Runtimes for 20 design iterations of a linear elastic minumim compliance cantilever example with a volume fraction of $V^* = 0.1$. The optimisations were performed using a filter radius of 1.5 elements, resulting in possibly different topologies obtained for every mesh size. The blue and green lines indicate the OpenMP implementations for CPU and GPU respectively. The purple and red lines show the Futhark implementation on two different GPUs. All shown times are computed using 4 multigrid levels, with the exception of the additional points shown with dashed line and square marks, which are computed using 5 levels.Z

the physical filter radius is decreased when the mesh is refined. This means that the resulting structure is different for every refinement considered. This will affect the iterative solver, which can potentially have great variation in the number of solver iterations across designs. The constant filter radius example uses a filter radius of a 20th of the domain width, corresponding to 1.6 elements on the coarse mesh to 8 elements on the fine mesh.

The timings for the variable filter can be seen in fig. 6. It can be seen that all implementations are capable of solving large problems in a reasonable time. The slowest implementation is the OpenMP-CPU implementation, solving the problem with 65.5 million elements. It still performs 20 design iterations in 3800 seconds (63 minutes), which can be extrapolated to about 3.15 hours for the entire topology optimisation problem. In contrast, the OpenMP code which is ported to the GPU completes the same 20 design iterations in 2300 seconds (40 minutes) using the A100 GPU, which extrapolates to about 2 hours to complete the full problem. Similarly, the Futhark implementation completes the 20 design iterations in 2096 seconds ($\sim$ 35 minutes). Problems with less that 4 million elements are solved faster by the pure OpenMP-CPU implementation, as the overhead of offloading outweighs the performance improvements.

The two GPU accelerated implementations perform better for the largest problems if a fifth multigrid level is added. These data points are shown as squares in fig. 6, and the computation times are reduces to 1105 seconds and 1264 seconds for Futhark and OpenMP-GPU respectively, reducing the estimated times for the entire optimisation problem to 55 and 63 minutes respectively. We speculate that the mechanisms for performance improvement are quite different in the GPU implementations. For OpenMP-GPU, the amount of data to be transferred between the GPU and CPU is reduced, along with the size of the direct system to be solved on the CPU. This is beneficial, even at the cost of a slight decrease in the accuracy of the multigrid approximation. In the Futhark implementation however, the accuracy of the coarse grid correction increases, as the fixed number of iterations on the coarse grid can reduce the error further.

Figure 6 also shows the performance when using the Nvidia 1080Ti GPU with the Futhark implementation. The 1080Ti is representative of a consumer grade GPU, which is more limited in both memory and compute power compared to the A100 GPU. The curve stops at 8 million elements, as the Futhark implementation goes out-of-memory for more finely refined meshes. It is seen that the Futhark implementation running on a 1080Ti is notably slower than the other considered implementations. This is a reflection of the fact that the 1080Ti is optimised towards single precision floating point operations, while our implementations rely heavily on double precision. It is also influenced by the fact that the retail price of the 1080Ti is an order of magnitude lower than the other used compute devices.

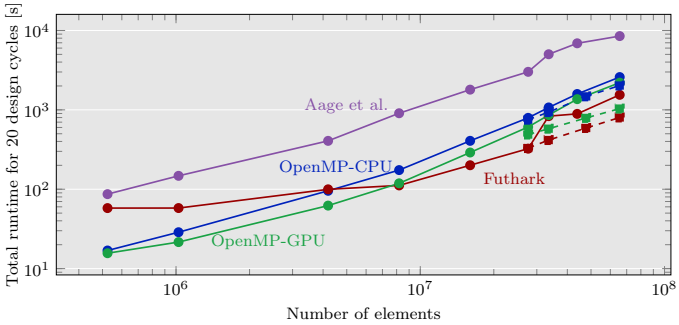


Figure 7: Runtimes for 20 design iterations of a linear elastic minumim compliance cantilever example with a volume fraction of $V^* = 0.1$. The optimisations were performed using a constant filter radius of 1/20 of the domain width. The blue and green lines indicate the OpenMP implementations for CPU and GPU respectively. The red line shows the Futhark implementation on the A100 GPU. All shown times are computed using 4 multigrid levels, with the exception of the additional points shown with dashed line and square marks, which are computed using 5 levels.

The performance comparison for the fixed filter size, as seen in fig. 7, results in less variation across mesh refinements, as the considered optimisation problem is constant. However, the cost of applying the filter grows with the filter radius, to a point where applying the filters and updating the densities account for approximately 10% of the runtime, for the Futhark implementation with 65.5 million elements, where the filter radius corresponds to 8 elements.

For the comparison with fixed filter size, we have also timed the implementation from Aage et al. [1] based on MPI and PETSc. This test is performed on the same machine as the OpenMP-CPU implementation. It should be noted that this implementation is designed for running on multiple compute nodes on a cluster, although this capability is not used for the present comparison. The problem solved is slightly different, as the PDE-based filter is used. The reason for this difference is that the PETSc implementation assembles the filtering matrix explicitly in memory, resulting in too high computation times and memory usage for large filter radii. By choosing the PDE-based filter, the practical difference for the timing results is minimised, as most of the reported time is spent solving the linear elastic equations. We have changed the solver settings on the coarsest grid from the default SOR preconditioned GMRES, to a direct solution by LU factorisation, to better match the used solver settings used in the other implementations. This change in solver settings improved the performance of the PETSc based implementation for all considered mesh sizes.

It can be seen that the PETSc based implementation is overall slower than the presented OpenMP-CPU, when running on a single desktop machine. This is expected, as the implementation is designed to solve relatively small systems at every MPI rank, scaling with more MPI ranks for larger problems, e.g. by explicitly assembling all matrices.

For both computational experiments it can be seen that the Futhark implementation has an increase in used runtime for 33M elements and above, most notably in the variable filter radius case. This is likely due to a change of used GPU kernel. Futhark compiles several kernels, and chooses which to launch at runtime based on the input size.

A relative performance comparison for the fixed filter size is shown in fig. 8. The wall-clock time of the OpenMP-CPU implementation is scaled with the respective wall-clock times, to give an indication of the improvement. We chose OpenMP-CPU as the reference, since it is the fastest implementation without GPU-acceleration.

We see that the OpenMP-GPU implementation is fastest in the coarse case, while the Futhark implementation is significantly slower. Inspecting fig. 7 it becomes clear that the Futhark implementation is almost unaffected in the runtime until 8 million elements. This indicates an overhead in launching the GPU kernels, which heavily affects the total compute time for small grids. As the OpenMP based implementations are

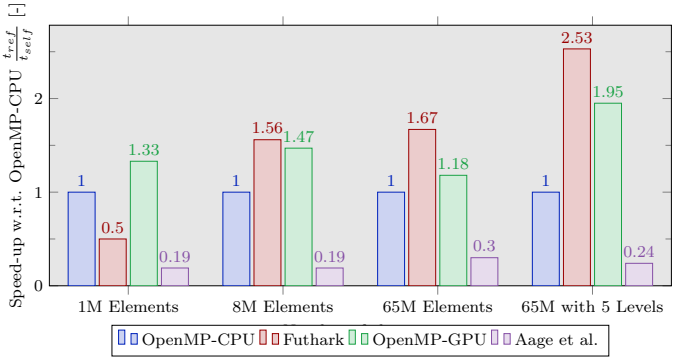


Figure 8: Relative runtime improvement over the OpenMP-CPU implementation for the first 20 iterations using 4 multigrid levels with constant filter radius. The three mesh refinements correspond to three vertical slices of fig. 7. Higher is better.

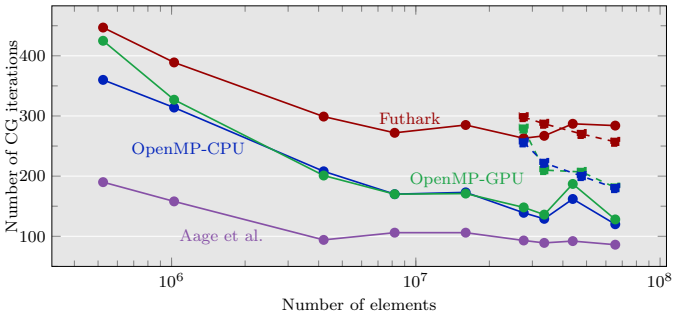


Figure 9: Sum of used iterations in the conjugate gradient solver for the first 20 design iterations. All shown times are computed using 4 multigrid levels, with the exception of the additional points shown with dashed line and square marks, which are computed using 5 levels.

able to complete in less time than the overhead, they outperform the Futhark implementation drastically for coarse problems.

Both GPU kernels perform well at the intermediate refinement of 8 million elements, while they suffer a slowing down in the finest meshes with 4 multigrid levels. As already discussed, we see that adding a fifth multigrid level improves the relative performance of the GPU accelerated implementations drastically due to different reasons. We note that adding a fifth multigrid level only slightly improves the runtime of OpenMP-CPU, as seen by the blue dashed lines with square marks in fig. 7. The large difference in relative performance when using 4 or 5 multigrid levels for 65M elements is thus explained by a small improvement in runtime for the OpenMP-CPU implementation, along with a drastic drop in runtime for both GPU accelerated implementations.

Interpreting the difference in total runtimes of the presented implementations is not straightforward, as the used multigrid preconditioners also vary. Some performance improvement is due to better utilisation of GPU hardware, some is due to a better suited preconditioner. In order to aid the comparison, fig. 9 shows the total amount of conjugate gradient iterations spent by the various implementations. The two OpenMP implementations use the same preconditioner, and can thus be compared directly, resulting in a improvement of a factor of 2-4 by using GPU acceleration. Figure 9 also shows very good agreement

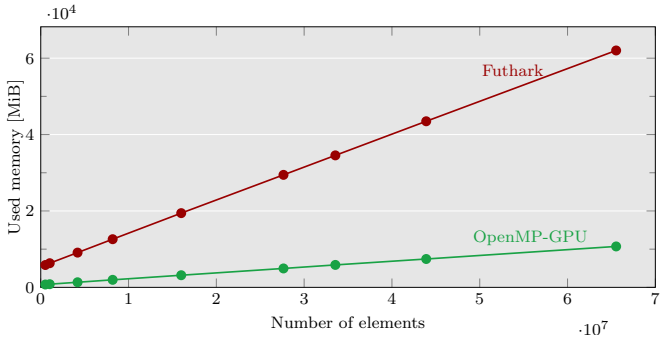


Figure 10: GPU memory usage of the Futhark and OpenMP-GPU implementations as a function of mesh refinement.

between the number of iterations used for these implementations. A small variations occurs between the two implementations, which can be attributed to the difference in floating point execution on the hardware. The Futhark implementation is able to achieve slightly better performance for large problems, even though it uses a coarse grid correction which results in more conjugate gradient iterations, as seen in fig. 9. Finally, it can be noted that the PETSc based implementation uses very few conjugate gradient iterations, even though it is the slowest implementation. This is due to the difference of implementation in the smoothing operations, which for the PETSc implementation reduce the error very effectively, but are not possible to implement efficiently for GPU accelerators. We can conclude that both presented approaches are viable for writing GPU accelerated topology optimisation implementations, specially considering the relative ease of implementation.

When increasing the number of multigrid levels, the number of solver iterations increase for the OpenMP based implementations, while remaining somewhat stationary for the Futhark based implementation. Still, the performance improves for both GPU accelerated implementations. In the case of the OpenMP-GPU implementation the amount of data which needs to be transferred between GPU and main memory every iteration is drastically reduced, which could explain the much improved relative performance. Also, the coarse space correction which is performed on the CPU is reduced in complexity as the number of levels increase. Hence, a larger fraction of computing is performed on the GPU. For the Futhark implementation, this decrease of computation time is likely due to a reduction in the computational effort required to solve the coarse grid correction on the GPU.

We note that the performances of the presented GPU codes shown in figs. 6 and 7 are not necessarily the fastest available. Some other works exploit the structure of topology optimisation problems to achieve high performance with multigrid preconditioning, e.g. Wu et al. [41] and Martínez-Frutos et al. [23]. These works also use lower-level implementations with better control of memory placement and other factors which greatly affect performance. However, as these codes are not publicly available it is not possible to quantify the performance gap between these and the presented implementations.

Memory usage. GPU memory is a very limited and costly resource. Most current cards have memory in the range of 8-16 GiB, while only few expensive cards designed for GPGPU come with more memory. One example of this is the Futhark implementation tested on a 1080Ti card with 8GiB, shown in fig. 6. The implementation runs out of memory when run with more than 8 million elements, which is why the curve stops earlier than the others.

The GPU memory usage of the two GPU implementations is shown in fig. 10. Here we see a clear difference between the two approaches, although they both grow linearly in the amount of memory used. The Futhark implementation uses significantly more memory across all mesh sizes. This is a consequence of the incremental flattening strategy implemented by the compiler to automatically transform nested parallelism. This memory usage is the reason that 65 million elements is the limit for the Futhark implementation when using the A100 GPU. The OpenMP-GPU implementation has much lower memory usage, enabling it to run

on cards with less memory, and solve larger problems. Here, the main bottleneck is transferring the coarse solution to the CPU every V-cycle. While the Futhark implementation uses much more memory than the OpenMP-GPU implementation, we note that it is still able to fit problems with 65 million elements onto a single GPU, which is sufficient for many use-cases.

5. Extension to Nonlinear Elasticity

To demonstrate the relative ease with which the presented GPU frameworks can be extended beyond the trivial linear elasticity problems, this sections discuss how to incorporate nonlinearity. Remark that there is only one nonlinear elastic implementation included, which is based on Futhark due to ease of implementation. The interested reader may find the implementation at doi:10.5281/zenodo.7791869.

Nonlinear matrix-free approach. It should be noted that the nonlinearity of the problem requires that the local matrices are numerically integrated for every matrix-vector product, as the tangent matrix is not stored in memory. This results in a large increase in computational work compared to the linear case, where a pre-computed local matrix was used to enable fast matrix vector products. Due to this change, an element-wise strategy was adopted for the matrix-vector product, to ensure that every local element matrix needs only to be computed once. In this matrix-free approach all element contributions are computed independently, and the finite element assembly of the nodal values are computed using a generalized histogram implemented in the Futhark core language [13]. That is, instead of using a formulation as seen in eq. (11), the assembly follows

$$f^e = E_e K^e u^e \tag{13}$$

for every element, followed by a local-to-global assembly of the resulting element forces f^e.

As an example, the Futhark linear implementation takes 1.316ms for a single matrix-vector product, while the nonlinear case takes 1075.4ms, using 2 million elements on the A100 GPU. While this may seem discouraging, it is still possible to perform topology optimisation using this approach, although the presented topology optimisation examples presented in this work do not go beyond 1 million elements.

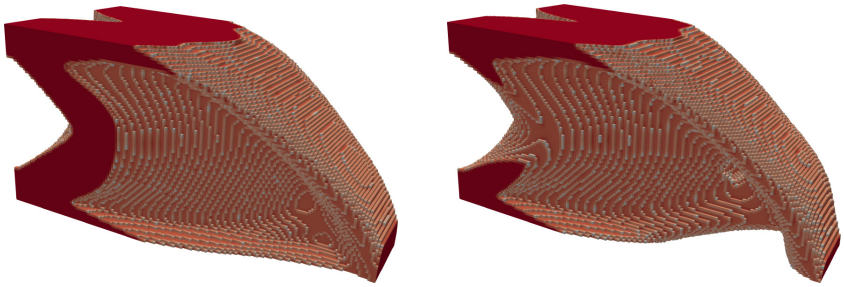
Newtons method. A Newton-Krylov method with a backtracking line-search is employed for the nonlinear problem [25]. A sequence of linearisations are solved, where each linearisation is approximately solved using the presented multigrid-preconditioned conjugate gradient method. The Newton method is built on top of the modified linear solver in Futhark. Due to the ease of composition in the functional language the implementation of the Newton method itself is very compact.

Load continuation for the nonlinear problem. The initial design iterations of the nonlinear problem can sometimes experience convergence issues, due to a low stiffness for the entire structure compared to the magnitude of the load. In order to resolve this, several approaches are possible. One is to create a continuation on the stiffness penalisation parameter p discussed in section 2.2, beginning with a linear interpolation, and incrementally increasing the value of p to finally obtain the desired penalisation. Another approach, which was used in this work, is to use a reduced load for the initial design iterations, linearly increasing the load over the first 20 design iterations, until the desired load is reached at iteration 20. Here the calling C code modifies the force vector passed into the Futhark methods, in order to implement this load continuation.

Cantilever. The cantilever example is revisited for the nonlinear elastic problem. The example is computed on a domain of $3m \times 1m \times 1m$ domain for a material with $E = 1$Pa and $\nu = 0.3$. A load of 0.002N is applied to the cantilever.

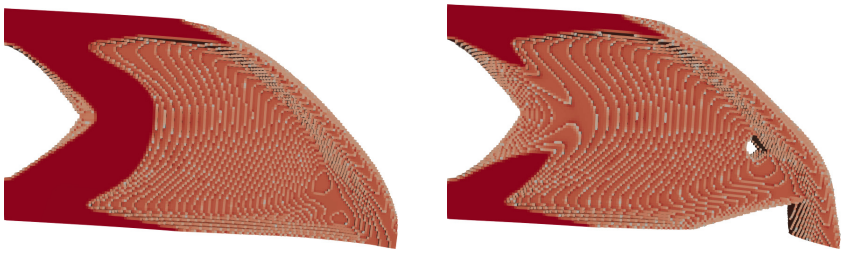
From fig. 11 it can be seen that the structure found using the nonlinear formulation is quite different from its linear counterpart. The nonlinear cantilever does not have a plate on the bottom connecting the support to the loading, as the structure is able to use the reorientation of the hanging beam to decrease the end-compliance. This is consistent with the findings of Buhl et al. [7], where similar hanging features are found. The nonlinear structure took approximately 14 hours to compute, which can be compared to its linear counterpart, which takes minutes to complete.

Figure 12 shows the deformed configurations of the cantilever designs, in the nonlinear formulation. It can be seen that for the nonlinear design the beam connecting the load reorients, such that it is in tension under loading. This effectively shortens the cantilever, and allows for a lower end-compliance.



(a) Linear. (b) Nonlinear.

Figure 11: Cantilever computed with 1 million elements, a volume fraction of 0.3, a filter radius of 1.5 elements, and a load of 0.002N.

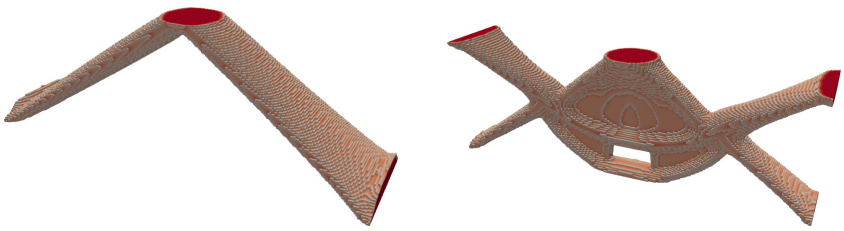


(a) Linear. (b) Nonlinear.

Figure 12: Deformed configurations of the designs from fig. 11, computed with the nonlinear elasticity. Elements with density below 0.5 are removed for visualization.

	Linear compliance	Nonlinear compliance
Linear design	$\mathbf{3.82 \times 10^{-4}\,Nm}$	$3.62 \times 10^{-4}\,\mathrm{Nm}$
Nonlinear design	$4.13 \times 10^{-4}\,\mathrm{Nm}$	$\mathbf{3.62 \times 10^{-4}\,Nm}$

Table 1: Comparison of compliance values for the two designs shown in fig. 11, under both linear and nonlinear elasticity.



(a) Linear. (b) Nonlinear.

Figure 13: Double clamped beam example computed with 768 000 elements, a volume fraction of 0.1, a filter radius of 2.5 elements, and a load of 0.00015N.

Table 1 shows a cross-check of the linear elastic and end-compliance values for the two cantilever designs. It can be seen that the linear design has a much lower linear compliance compared to the nonlinear design. This is expected due to the bottom plate, which greatly increases the stiffness in the linear regime. The nonlinear end-compliance of the two structures is almost identical, with the nonlinear optimisation being slightly lower, than the linear counterpart (on the 4th decimal).

Double clamped beam. The double clamped beam from Buhl et al. [7] is interpreted in 3D and included here. The design domain shown in fig. 3c shows the modelled domain with a symmetry condition shown in blue. The structure is discretised using $120 \times 80 \times 80$ (768 000) elements. The topology optimisation is performed with a filter radius of 2.5 elements, and a volume fraction of 10%. The nonlinear end-compliance is optimised for a load of 0.00015 newton, on a domain of $1.5 \times 1 \times 1$ meters, using Young's module 1 Pa for the solid.

The linear result is shown in fig. 13a, where it can be seen that the linear analysis makes a simple truss structure with two straight bars. While this structure is very stiff for small deformations, it will buckle if the applied load becomes too high. The nonlinear structure shown in fig. 13b is different from the linear structure, as it has two bars connecting to the support on either side of the loaded surface. The upper bars stabilise the structure in regards to buckling as they are loaded in tension and do not buckle. The central part of the structure has a vertical plate underneath the loaded surface, which is supported by the trusses where the structure hangs from. The upper part of the structure is similar to the two dimensional example provided in Buhl et al. [7], while the lower structure is similar to the linear result.

6. Discussion

The three implementations for compliance minimisation of linear elasticity show that it is possible to write simple but efficient GPU codes without resorting to explicitly allocating memory and writing separate kernels, as done in e.g. CUDA. The slowest implementation, using OpenMP without GPU acceleration, is able to solve optimisation problems with 65 million elements in approximately 3.15 hours on a desktop machine. This is already a large-scale problem, of a scale that usually requires a high performance computing cluster. The GPU accelerated implementations both cut this time to 2 hours when using a single Nvidia A100 GPU. This shows that efficient large-scale topology optimisation is not restricted to highly tuned implementations, and can be obtained with high levels of abstraction.

It is especially noteworthy that the Futhark based implementation performs as fast as it does, even though it only employs an approximate solver for the coarse space correction. The Futhark language allows

for a concise implementation, which is in contrast to the C implementation where an efficient implementation is seldom the simplest. For instance the overhead of manually tracking indexation between grids accounts for a significant additional amount of code and complexity in the C implementations. This is in large part thanks to the work done on the Futhark compiler, which to our experience emits efficient GPU kernels, with little tuning to the Futhark code.

The implementations rely heavily on the structured nature of the grid. Specifically, the local element matrices are all identical to some scaling, since all elements have identical geometry. This allows for one single element matrix to be integrated offline, and reused for the application of all elements, circumventing the need for numerical integration when applying the stiffness matrix. It might be possible to rewrite the linear implementations for unstructured grids, but not without a significant drop in performance. Another approach to handle arbitrary design domains could be to mesh the entire bounding box, and include passive void domains to restrict the design to the desired domain, as done in Aage et al. [2]. While it might seem counter-intuitive, it might be best for performance to model void elements around a complex domain. The performance improvements obtained from the simplified indexing and similar element matrices, could outweigh the cost of including passive void elements.

Other works such as Wu et al. [41] and Liu et al. [20] both use a structured mesh, but avoid working with the passive void elements arising from embedding a non-trivial design domain. As further work, the presented codes can be extended to enable non-trivial domains by allowing to remove passive void elements. Another option is to include a classic hard-kill strategy, where void elements are removed throughout the optimisation iterations. However, one should take care with updating the used elements, as indexing and non-trivial memory access very quickly becomes prohibitively costly on GPU hardware, potentially negating the performance improvements made by avoiding computation on some elements.

It has also been shown that it is possible to solve nonlinear problems with a million degrees of freedom on the GPU in a reasonable time-frame, although some work is needed before large-scale applications become possible. The nonlinear formulation requires numerical integration of all elements every time the stiffness matrix is needed. This makes the used matrix-free approach more computationally costly, compared to the linear problem, as the elements need to be numerically integrated every time the tangent matrix is applied. One possible alleviation is to explicitly assemble the full tangent matrix for the nonlinear problem, although this would greatly increase the memory consumption of the program. This would be feasible for the examples considered in this article, as they are no larger than 1 million elements, but could become infeasible for larger meshes as they would no longer fit in the GPU memory.

To summarise, GPU acceleration is now at the point where it is feasible to solve large-scale linear elastic topology optimisation problems on a single desktop system. High level languages and compilers like Futhark simplify the implementation process even further. It is our hope that the provided codes may serve as a basis for future research in topology optimisation.

7. Acknowledgements

The authors would like to acknowledge support from the Villum Foundation through the Villum Investigator Project InnoTop.

This project has further received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, United Kingdom, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, Republic of North Macedonia, Iceland, Montenegro.

References

[1] Aage, N., Andreassen, E., Lazarov, B.S., 2015. Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. Structural and Multidisciplinary Optimization 51, 565–572. doi:doi:10.1007/s00158-014-1157-0.

[2] Aage, N., Andreassen, E., Lazarov, B.S., Sigmund, O., 2017. Giga-voxel computational morphogenesis for structural design. Nature 550, 84–86. doi:doi:10.1038/nature23911.
[3] Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B.S., Sigmund, O., 2011. Efficient topology optimization in matlab using 88 lines of code. Structural and Multidisciplinary Optimization 43, 1–16. doi:doi:10.1007/s00158-010-0594-7.
[4] Baandrup, M., Sigmund, O., Polk, H., Aage, N., 2020. Closing the gap towards super-long suspension bridges using computational morphogenesis. Nature Communications 11, 2735. doi:doi:10.1038/s41467-020-16599-6.
[5] Bendsøe, M., Sigmund, O., 2003. Topology Optimization: Theory, Methods, and Applications. Springer Berlin Heidelberg.
[6] Bourdin, B., 2001. Filters in topology optimization. International Journal for Numerical Methods in Engineering 50, 2143–2158. doi:doi:10.1002/nme.116.
[7] Buhl, T., Pedersen, C., Sigmund, O., 2000. Stiffness design of geometrically nonlinear structures using topology optimization. Structural and Multidisciplinary Optimization 19, 93–104. doi:doi:10.1007/s001580050089.
[8] Chandra, R., Dagum, L., Kohr, D., Menon, R., Maydan, D., McDonald, J., 2001. Parallel programming in OpenMP. Morgan kaufmann.
[9] Chen, Y., Davis, T.A., Hager, W.W., Rajamanickam, S., 2008. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. ACM Trans. Math. Softw. 35. doi:doi:10.1145/1391989.1391995.
[10] Cook, R.D., Malkus, D.S., Plesha, M.E., 2001. Concepts and applications of finite element analysis. 4th ed ed., Wiley, New York, NY.
[11] Farber, R. (Ed.), 2017. Parallel Programming with OpenACC. Morgan Kaufmann, Boston. doi:doi:10.1016/B978-0-12-410397-9.09988-1.
[12] Henriksen, T., Elsman, M., 2021. Towards size-dependent types for array programming, in: Proceedings of the 7th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming, Association for Computing Machinery, New York, NY, USA. p. 1â€“14. doi:doi:10.1145/3460944.3464310.
[13] Henriksen, T., Hellfritzsch, S., Sadayappan, P., Oancea, C., 2020. Compiling generalized histograms for gpu, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press. pp. 1–14. doi:doi:10.1109/SC41405.2020.00101.
[14] Henriksen, T., Serup, N.G.W., Elsman, M., Henglein, F., Oancea, C.E., 2017. Futhark: Purely functional gpu-programming with nested parallelism and in-place array updates, in: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM, New York, NY, USA. pp. 556–571. doi:doi:10.1145/3062341.3062354.
[15] Henriksen, T., Thorøe, F., Elsman, M., Oancea, C., 2019. Incremental flattening for nested data parallelism, in: Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming, ACM, New York, NY, USA. pp. 53–67. doi:doi:10.1145/3293883.3295707.
[16] Herrero-Pérez, D., Martínez Castejón, P.J., 2021. Multi-gpu acceleration of large-scale density-based topology optimization. Advances in Engineering Software 157–158, 103006. doi:doi:10.1016/j.advengsoft.2021.103006.
[17] Intel, 2022a. Intel xeon® platinum 8160 processor specification. URL: `https://ark.intel.com/content/www/us/en/ark/products/120501/intel-xeon-platinum-8160-processor-33m-cache-2-10-ghz.html`. accessed June 20, 2022.
[18] Intel, 2022b. Intel xeon® w-3335 processor specification. URL: `https://ark.intel.com/content/www/us/en/ark/products/217244/intel-xeon-w3335-processor-24m-cache-up-to-4-00-ghz.html`. accessed June 20, 2022.
[19] Klarbring, A., Strömberg, N., 2013. Topology optimization of hyperelastic bodies including non-zero prescribed displacements. Structural and Multidisciplinary Optimization 47, 37–48. doi:doi:10.1007/s00158-012-0819-z.
[20] Liu, H., Hu, Y., Zhu, B., Matusik, W., Sifakis, E., 2019. Narrow-band topology optimization on a sparsely populated grid. ACM Transactions on Graphics 37, 1–14. doi:doi:10.1145/3272127.3275012.
[21] Liu, J., Xian, Z., Zhou, Y., Nomura, T., Dede, E.M., Zhu, B., 2022. A marker-and-cell method for large-scale flow-based topology optimization on gpu. Structural and Multidisciplinary Optimization 65, 125. doi:doi:10.1007/s00158-022-03214-z.
[22] Martínez-Frutos, J., Martínez-Castejón, P.J., Herrero-Pérez, D., 2015. Fine-grained gpu implementation of assembly-free iterative solver for finite element problems. Computers & Structures 157, 9–18. doi:doi:10.1016/j.compstruc.2015.05.010.
[23] Martínez-Frutos, J., Martínez-Castejón, P.J., Herrero-Pérez, D., 2017. Efficient topology optimization using gpu computing with multilevel granularity. Advances in Engineering Software 106, 47–62. doi:doi:10.1016/j.advengsoft.2017.01.009.
[24] Nguyen, T.H., Paulino, G.H., Song, J., Le, C.H., 2010. A computational paradigm for multiresolution topology optimization (mtop). Structural and Multidisciplinary Optimization 41, 525–539. doi:doi:10.1007/s00158-009-0443-8.
[25] Nocedal, J., Wright, S.J., 2006. Numerical Optimization. 2e ed., Springer, New York, NY, USA.
[26] NVIDIA, a. Nvidia cuda example - convolution separable. `https://github.com/NVIDIA/cuda-samples/blob/master/Samples/2_Concepts_and_Techniques/convolutionSeparable/convolutionSeparable.cu`. Accessed: 2023-02-09.
[27] NVIDIA, b. Nvidia cuda example - convolution texture. `https://github.com/NVIDIA/cuda-samples/blob/master/Samples/2_Concepts_and_Techniques/convolutionTexture/convolutionTexture.cu`. Accessed: 2023-02-09.
[28] Nvidia, 2022. Nvidia a100 tensor core gpu. URL: `https://www.nvidia.com/en-us/data-center/a100/`. accessed June 20, 2022.
[29] Nvidia, 2022a. Nvidia geforce rtx 10 family. URL: `https://www.nvidia.com/en-gb/geforce/10-series/`. accessed June 20, 2022.
[30] Nvidia, 2022b. Nvidia geforce rtx 3080 family. URL: `https://www.nvidia.com/en-gb/geforce/graphics-cards/30-series/rtx-3080-3080ti/`. accessed June 20, 2022.
[31] NVIDIA, Vingelmann, P., Fitzek, F.H., 2020. Cuda, release: 10.2.89. URL: `https://developer.nvidia.com/cuda-toolkit`.
[32] Paszke, A., Johnson, D., Duvenaud, D., Vytiniotis, D., Radul, A., Johnson, M., Ragan-Kelley, J., Maclaurin, D., 2021. Getting to the point. index sets and parallelism-preserving autodiff for pointful array programming. doi:doi:10.48550/ARXIV.2104.05372.
[33] Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. 2nd Edition. doi:doi:10.2113/gsjfr.6.1.30, `arXiv:0806.3802`.

[34] Schmidt, S., Schulz, V., 2011. A 2589 line topology optimization code written for the graphics card. Computing and Visualization in Science 14, 249–256. doi:doi:10.1007/s00791-012-0180-1.
[35] Sigmund, O., 2001. A 99 line topology optimization code written in matlab. Structural and Multidisciplinary Optimization 21, 120–127. doi:doi:10.1007/s001580050176.
[36] Sigmund, O., Maute, K., 2013. Topology optimization approaches: A comparative review. Structural and Multidisciplinary Optimization 48, 1031–1055. doi:doi:10.1007/s00158-013-0978-6.
[37] Steuwer, M., Remmelg, T., Dubach, C., 2017. Lift: A functional data-parallel ir for high-performance gpu code generation, in: Proceedings of the 2017 International Symposium on Code Generation and Optimization, IEEE Press. p. 74–85.
[38] Stone, J.E., Gohara, D., Shi, G., 2010. Opencl: A parallel programming standard for heterogeneous computing systems. Computing in Science & Engineering 12, 66–73. doi:doi:10.1109/MCSE.2010.69.
[39] Wadbro, E., Berggren, M., 2009. Megapixel topology optimization on a graphics processing unit. SIAM Review 51, 707–721. doi:doi:10.1137/070699822.
[40] Wang, F., Lazarov, B.S., Sigmund, O., Jensen, J.S., 2014. Interpolation scheme for fictitious domain techniques and topology optimization of finite strain elastic problems. Computer Methods in Applied Mechanics and Engineering 276, 453–472. doi:doi:10.1016/j.cma.2014.03.021.
[41] Wu, J., Dick, C., Westermann, R., 2016a. A system for high-resolution topology optimization. IEEE Transactions on Visualization and Computer Graphics 22, 1195–1208. doi:doi:10.1109/TVCG.2015.2502588.
[42] Wu, J., Dick, C., Westermann, R., 2016b. A system for high-resolution topology optimization. IEEE Transactions on Visualization and Computer Graphics 22, 1195–1208. doi:doi:10.1109/TVCG.2015.2502588.
[43] Zegard, T., Paulino, G.H., 2013. Toward gpu accelerated topology optimization on unstructured meshes. Structural and Multidisciplinary Optimization 48, 473–485. doi:doi:10.1007/s00158-013-0920-y.
[44] Zienkiewicz, O.C., Taylor, R.L., 2005. The finite element method for solid and structural mechanics. Elsevier.

Appendix A. Nonlinear element formulation

This appendix aims to describe the implemented nonlinear formulation thoroughly.

Appendix A.1. Basic definitions

Given the undeformed configuration X, and some deformed configuration x, they can be related through a deformation u.

$$x = X + u, \tag{A.1}$$

From this we can find the deformation gradient f_{ij}.

$$f_{ij} = \frac{\partial x_i}{\partial X_j} = \delta_{ij} + \frac{\partial u_i}{\partial X_j}, \tag{A.2}$$

And the Cauchy-Green tensor

$$C_{ij} = f_{ik} f_{kj}, \tag{A.3}$$

From this the Green-Lagrange strain tensor is defined

$$\epsilon_{ij} = \frac{1}{2}(C_{ij} - \delta_{ij}). \tag{A.4}$$

Appendix A.2. Neo-Hookean material model

The used Neo-Hookean energy funciton is

$$\Pi_{int} = \frac{\lambda}{2}(\ln J)^2 + \frac{\mu}{2}(C_{ii} - 3) - \mu \ln J, \quad J = |f_{ij}| \tag{A.5}$$

note that J denotes the determinant of the deformation gradient.

Here the lame parameters are used

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}, \tag{A.6}$$

The second Piola-Kirchhoff stress for the Neo-Hookean energy is

$$\sigma_{ij} = \frac{\partial \Pi_{int}}{\partial \epsilon_{ij}} = \lambda \ln J C^{-1}_{ij} + \mu(\delta_{ij} - C^{-1}_{ij}), \tag{A.7}$$

And the elasticity tensor is

$$C_{ijkl} = \frac{\partial \Pi_{int}}{\partial \epsilon_{ij} \partial \epsilon_{kl}} = \lambda C^{-1}_{ij} C^{-1}_{kl} + (\mu - \lambda \ln J)(C^{-1}_{ik} C^{-1}_{jl} + C^{-1}_{il} C^{-1}_{jk}). \tag{A.8}$$

Appendix A.3. Finite element discretization of Neo-Hookean material model

Assuming basic FE knowledge and Cook-like notation.

$$[D]_{ref} = \begin{bmatrix} \frac{\partial u^1}{\partial \xi^1} & \frac{\partial u^2}{\partial \xi^1} & \frac{\partial u^3}{\partial \xi^1} \\ \frac{\partial u^1}{\partial \xi^2} & \frac{\partial u^2}{\partial \xi^2} & \frac{\partial u^3}{\partial \xi^2} \\ \frac{\partial u^1}{\partial \xi^3} & \frac{\partial u^2}{\partial \xi^3} & \frac{\partial u^3}{\partial \xi^3} \end{bmatrix} = \begin{bmatrix} \sum_i N_{i,\xi^1} u^1_i & \sum_i N_{i,\xi^1} u^2_i & \sum_i N_{i,\xi^1} u^3_i \\ \sum_i N_{i,\xi^2} u^1_i & \sum_i N_{i,\xi^2} u^2_i & \sum_i N_{i,\xi^2} u^3_i \\ \sum_i N_{i,\xi^3} u^1_i & \sum_i N_{i,\xi^3} u^2_i & \sum_i N_{i,\xi^3} u^3_i \end{bmatrix} \tag{A.9}$$

$$[D] = \begin{bmatrix} \frac{\partial u^1}{\partial x^1} & \frac{\partial u^2}{\partial x^1} & \frac{\partial u^3}{\partial x^1} \\ \frac{\partial u^1}{\partial x^2} & \frac{\partial u^2}{\partial x^2} & \frac{\partial u^3}{\partial x^2} \\ \frac{\partial u^1}{\partial x^3} & \frac{\partial u^2}{\partial x^3} & \frac{\partial u^3}{\partial x^3} \end{bmatrix} = [J]^{-1}[D]_{ref} \tag{A.10}$$

$$[F] = [D] + [I], \quad J = |[F]| \tag{A.11}$$

$$[C] = [F]^\top [F] \tag{A.12}$$

$$[\epsilon] = \frac{1}{2}([C] - [I]) \tag{A.13}$$

$$[\sigma] = \lambda \ln |[F]| [C]^{-1} + \mu([I] - [C]^{-1}) \tag{A.14}$$

The elasticity tensor is renamed E, to avoid clash with the Cauchy-Green deformation tensor.

$$[E] = C_{ijkl} \quad \text{transformed to voigt notation} \tag{A.15}$$

Appendix A.4. Finite element discretization of strains

The following write-up of the Green-Lagrange strains is based on Zienkiewicz and Taylor [44]
The Green-Lagrange strains can be written as

$$\{\epsilon\} = \{\epsilon_0\} + \{\epsilon_L\} \tag{A.16}$$

where $\{\epsilon_0\}$ are the usual linear strains

$$\{\epsilon_0\} = [B_0]\{u\}, \tag{A.17}$$

And $\{\epsilon_L\}$ denote the additional Lagrangian strains,

$$\{\epsilon_L\} = \frac{1}{2}[A]\{\Theta\} = \frac{1}{2} \begin{bmatrix} \{\Theta_x\}^\top & 0 & 0 \\ 0 & \{\Theta_y\}^\top & 0 \\ 0 & 0 & \{\Theta_z\}^\top \\ 0 & \{\Theta_z\}^\top & \{\Theta_y\}^\top \\ \{\Theta_z\}^\top & 0 & \{\Theta_x\}^\top \\ \{\Theta_y\}^\top & \{\Theta_x\}^\top & 0 \end{bmatrix} \begin{Bmatrix} \{\Theta_x\} \\ \{\Theta_y\} \\ \{\Theta_z\} \end{Bmatrix} \tag{A.18}$$

here $\{\Theta_x\}^\top = \{\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}, \frac{\partial w}{\partial x}\}$, $\{\Theta_y\}^\top = \{\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial y}\}$, and $\{\Theta_z\}^\top = \{\frac{\partial u}{\partial z}, \frac{\partial v}{\partial z}, \frac{\partial w}{\partial z}\}$. Which can be computed similarly to eq. (A.10).

Similarly, the Lagrange interpolation can be found by

$$[B_L] = [A][G] = [A]\,[[g_1][g_2][g_3][g_4][g_5][g_6][g_7][g_8]] \tag{A.19}$$

where

$$[g_i] = \begin{bmatrix} N_{i,x} & 0 & 0 \\ 0 & N_{i,x} & 0 \\ 0 & 0 & N_{i,x} \\ N_{i,y} & 0 & 0 \\ 0 & N_{i,y} & 0 \\ 0 & 0 & N_{i,y} \\ N_{i,z} & 0 & 0 \\ 0 & N_{i,z} & 0 \\ 0 & 0 & N_{i,z} \end{bmatrix} \tag{A.20}$$

Now the full interpolation is defined as

$$[\bar{B}] = [B_L] + [B_0] \tag{A.21}$$

Finally, the element residual and tangent matrix can be found as:

$$\{R_e\} = \int_{V^e} [\bar{B}]\{\sigma\} dV - \{P_e\} \tag{A.22}$$

$$[k^e] = \int_{V^e} [G]^\top [M][G] dV + \int_{V^e} [\bar{B}]^\top [C][\bar{B}] dV \tag{A.23}$$

where

$$[M] = \begin{bmatrix} \sigma_{11}[I_3] & \sigma_{12}[I_3] & \sigma_{13}[I_3] \\ \sigma_{12}[I_3] & \sigma_{22}[I_3] & \sigma_{23}[I_3] \\ \sigma_{13}[I_3] & \sigma_{23}[I_3] & \sigma_{33}[I_3] \end{bmatrix} \tag{A.24}$$

[P5]

Optimisation of Shape and Thickness of Shell Structures.

E. A. Träff, N. Aage, M. Langelaar, O. Sigmund and F. van Keulen.

Thin-Walled Structures.

[In preparation]

Simultaneous Optimisation of Shape and Thickness of Shell Structures.

E. A. Träff, N. Aage, M. Langelaar, O. Sigmund, F. van Keulen

December 2022

Abstract

This work presents a consistent formulation for concurrent optimisation of shape and thickness of thin-walled structures. The presented formulation, which is based on the principle of relocating the nodes, achieves consistency by using the initial finite element mesh as a reference for the updated coordinates. To ensure physical validity the coordinate changes are regularised by a filter and a constraint on an aggregation of mesh quality metrics. This resulting method is shown to robustly optimise thin-walled structures, and the benefits and limitations of the method through a series of simple examples.

1 Introduction

Thin-walled structures are a cornerstone in many engineering applications. Due to high stiffness-to-weight ratios, they appear in many constructions where low weight is desired, ranging from packaging to aircraft. Well designed shell structures can take form as e.g. increased fuel efficiency, reduced expenditure of material and resources, or decreased cost of production. Due to this ubiquity, there is a need for simple, versatile, and efficient automatic design tools.

The optimisation of shell structures is by no means a new topic. Some initial developments can be found in Ramakrishnan and Francavilla (1974) and Mohr (1979), which optimised shells of revolution. Mohr (1979) used facet elements which model the shell by piecewise flat elements combined with symmetry conditions, parametrising the shape of the shell by nodal relocations, an approach similar to current state-of-the-art (Antonau *et al.*, 2022).

Botkin (1982) also performed shape optimisation with facet elements, but introduced so-called "design-elements", which parametrise the design with a small number of control variables. Complex shell structures are considered in Gates and Accorsi (1993), where the shape of an existing design is parametrised by the movement of edges in the geometry, as control points for the underlying splines.

Eschenauer and Weinert (1992) consider parametrising the shape by interpolating between control points using piecewise linear functions, B-splines, and Bezier curves. Parametrising the design by nodal positions of the finite element mesh (as we do herein) can be considered as a special case of the piecewise linear functions, with a large number of control points. Ramm *et al.* (1993) also develops an approach based on interpolation between control points.

Free-form shape optimisation, considering nodal coordinates as design variables, has in the past decade received attention (Arnout *et al.*, 2012; Bletzinger, 2014; Hojjat *et al.*, 2014). These works propose that filtered nodal coordinates achieve both the necessary regularity, and high design freedom. Even more recent works are focused on improving the mathematical programming aspects and the introduction of variable filters, as summarised in Antonau *et al.* (2022). The method presented in this article may indeed be considered a continuation of the vertex morphing method of Hojjat *et al.* (2014).

Several of the works outlined above consider the simultaneous optimisation of thickness (Eschenauer and Weinert, 1992; Ramm *et al.*, 1993). There exists a close connection between optimising the thickness defined for every node or element and density based topology optimisation (Maute and Ramm, 1997).

This article presents a simple method for concurrent optimisation of the shape and thickness of shell structures. This form-finding method starts from an initial shell shape and thickness distribution, and improves the performance characteristics of the shell by solving a mathematical optimisation problem to find a change in shape and new thickness distribution. Such shape and thickness optimisation can be used as tools by engineers designing shell structures, to help guide towards the best possible structures.

The presented method is a novel combination of

several well-studied concepts: A parametrisation of shape by relocation of nodes in the finite element mesh (Bletzinger, 2014), a simple yet robust shell element formulation (Van Keulen and Booij, 1996), a filtering technique for relocation and thickness (Lazarov and Sigmund, 2011), an aggregated constraint on a measure of element qualities, all within a simple and consistent mathematical programming problem solved with a standard sequential convex programming technique (Svanberg, 1987). The volume-constrained linear elastic compliance minimisation problem is considered, although the composition presented here can be extended for other objectives and constraints.

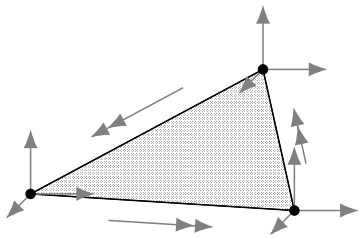


Figure 1: Sketch of degrees-of-freedom in local coordinates. One out-of-plane and two in-plane deformations at every node, and one rotation about every edge.

2 Shell Element Formulation

We use the simplest shell element formulation of the family developed in van Keulen (1993), also summarised in Van Keulen and Booij (1996). In this section, we summarise properties and implementation details of the formulation. The used linear variation of the shell element is described with more detail in appendix A.

The element formulation assumes that every element is a facet, i.e. a flat section, modelled with separate membrane and plate elements. The used membrane element is the well-known constant strain triangle (Zienkiewicz *et al.*, 2013). The bending is modelled by a constant bending element which uses three out-of-plane displacements and rotations about the element edges to characterise the bending (Morley, 1971). The resulting degrees-of-freedom used by the element are sketched in fig. 1. In this work we restrict the formulation to infinitesimal deformations, although the original formulation accounts for finite deformations.

Due to the faceted nature of the element employed, many elements are required to model complex shell curvature. Furthermore, constant thickness is assumed for every facet, requiring many elements also to correctly capture smooth transitions in thickness across the structure. Studies of the accuracy of the element are presented in van Keulen (1993); Van Keulen and Booij (1996).

An advantage of the formulation is that it does not require surface normals at the nodes of the mesh, as is the case for many shell formulations. This simplifies changing the shape of the finite element mesh, as there is no need to update surface normals. Another benefit is that the element formulation also trivially handles kinks in the shell surfaces and arbitrary intersections between multiple shell surfaces, without any drilling degrees-of-freedom or such. Although simple, the behaviour and deficiencies of the element are relatively well understood.

Slender structures, such as shells, are susceptible to issues with stability and large deformations. By choosing an infinitesimal displacement model of the shells, we must also restrict our use of the optimisation tool, and our interpretation of the optimised structures, to the regime where our underlying assumptions are met. The presented approach can in principle be extended to the finite deformation regime. Only the shell element formulation and compliance objective function need to be adapted. In practise, moving to a finite deformation model can complicate the optimisation, as the solving the non-linear state equations can fail if a proper load increment strategy is not chosen.

3 Parametrisation of Structure

The choice of design representation in structural optimisation is not always trivial, and it has a great effect on the usefulness of the final approach. A parametrisation which is too restrictive can result in structures which are far from mechanically optimal. A parametrisation which is too permissive can result in structures which are only performing well due to modelling errors. Here, we describe the chosen parametrisations, nodal relocations $\mathbf{x}_\mathrm{d}$ and nodal thickness $\mathbf{h}_\mathrm{d}$, and provide some reasoning behind the choices made.

3.1 Shape

The shape is parametrised by relocations of the original node positions of the mesh. These relocations are regularised by the application of a filter

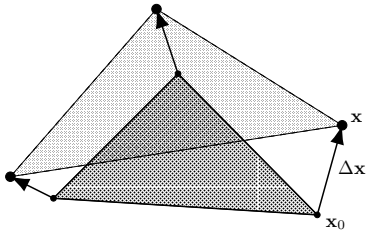


Figure 2: Sketch of updated nodal positions $\mathbf{x}$ defined as a relocation $\Delta\mathbf{x}$ from the initial mesh configuration $\mathbf{x}_0$.

and a constraint on a measure of the mesh quality, details in sections 3.1.1 and 3.1.2.

Given some design variables $\mathbf{x}_\mathrm{d} \in \mathbb{R}^{3n_n}$ we wish to find the resulting node positions of the finite element mesh $\mathbf{x} \in \mathbb{R}^{3n_n}$. Here n_n denotes the number of nodes in the finite element mesh, i.e. we have 3 relocation variables for every node in the finite element mesh. Conceptually, the relocation variables can be considered similar to the displacement variables of the elastic finite element problem.

First, the design variables are filtered to attain the filtered node relocations

$$\Delta\mathbf{x} = \mathbf{F}\,\mathbf{x}_\mathrm{d} \tag{1}$$

Here $\mathbf{F} : \mathbb{R}^{3n_n} \to \mathbb{R}^{3n_n}$ denotes the filter as a linear operator, details are given in section 3.1.1.

The filtered node relocations $\Delta\mathbf{x}$ are simply added to the the initial nodal coordinates in the finite element mesh $\mathbf{x}_0 \in \mathbb{R}^{3n_n}$

$$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x} \tag{2}$$

Note that the presented formulation treats the current mesh node positions $\mathbf{x}$ as a combination of the initial position $\mathbf{x}_0$ and some relocation defined by $\mathbf{x}_\mathrm{d}$. Also since $\mathbf{x}_\mathrm{d} = 0$ implies that $\Delta\mathbf{x} = 0$ due to the linearity of $\mathbf{F}$, we have that $\mathbf{x} = \mathbf{x}_0$. Therefore, $\mathbf{x}_\mathrm{d} = 0$ is always chosen as the initial value for the optimisation problem.

The main advantage of using nodal relocations to define the change in shape is a huge design-freedom, as the parametrisation is incredibly versatile. Another notable advantage is the simplicity, as updating the mesh with a nodal relocation can be done by relocating all node coordinates independently. One drawback of the parametrisation is the amount of design variables, which due to efficiency and practical considerations will limit the choice of optimisation algorithm to first-order methods. It is also crucial to retain a good finite element mesh in order to guarantee a sufficiently accurate solution of the elastic equations.

3.1.1 Regularisation by Filter

To regularise the resulting node relocations, a filtering strategy is applied. The used implicit filtering strategy was originally developed for densities in topology optimisation problems by Lazarov and Sigmund (2011). Recently, Asl and Bletzinger (2022) showed that the method can be applied on surfaces for shape optimisation. Furthermore, we note that the filtering process can be applied to intersecting and discontinuous surfaces, such as the one shown in fig. 3.

The implicit filter is defined by solving the following differential equation for each relocation component on the initial mesh configuration.

$$-r^2\nabla^2\Delta x^i + \Delta x^i = x^i_\mathrm{d} \quad \text{on } \Gamma_0 \tag{3}$$

Where $i \in \{1, 2, 3\}$ denotes the three spatial components of the relocations. Here Δx and x_d are considered continuous functions, defined on the initial geometry of the shell Γ_0.

The scalar r controls the filter radius, and is chosen a-priori as constant for the entire domain. Note that the filter radius r gives rise to a support of radius $r_\mathrm{supp} = 2\sqrt{3}r \approx 3.5r$ (Lazarov *et al.*, 2016). We only report and discuss the used support radius r_supp, as this gives a better physical interpretation of the filter radius.

The differential equation can be modified to include Dirichlet boundary conditions on a selected subset of the shell surface $\delta\Gamma^i_0$.

$$\Delta x^i = 0 \quad \text{on } \delta\Gamma^i_0 \tag{4}$$

This is used to define non-design regions, where the shape of the shell remains unchanged. The set $\delta\Gamma^i_0$ can be chosen independently for each relocation component i in order to constrain a point to a surface or line which is aligned with the standard basis. In principle, arbitrary constraints can be considered by solving all three relocation components as one system, and applying multi-point constraints across components.

The implicit filter is solved using a standard finite element approach. The element matrices $\mathbf{L}_e$ and $\mathbf{M}_e$ for the linear triangle are given in appendix B. These element matrices are assembled to the global matrices $\mathbf{L}$ and $\mathbf{M}$ respectively. The final discretised system is then

$$\left[r^2\mathbf{L} + \mathbf{M}\right]\Delta\mathbf{x}^i = \mathbf{M}\mathbf{x}^i_\mathrm{d} \tag{5}$$

Implicitly, the boundary conditions from eq. (4) are applied to $\left[r^2\mathbf{L}+\mathbf{M}\right]$. If the boundary conditions $\delta\Gamma_0^i$ are identical for the three coordinate components, the matrix $\left[r^2\mathbf{L}+\mathbf{M}\right]$ can be reused. Otherwise, three different matrices must be constructed, one for each relocation component.

The system matrix(es) $\left[r^2\mathbf{L}+\mathbf{M}\right]$ is symmetric and positive definite, and can thus be solved using Cholesky factorisation or conjugate gradients.

For completeness, the linear operator $\mathbf{F}$ discussed in section 3.1.1 can be formally stated as applying the linear operator $\mathbf{F}^i$ for every coordinate component $i \in \{1,2,3\}$

$$\mathbf{F}^i = \left[r^2\mathbf{L}+\mathbf{M}\right]^{-1}\mathbf{M} \tag{6}$$

3.1.2 Radius Ratio Constraint

In order to ensure that the finite element analysis of the elastic equations is sufficiently accurate at every iteration, a constraint on a quality measure of the elements is added to the optimisation problem.

The chosen quality measure is the so-called radius ratio ρ, which is defined as $\frac{R}{2r}$, where R and r denote the circum-radius and in-radius respectively. A discussion on this quality metric is given in Pébay and Baker (2003).

The radius ratio can be computed for a triangle using the edge-lengths a, b, c

$$\rho = \frac{R}{2r} = \frac{abc}{(b+c-a)(c+a-b)(a+b-c)} \tag{7}$$

The radius ratio has the properties that $\rho \geq 1$ always. The value $\rho = 1$ implies an equilateral triangle, while higher values of ρ imply more distortion of the triangle.

The radius ratios for all elements are aggregated to a single vale using the p-mean function.

$$\rho_{\text{agg}} = \left(\frac{1}{n_e}\sum_{i=1}^{n_e}{\rho_i}^p\right)^{\frac{1}{p}} \tag{8}$$

where p is a penalisation in the p-mean. As $p \to \infty$ the aggregation approaches the max operator, which is non-differentiable. ρ_i is used to denote the radius ratio ρ of element i. For this work the penalisation $p = 20$ is used unless otherwise stated.

It is important to note that the used aggregation provides no guarantees to the actual maximal value in the mesh, but rather constrains a mean value which behaves somewhat like the maximum. We note that the inexact nature of aggregation is not necessarily a big issue, as it is sufficient that the maximal radius ratio is only enforced in an average sense.

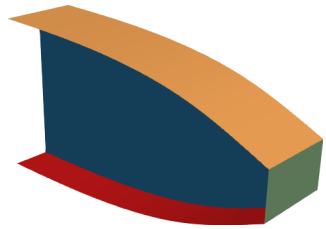


Figure 3: Example of shell structure composed of multiple surfaces. Here every surface is coloured distinctly.

3.2 Thickness

The shell element assumes a spatially constant thickness for every element. We choose a slightly different parametrisation for use in the optimisation approach, defining a thickness for every node on every surface. This means that if a node lies on several surfaces, e.g. fig. 3, it will have several thickness variables, one for each surface.

The nodal design thickness is filtered with the same process as the nodal relocations, section 3.1.1, on every surface.

$$-r^2\nabla^2 h_{\text{f}}^s + h_{\text{f}}^s = h_{\text{d}}^s \quad \text{on } \Gamma_0^s \tag{9}$$

where Γ_0^s denotes the surface s of an initial geometry. For the thickness, only the natural Neumann boundary condition is used. In practise, the exact same discretisation presented in section 3.1.1 is used for every surface. We denote the filter by the linear operator $\mathbf{F}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ which takes a vector of thickness on every surface, where n_h denotes the total number of nodal thicknesses, and applies the filter for every surface independently.

Every element thickness h_e is defined as the average value of the three coinciding node values of the filtered thickness $\mathbf{h}_{\text{f}} = \mathbf{F}^h\mathbf{h}_d$ on the corresponding surface.

$$h_e = \frac{1}{3}\sum_{i\in\mathcal{N}_e}\left[\mathbf{h}_{\text{f}}\right]_i^s \tag{10}$$

where $\mathcal{N}_e$ denotes the node indices of the three nodes associated with element e, and s denotes the surface in which element e lies, and $[\cdot]$ is used to denote accessing the corresponding component of the vector.

4 Optimisation Problem

In this work we consider compliance minimisation with a constraint on the amount of material used,

and the constraint on the mesh quality. Here we present the formal optimisation problem for optimising the shape and thickness of shell structures. First, some last pieces used in the formal optimisation problem are defined.

4.1 Linear Elastic Compliance

Linear elastic compliance is a natural objective for optimising the linear stiffness of a structure. The choice of linear modelling is discussed in detail in section 2.

The linear elastic compliance is denoted as

$$\mathcal{C} = \mathbf{u}^\top \mathbf{K}\mathbf{u} = \mathbf{u}^\top \mathbf{f} \tag{11}$$

Where $\mathbf{K}$ denotes the assembled stiffness matrix, and is implicitly dependent on the current nodal coordinates $\mathbf{x}$ and the element thickness $\mathbf{h}$. The force is denoted $\mathbf{f}$, which is possibly dependent on the nodal coordinates $\mathbf{x}$, if pressure loading is considered.

For shells the linear compliance can be decomposed into a contribution from the associated with bending $\mathcal{C}_b$ and a contribution from the strains associated with the membrane $\mathcal{C}_m$.

$$\mathcal{C} = \mathcal{C}_b + \mathcal{C}_m \tag{12}$$

Each contribution can be found for every element, by considering only the relevant degrees-of-freedom in the local coordinate system. I.e. the bending strain energy of an element with local deformations $\mathbf{u}_e$ is

$$\mathcal{C}_{b\,e} = A_e \mathbf{u}_e^\top \mathbf{T}^\top \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{B}_\mathrm{b}^\top \mathbf{C}_\mathrm{b} \mathbf{B}_\mathrm{b} \end{bmatrix} \mathbf{T}\mathbf{u}_e \tag{13}$$

Similarly, the membrane contribution is found by

$$\mathcal{C}_{m\,e} = A_e \mathbf{u}_e^\top \mathbf{T}^\top \begin{bmatrix} \mathbf{B}_\mathrm{m}^\top \mathbf{C}_\mathrm{m} \mathbf{B}_\mathrm{m} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{T}\mathbf{u}_e \tag{14}$$

The global quantities $\mathcal{C}_b, \mathcal{C}_m$ can then be found by summation of element contributions.

4.2 Volume Constraint

When optimising the thickness of a shell structure for stiffness it is usually very useful to consider the total volume as either the objective or a constraint. This stems from the property that adding material will always result in increased stiffness. If there are no limitations on the amount of material, then the resulting structure will thus have maximal thickness everywhere. One exception is when a volume dependent load is considered, such as gravity, it is no longer clear that maximum thickness everywhere is optimal.

The total volume of the domain is defined as

$$v = \sum_{i=1}^{n_e} \int_{S_e} h_i dS_e = \sum_{i=1}^{n_e} h_i A_i \tag{15}$$

where the element thickness h_i depend on the thickness representation, and the element areas A_i depend on the mesh relocation x_d.

In order to ease the conceptual use of the constraint, it is normalised to a fraction of the initial volume

$$v^* = \frac{v}{v_0} \tag{16}$$

where v^* is here called the volume fraction, and v_0 is the value of v in the initial configuration of the design.

4.3 Problem Formulation

Now we state the considered formal optimisation problem in eq. (17). We note that some definitions and interdependencies are left implicit, to keep the problem statement somewhat concise. These implicit details are clarified below.

$$\underset{\mathbf{x}_\mathrm{d}, \mathbf{h}_\mathrm{d}}{\text{minimise}} \qquad \mathcal{C} = \mathbf{u}^\top \mathbf{K}\mathbf{u} \tag{17a}$$

subject to

$$\text{state equation} \qquad \mathbf{K}\mathbf{u} = \mathbf{f} \tag{17b}$$

$$\text{filters} \qquad \Delta\mathbf{x} = \mathbf{F}\mathbf{x}_\mathrm{d} \tag{17c}$$

$$\mathbf{h}_\mathrm{f} = \mathbf{F}^h \mathbf{h}_\mathrm{d} \tag{17d}$$

$$\text{volume} \qquad v^* \leq V^* \tag{17e}$$

$$\text{radius ratio} \qquad \rho_\mathrm{agg} \leq \rho_\mathrm{lim} \tag{17f}$$

$$\text{bounds} \qquad \mathbf{x}_\mathrm{low} \leq [\mathbf{x}_\mathrm{d}]_i \leq \mathbf{x}_\mathrm{upp}\ \forall i \tag{17g}$$

$$\mathbf{h}_\mathrm{low} \leq [\mathbf{h}_\mathrm{d}]_i \leq \mathbf{h}_\mathrm{upp}\ \forall i \tag{17h}$$

The design variables stated in eq. (17a) are defined as vectors of all nodal relocations $\mathbf{x}_\mathrm{d} \in \mathbb{R}^{3n_n}$ and nodal-surface thickness $\mathbf{h}_\mathrm{d} \in \mathbb{R}^{n_h}$. The notation $[\cdot]_i$ is used to denote value of the ith index in a vector.

The system matrix $\mathbf{K}$ in eq. (17b) depends on $\Delta\mathbf{x}$ and the elementwise thickness, derived implicitly from $\mathbf{h}_\mathrm{f}$ by eq. (10). Likewise the force vector $\mathbf{f}$ can potentially depend on the nodal relocations $\Delta\mathbf{x}$.

The volume fraction limit V^* and radius ratio measure limit ρ_lim are both user choices. They are chosen as $V^* = 1$ and $\rho_\mathrm{lim} = 1.1$ by default.

The bounds or box constraints shown in eqs. (17g) and (17h) are implicitly defined for every element in the two design vectors $\mathbf{x}_\mathrm{d}$ and $\mathbf{h}_\mathrm{d}$.

The bound values of the relocations $\mathbf{x}_\mathrm{low}, \mathbf{x}_\mathrm{upp}$ are user choices. They can be set to very large to allow arbitrary relocation, or can be set close to 0 to disallow large change in position of the mesh nodes. It is also possible to specify variable relocation bounds for every coordinate, to ensure that the resulting shape lies within a predefined coordinate aligned box.

The bounds values for the shell thickness $\mathbf{h}_\mathrm{low}, \mathbf{h}_\mathrm{upp}$ are also user choices. The upper bound should not be excessively thick, as it could possibly break the assumptions of the underlying shell theory, which breaks the accuracy of the finite element model, which in turn breaks the physical interpretation of the optimisation problem. The lower bound of the thickness can be set to a value orders of magnitude lower than the initial values and upper bounds, to mimic the removal of that shell section. This is inspired by the techniques used in topology optimisation (Bendsøe and Sigmund, 2004).

The problem formulation stated in eq. (17) can be simplified to a pure shape or thickness optimisation problem. If optimising only for shape, the design variable $\mathbf{h}_\mathrm{d}$ is removed from eq. (17a), and eq. (17h) is removed. If optimising only for thickness the design variable $\mathbf{x}_\mathrm{d}$ is removed from eq. (17a), along with eqs. (17e) to (17g).

All examples presented later in this work are solved using the Method of Moving Asymptotes (Svanberg, 1987), which is a method commonly used for topology optimisation. The choice of optimisation algorithm can have a large effect on the resulting designs, and it is not clear that the Method of Moving Asymptotes is the best method for solving shape-optimisation problems. Nevertheless, we chose the Method of Moving Asymptotes due to its availability.

4.4 Gradients

In order to efficiently solve the optimisation problem, the gradients of the objective function $\mathcal{C}$ and constraints v^*, ρ_agg with respect to the design variables $\mathbf{x}_\mathrm{desgin}, \mathbf{h}$ are needed.

The partial derivative with respect to the relocation $\mathbf{x}_\mathrm{d}^i$ can be computed from a partial derivative with respect to relocation $\Delta\mathbf{x}^i$ by

$$\frac{\partial \cdot}{\partial \mathbf{x}_\mathrm{d}^i} = \mathbf{M}\left[r^2\mathbf{L} + \mathbf{M}\right]^{-1} \frac{\partial \cdot}{\partial \Delta\mathbf{x}^i}, \quad i \in \{1,2,3\} \tag{18}$$

where some transpositions have vanished due to the symmetry of $\mathbf{M}$ and $\left[r^2\mathbf{L} + \mathbf{M}\right]$.

From eq. (2) we can relate the gradients of the coordinates and the gradients of the filtered relocations.

$$\frac{\partial \cdot}{\partial \Delta\mathbf{x}} = \frac{\partial \cdot}{\partial \mathbf{x}} \tag{19}$$

This can be applied with eq. (18) to obtain the gradients with respect to the design variables used in the problem formulation.

The gradient of the objective function with respect to the coordinates and thickness can be found using the adjoint method, using the self-adjoint property of the problem (Bendsøe and Sigmund, 2004).

$$\frac{\partial \mathcal{C}}{\partial [\mathbf{x}]_i} = -\mathbf{u}^\top \frac{\partial \mathbf{K}}{\partial [\mathbf{x}]_i}\mathbf{u} + 2\mathbf{u}^\top \frac{\partial \mathbf{f}}{\partial [\mathbf{x}]_i} \tag{20}$$

Due to the locality of the partial derivatives the only non-zero contributions to $\frac{\partial \mathbf{K}}{\partial [\mathbf{x}]_i}$ and $\frac{\partial \mathbf{f}}{\partial [\mathbf{x}]_i}$ come from adjacent elements. This is also the case for the nodal thickness. Thus the gradient $\frac{\partial \mathcal{C}}{\partial \mathbf{x}}$ can be treated as a finite element assembly of the local terms of eq. (20). The gradients with respect to thickness are treated identically.

The local terms such as $\frac{\partial \mathbf{k}_e}{\partial [\mathbf{x}]_i}$ can be quite non-trivial to derive, such as the following expression

$$\frac{\partial \mathbf{k}_e}{\partial [\mathbf{x}]_i} = \frac{\partial}{\partial [\mathbf{x}]_i}\left(A_e T_e^\top B_e^\top C_e B_e T_e\right) \tag{21}$$

Where A_e, T_e, B_e are all dependent on every coordinate. Therefore, all local gradient terms are computed using automatic differentiation, sidestepping the necessity of derivation completely. The automatic differentiation is carried out using the autodiff library (Leal, 2018).

5 Examples

We present some examples of the shown method. The examples are ordered in increasing complexity, from simple sanity checks to more complex examples. The first examples, sections 5.1 to 5.4, study the effect of variations in some of the parameters used in the optimisation problem, in order to gain some understanding of their influence. The following examples, sections 5.5 to 5.8, show the method applied to more complex examples with pressure load, intersecting shells, or from the literature.

All examples have many parameters, which are not necessarily interesting for the interpretation of the results, but are crucial for their reproduction. These parameters are therefore summarised in table 1 in the appendix.

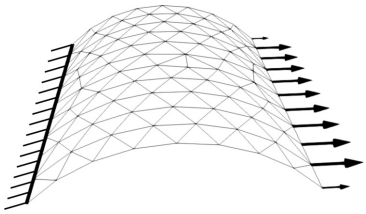


Figure 4: Initial geometry of curved sanity-check with boundary conditions and loads. The compliance of the initial structure is $\mathcal{C} = 3.34\,\mathrm{MJ}$ ($\mathcal{C}_m = 22\,\mathrm{kJ}, \mathcal{C}_b = 3.32\,\mathrm{MJ}$).

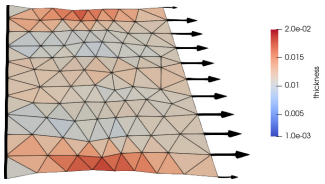


Figure 5: Resulting optimised structure with loads and line indicating boundary conditions. The resulting element thickness, used for the finite element analysis are shown by color. The used radius ratio constraint is $\rho_{\mathrm{lim}} = 1.1$, while the used filter is $r_{\mathrm{supp}} = 0.5$. The computed compliance for this structure is $\mathcal{C} = 512.7\,\mathrm{J}$ ($\mathcal{C}_m = 511.3\,\mathrm{J}, \mathcal{C}_b = 1.46\,\mathrm{J}$)

5.1 Curved Shell

Initially we consider a simple optimisation problem, on a curved shell. We consider an initial geometry which is the quarter of a cylinder of height 2m, radius 1m, and thickness 0.01m. One line is clamped, while the other is subjected to a distributed traction, oriented to be in the plane which contains the two lines, as shown in fig. 4. The surface is discretised using 162 triangles.

The two lines are forced to retain their original position through the boundary conditions of the relocation filter.

The resulting structure shown in fig. 5 is a flat plate. This is exactly as expected, as the membrane stiffness is high, compared to the bending stiffness. We note that there is still some bending in the resulting structure as $\mathcal{C}_b \neq 0$. It is possible to compute the analytic membrane compliance for a uniform-thickness flat plate, with the same volume as the considered example. We find the ideal compliance as $\mathcal{C} = \mathcal{C}_m = 509.6\,\mathrm{J}$, which is lower than our found structure. This difference can be ascribed to the curved boundaries of the plate, with higher thickness, which should not occur since $\nu = 0$. All-in-all, our approach passes the first sanity check.

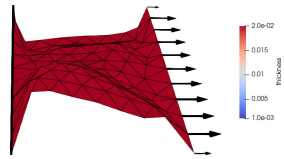


Figure 6: Resulting optimised structure with no constraint on the radius ratio $\rho_{\mathrm{lim}} = \infty$. The computed compliance for this structure is $\mathcal{C} = 1970\,\mathrm{J}$ ($\mathcal{C}_m = 1768\,\mathrm{J}, \mathcal{C}_b = 202\,\mathrm{J}$). The resulting volume fraction is approximately 0.55, well below the constraint value of 1.

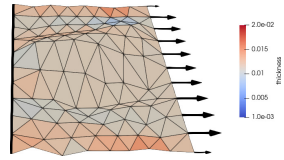


Figure 7: Resulting optimised structure with a constraint on the radius ratio of $\rho_{\mathrm{lim}} = 3$. The computed compliance for this structure is $\mathcal{C} = 511.2\,\mathrm{J}$ ($\mathcal{C}_m = 510.4\,\mathrm{J}, \mathcal{C}_b = 0.79\,\mathrm{J}$).

5.2 The Effect of the Radius Ratio Constraint

We study the effect of the radius ratio constraint on the simple example presented in section 5.1, which used a constraint value of $\rho_{\mathrm{lim}} = 1.1$. The purpose of this study is to gain some insight to the effect of the quality constraint, and whether or not such a constraint is necessary. All other parameters are kept equal.

First, the same example is run without the radius ratio constraint. From the resulting structure, fig. 6, it is clear that the resulting structure is exploiting modelling errors. All nodes of the finite element mesh are relocated to an in-plane central bar, which exploits modelling errors to gain spurious stiffness. Interestingly, this structure is not able to utilise all available material, due to the low surface area of the resulting plate. This leads to a drastically decreased compliance value, when compared with fig. 5, although the resulting compliance value of 13 505 J cannot be interpreted physically, due to the mesh distortion.

Adding a high radius ratio constraint $\rho_{\mathrm{lim}} = 3$ to

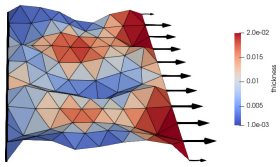


Figure 8: Resulting optimised structure with a constraint on the radius ratio of $\rho_{\text{lim}} = 1$. The computed compliance for this structure is $\mathcal{C} = 2679\,\text{J}$ ($\mathcal{C}_m = 1226\,\text{J}, \mathcal{C}_b = 1453\,\text{J}$).

the same problem gives the result shown in fig. 7. Here we see a plate appearing, but with distorted elements in the center and sides of the plate. Again, the resulting compliance value is hard to compare fairly, as the mesh distortion hinders a physical interpretation of the number. But, the resulting value is slightly lower than the result from fig. 5, indicating that the optimisation algorithm is able to utilise some modelling error to improve the numerically estimated compliance. Another factor, is that the in-plane skew of triangles, might allow for more advantageous distributions of thickness, leading to better compliance values.

Finally, we consider tightening the constraint too far, setting $\rho_{\text{lim}} = 1$, i.e. requiring all triangles to be equilateral. It should be noted that the initial conditions are now infeasible with respect to the radius ratio constraint. Figure 8 shows the resulting structure, which is no longer a flat plate. In an attempt to make all triangles equilateral, the mesh now has a high curvature at almost every edge. The resulting structure is still infeasible with respect to the constraint, as some of the triangles near the fixed boundary cannot become equilateral. As expected, the compliance value is significantly worse than the case from section 5.1, although the physical interpretation is not possible, again, this time due to the high curvature of the resulting mesh.

We can summarise that the radius ratio constraint indeed does have an effect on the resulting structure. It can be used to ensure sufficiently good resulting meshes, effectively enforcing some regularity in the resulting finite element mesh.

5.3 The Effect of Filter Radius

As with the previous section, we investigate the effect of regularisation on the simple example from section 5.1. Here, we investigate the effect of the filter radius r_{supp}.

We consider a zero filter radius $r_{\text{supp}} = 0$, shown

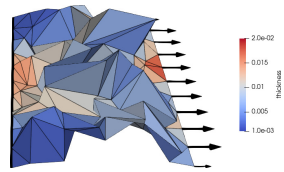


Figure 9: Resulting optimised structure with relocation filter support radius of $r_{\text{supp}} = 0$. The computed compliance for this structure is $\mathcal{C} = 1829\,\text{J}$ ($\mathcal{C}_m = 1824\,\text{J}, \mathcal{C}_b = 5.32\,\text{J}$).

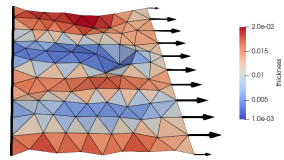


Figure 10: Resulting optimised structure with relocation filter support radius of $r_{\text{supp}} = 0.25$. The computed compliance for this structure is $\mathcal{C} = 579\,\text{J}$ ($\mathcal{C}_m = 511.3\,\text{J}, \mathcal{C}_b = 10.6\,\text{J}$).

in fig. 9. This results in a very distorted surface, and corresponding high compliance value. It is interesting that lowering the filter radius, which should increase the design-freedom, actually leads to worse solutions. Here it is clear that the filter provides some regularity for the shape updates. This is in accordance Ghantasala *et al.* (2021), which also show distorted results obtained with a gradient descent method if no filter is applied.

Increasing the filter radius to $r_{\text{supp}} = 0.25$, which is slightly larger than the average element side-length of 0.2, we see that the structure becomes smoother, as shown in fig. 10. It can be seen that a filter radius with support slightly larger than an element is not sufficient to regularise the problem to result in a flat plate. It can be seen that the compliance value is slightly higher compared to the flat result from fig. 5, as some of the out-of-plane shell is mitigated by moving the plate thickness to the flat areas of the structure.

Increasing the filter radius to $r_{\text{supp}} = 2$, i.e. twice the radius used for the result presented in fig. 5, gives a perfectly flat plate, as shown in fig. 11. This filter radius support corresponds to the length of the clamped and loaded lines. Here the boundary of the flat plate is smoother than the result shown in fig. 5, and the thickness distribution is also slightly more evenly distributed. This increased smoothness is potentially why the larger filter radius also results in a lower compliance value.

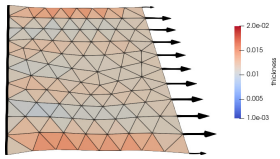


Figure 11: Resulting optimised structure with relocation filter support radius of $r_{\text{supp}} = 2$. The computed compliance for this structure is $\mathcal{C} = 511.8\,\text{J}$ ($\mathcal{C}_m = 510.2\,\text{J}, \mathcal{C}_b = 1.64\,\text{J}$).

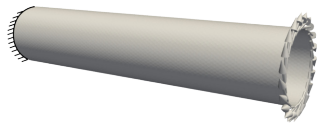


Figure 12: Geometry of the cylinder example. One side of the cylinder is clamped, while the other is subjected to a rotation. The computed compliance for this initial structure is $\mathcal{C} = 1.62\,\text{J}$ ($\mathcal{C}_m = 1.62\,\text{J}, \mathcal{C}_b = 0.003\,\text{J}$).

5.4 Cylinder in Torsion

In order to construct the limitations of applying too much regularisation, we study a cylinder under torsion. This example is based on the well known Michell sphere (Michell, 1904) which is originally composed of trusses. Sigmund *et al.* (2016) show that the optimal structure is actually a spherical shell. By considering a cylinder under torsion, shown in fig. 12, we recreate boundary conditions similar to those of the Michell sphere. We consider a cylinder of length 1m, radius 0.1m, and thickness 0.01m.

The choice of high filter and strict quality measure seem apt based on the experiences from the previous examples. Using these parameters we obtain the structure shown in fig. 13. We observe that the structure is not spherical. Upon further inspection it is found that the radius ratio constraint is the limiting factor, as all elements are pushing the boundary of their allowed ratio.

Considering the limiting factor of the radius ratio constraint, we increase the bound of the constraint to $\rho_{\text{lim}} = 1.5$ and obtain the structure shown in fig. 14. We observe that the structure is still not spherical. Here, the radius of the structure has increased significantly, compared to the previous case, but the structure seems to have developed a slightly cube-like shape. The cube-like features are oriented with the standard basis vectors. Recall that the filter treats the three coordinate components separately, which could indicate that these cube-like features are a result of the high filter radius.

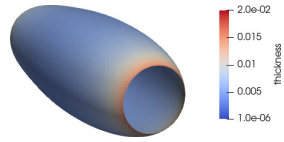


Figure 13: Original solution to the cylinder problem, using the parameters from table 1. The resulting compliance is $\mathcal{C} = 0.74\,\text{J}$ ($\mathcal{C}_m = 0.74\,\text{J}, \mathcal{C}_b = 0.0002\,\text{J}$).

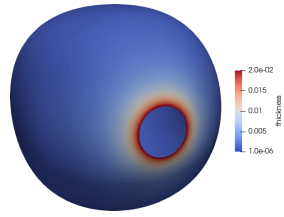


Figure 14: The cylinder example, where the radius ratio constraint is slackened to $\rho_{\text{lim}} = 1.5$. The resulting compliance is $\mathcal{C} = 0.350\,\text{J}$ ($\mathcal{C}_m = 0.350\,\text{J}, \mathcal{C}_b = 4 \times 10^{-5}\,\text{J}$).

We now decrease the filter radius to $r_{\text{supp}} = 0.2$ and keep the increased bound of the constraint. The resulting structure is shown in fig. 15. We observe that the structure is spherical as initially expected.

This small study on the torsional cylinder has shown some of the limitations of the presented formulation for shape optimisation of shell structures. While the radius ratio constraint is important to retain an accurate finite element model, it can also

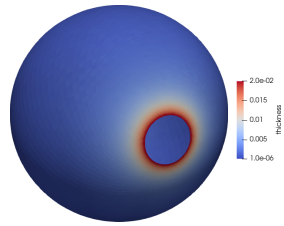
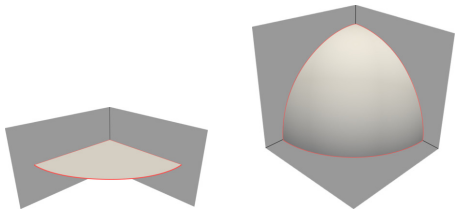


Figure 15: The cylinder example, where the radius ratio constraint is slackened to $\rho_{\text{lim}} = 1.5$, and the filter radius is reduced to $r_{\text{supp}} = 0.2$. The resulting compliance is $\mathcal{C} = 0.347\,\text{J}$ ($\mathcal{C}_m = 0.347\,\text{J}, \mathcal{C}_b = 6 \times 10^{-5}\,\text{J}$).



(a) Circular Plate
$\mathcal{C} = 5.59\,\mathrm{MJ}$
$(\mathcal{C}_m = 0\,\mathrm{J}, \mathcal{C}_b = 5.59\,\mathrm{MJ})$

(b) Sphere
$\mathcal{C} = 34.36\,\mathrm{J}$
$(\mathcal{C}_m = 34.36\,\mathrm{J}, \mathcal{C}_b \approx 0\,\mathrm{J})$

Figure 16: Initial designs for pressure loaded designs. The gray surfaces indicate symmetry condition. The red lines indicate zero-rotation rolling boundary conditions, i.e. symmetry conditions.

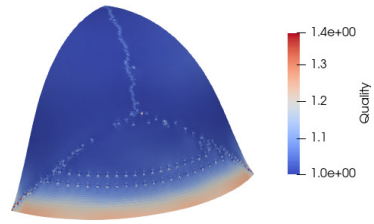


Figure 17: Resulting optimised structure with pressure load and circular plate as initial condition. The mesh is coloured with the element radius ratio element quality measure. The computed compliance for this structure is $\mathcal{C} = 81.92\,\mathrm{J}$ $(\mathcal{C}_m = 51.36\,\mathrm{J}, \mathcal{C}_b = 30.56\,\mathrm{J})$.

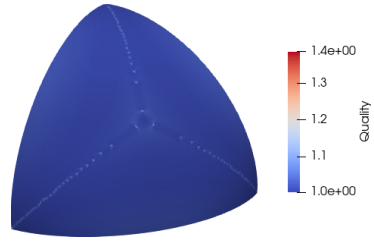


Figure 18: Resulting optimised structure with pressure load and sphere as initial condition. The mesh is coloured with the element radius ratio element quality measure. The computed compliance for this structure is $\mathcal{C} = 6.70\,\mathrm{J}$ $(\mathcal{C}_m = 6.70\,\mathrm{J}, \mathcal{C}_b = 6.6 \times 10^{-6}\,\mathrm{J})$.

limit the relocation of the mesh towards a mesh-agnostic optimal shape. Similarly, the filer on the relocation generally helps convergence to smooth surfaces, but can also limit the realisable surfaces. Our experience is in general that structures where the final shape is curved are more sensitive to high filter radii.

5.5 A Study of Spheres

In order to study the effects of initial geometries, we consider a pressure loaded structure. Two initial structures, shown in fig. 16, are considered with matching boundary conditions. The zero-rotation rolling boundary condition applied to the initial circular plate will potentially act as a symmetry condition. The relocations are constrained to force the structure to remain on the respective symmetry surfaces. The initial sphere and circular plate both have radius 1m, and are discretised with element sizes of 0.05m.

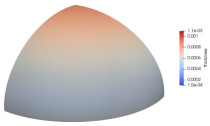
For this case, a pressure loaded unsupported structure, we would expect the resulting structure to be a sphere of uniform thickness, as this carries all loads in the membrane. More specifically, we expect the smallest possible sphere with the highest possible thickness, as lower surface area implies a lower loads. Due to this, we would expect our structure to vanish, as zero area would imply zero force.

We change the direction of the inequality volume constraint in eq. (17e), to enforce a minimally allowed amount of material. We set the minimum volume in absolute terms to $0.0002\,\mathrm{m}^3$, to ensure the same constraint value for the two initial structures.

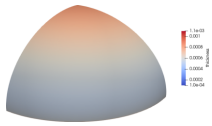
The resulting structures are shown in figs. 17 and 18. It can be seen that the initial spherical structure remains spherical. Although not visible from fig. 18, the radius of the sphere has decreased, and the thickness is the maximally allowed thickness everywhere, as expected.

The structure arising from the flat plate, fig. 17, also has the maximally allowed thickness everywhere, but is not spherical. It should be noted that the structure is still changing slowly after 500 design iterations, unlike other presented results shown here.

Near the original edge of the shell structure, fig. 17, a lip has formed. The elements on this lip can be seen to have a high radius ratio, indicating high skewness. Chances are that these elements can no longer change in the desired direction, as this would violate the radius ratio constraint. These elongated elements will also cause some spurious stiffness due to their shape, although it is not clear to what extent.



(a) Circular Plate.
$\mathcal{C} = 44.61\,\mathrm{J}$
$\mathcal{C}_m = 44.60\,\mathrm{J}$
$\mathcal{C}_b = 0.01\,\mathrm{J}$



(b) Sphere.
$\mathcal{C} = 44.61\,\mathrm{J}$
$\mathcal{C}_m = 44.60\,\mathrm{J}$
$\mathcal{C}_b = 0.01\,\mathrm{J}$.

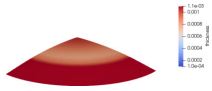
Figure 19: Resulting designs for a pressure loaded plate and sphere, when the outer circular curve is fixed in place.

We observe that the spherical structure has a drastically lower compliance value, compared to the non-spherical result obtained from a flat initial structure. This highlights a fundamental challenge in node-based shape optimisation; The resulting structure needs to be representable by morphing the nodes of the initial mesh. Sometimes, this is not easily achieved, as the finite elements in the morphed mesh are of poor quality. Sometimes, the form updates do not reach a known optimum, due to non-convexity in the underlying optimisation problem.

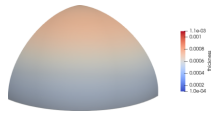
A variation of the pressure loaded problem is to disallow design change in the circular boundary of the plate, and one of the circular cross-sections of the sphere. Since the structure is now ensured from vanishing due to the forced equator, we use the usual volume constraint again, still with an absolute limit of $0.0002\,\mathrm{m}^3$. By forcing the spherical radius in what amounts to an equator for the optimised structures, the expected optimal structure is no longer necessarily a perfect sphere.

We see the resulting structures with fixed equators in fig. 19, where two things are immediately apparent. Both initial structures result in near identical resulting structures, whose main difference seems to be caused by discretisation. Secondly, the resulting structure is not spherical, but ellipsoid. The fixed equator of the ellipsoid is the major radius of the structure, while the radius is reduced in the perpendicular direction to reduce the area, and thus reduce the magnitude of the pressure loading. The spherical initial guess converges faster than a flat initial guess. This is seen from fig. 20, which shows some of the intermediate structures during optimisation.

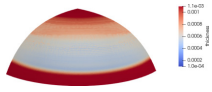
The inclusion of the fixed equator can be interpreted as a reduction of the design space, which apparently changes the problem sufficiently such that both initial structures result in near identical results.



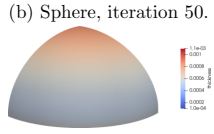
(a) Plate, iteration 50.



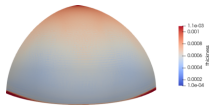
(b) Sphere, iteration 50.



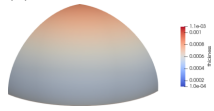
(c) Plate, iteration 150.



(d) Sphere, iteration 150.



(e) Plate, iteration 250.



(f) Sphere, iteration 250.

Figure 20: Structure during selected iterations for circular plate and sphere with fixed equators.

5.6 Cantilever

We present a small study of a cantilever beam, where we consider separate surfaces. The initial geometry is shown in fig. 3, where the green surface is loaded with a downward pointing distributed load. The opposite end of the structure is clamped along the shell edges. All edges and surfaces with loads or boundary conditions are fixed in place by boundary conditions in the shape filter. Parameter details are given in table 1.

As an additional twist on the example, the move limits of the relocation $x_{\mathrm{low}}, x_{\mathrm{upp}}$ are adapted based on the initial position, such that the relocation values will stay within a box of size $2 \times 1 \times 1$, which is the domain of the classic topology optimisation problem. However, we note that the resulting structure might not reside exactly within these bounds, as the bounds are imposed on the unfiltered variables.

The resulting structure and bounding box are shown in fig. 21. We note that the resulting structure is slightly outside of the soft bounding box. The thickness distribution along the central plate resembles the variable thickness sheet results from topology optimisation (Bendsøe and Sigmund, 2004). It can also be seen that the shape of the stiffening flange is changed, to help carry the bending moments.

The intersections between surfaces in the initial geometry are kept during optimisation, due to the nature of filtering on the change from initial

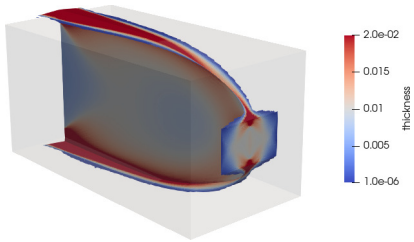


Figure 21: Resulting structure of the cantilever example. Elements with a resulting thickness lower than 1×10^{-5} are removed for clarity. The resulting compliance is $\mathcal{C} = 27.17\,\mathrm{J}$ ($\mathcal{C}_m = 27.02\,\mathrm{J}, \mathcal{C}_b = 0.15\,\mathrm{J}$).

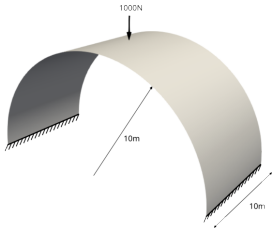


Figure 22: Geometry for half-cylinder example. The two indicated boundary edges are fully clamped. The compliance of the initial structure is $\mathcal{C} = 62.83\,\mathrm{J}$ ($\mathcal{C}_m = 0.27\,\mathrm{J}, \mathcal{C}_b = 62.56\,\mathrm{J}$).

geometry. If only minor changes to an existing structure are desired, this can be considered an advantage. If, however, a fully free form-finding is desired, this is a clear disadvantage.

5.7 Half-cylinder

We study the half-cylinder under a point load presented in Asl and Bletzinger (2022), with initial geometry shown in fig. 22. This is a pure shape optimization problem, where the thickness is kept constant. The two clamped edges are kept in place, while the two remaining edges are kept in their original planes. Three different finite element mesh types are considered, in order to show the influence of meshing in this example, illustrated in fig. 23. The first mesh is unstructured and unsymmetric, denoted original. The second mesh is also unstructured, but now four-fold symmetric. The third and final mesh is generated by splitting a structured mesh of quardilaterals into triangles, it is hence also four-fold symmetric. All three meshes agree on the initial compliance values stated in the caption of fig. 22. The optimisation parameters are kept identical for the three examples, unless stated otherwise, and can be found in table 1.

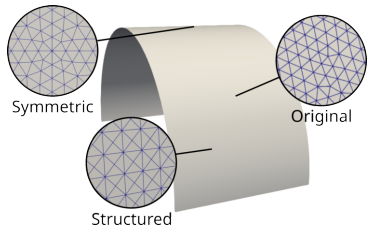


Figure 23: Illustration of the three studied mesh types for the half-cylinder example.

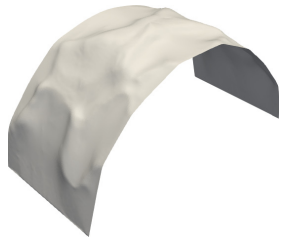


Figure 24: Resulting structure of half-cylinder example with the original unsymmetric unstructured mesh. The resulting compliance is $\mathcal{C} = 0.096\,\mathrm{J}$ ($\mathcal{C}_m = 0.088\,\mathrm{J}, \mathcal{C}_b = 0.008\,\mathrm{J}$).

The resulting structure of the non-symmetric original mesh is shown in fig. 24. The upper part of the structure has a cross shaped stiffener, which distributes the point force. Stiffeners are also added further down the curvature of the structure, although they are not symmetric. As the load is inherently symmetric, we would expect symmetry in this resulting structures. This result might be due to the unsymmetric node placement, which means that the symmetry of the problem is not reflected in the used design variables.

Changing the underlying mesh to be symmetric, but still unstructured, we obtain the results presented in fig. 25. Here we see that the resulting structure is symmetric as expected. The resulting compliance value is also lower, meaning that the symmetric structure performs better than the non-symmetric counterpart.

Finally, we consider a fully structured mesh which is also symmetric. We note that the initial value of the radius ratio constraint for this mesh is $\rho_{\mathrm{agg}} \approx 1.2$. We therefore choose to increase the allowed element radius ratio from 1.1 to

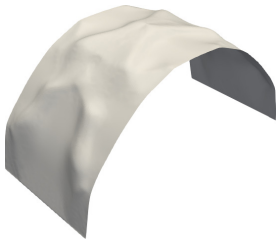


Figure 25: Resulting structure of half-cylinder example with a symmetric unstructured mesh. Here the used radius ratio constraint has been increased to $\rho_{\text{lim}} = 1.3$. The resulting compliance is $\mathcal{C} = 0.086\,\text{J}$ ($\mathcal{C}_m = 0.077\,\text{J}, \mathcal{C}_b = 0.009\,\text{J}$).

Figure 26: Resulting structure of half-cylinder example with a symmetric structured mesh. The resulting compliance is $\mathcal{C} = 0.090\,\text{J}$ ($\mathcal{C}_m = 0.082\,\text{J}, \mathcal{C}_b = 0.008\,\text{J}$).

1.3, to avoid a situation similar to that presented in fig. 8. The resulting structure is shown in fig. 26, where we see a near symmetric resulting structure. While not clear from any visualisation, the top cross stiffener is slightly skewed to one side, rendering the structure non-symmetric. The resulting compliance is also slightly higher than for the fully symmetric result. We speculate that this might be due to a slight non-symmetry in the initial mesh.

We have seen the importance of the underlying mesh, once again, in this example. Symmetry, or rather node distribution, plays an important role for the resulting structures. However, we also see that the core features are similar for all three results. All achieve a cross shaped stiffener on the top, with some stiffener around the curvature. While not seen in fig. 24, the back side of the non-symmetric structure is similar to the two symmetric results, indicating slightly better agreement than shown.

If truly symmetric structures are desired, the mesh should be modelled with symmetry directly, as done in the example from section 5.5, which would also reduce the computational cost of the model. This was not chosen for this example, as we wish to study the effect of non-symmetry of the mesh to the resulting structures.

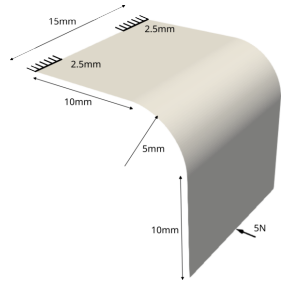


Figure 27: Geometry for cowling example. The two black edges of 2.5m, indicated with thick lines, indicate fully clamped edges. The compliance of the initial structure is $\mathcal{C} = 3241\,\text{J}$ ($\mathcal{C}_m = 3.98\,\text{J}, \mathcal{C}_b = 3237\,\text{J}$).

5.8 Cowling

We study the cowling example with a point load, as shown in fig. 27. This example is originally presented in Arnout *et al.* (2012), where it is designed for high bending in the original geometry. We have tried to reconstruct the example as close as possible. One possible source of difference is our interpretation of the geometric constraints. We constrain all four straight edges of the cowling to remain in their respective planes of the initial shell. A maximal relocation, or bead depth, is imposed by box constraints. Similarly, the shell is not allowed to grow larger than the initial height and depth, imposed again by box constraints. The remaining parameters are given in table 1.

The thickness-to-radius ratio of the initial shell structure from Arnout *et al.* (2012) is 10. Such a thick shell is a poor match for the used shell element, specially considering that shape optimisation tends to generate high curvature areas, which would further decrease the thickness-to-radius ratio. We therefore consider a variation of the example, by considering a thinner shell, with base thickness of 0.05mm, resulting in an initial thickness-to-radius ratio of 100.

The result are shown in fig. 28 for pure shape optimisation. Here curved part of the shell is stiffened by flattening, and inserting bent stiffeners. The shell is bent slightly inward in both top and bottom. It can be seen that the compliance is

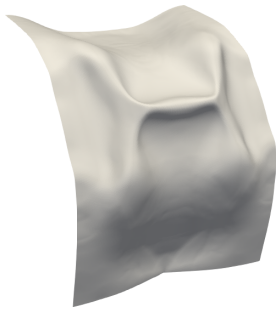


Figure 28: Result of shape optimization of cowling example. The resulting compliance is $\mathcal{C} = 4.09\,\mathrm{J}$ ($\mathcal{C}_m = 3.89\,\mathrm{J}, \mathcal{C}_b = 0.20\,\mathrm{J}$).

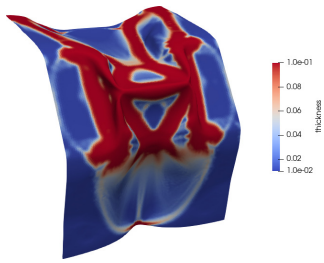


Figure 29: Result of shape and thickness optimization of cowling example. The resulting compliance is $\mathcal{C} = 3.67\,\mathrm{J}$ ($\mathcal{C}_m = 3.15\,\mathrm{J}, \mathcal{C}_b = 0.51\,\mathrm{J}$).

reduced drastically from the initial configuration, and that the compliance of the resulting structure is mainly due to membrane strains.

Figure 29 shows the result of combined optimisation of thickness and shape. Here, similar feature the pure shape optimisation can be seen, although the bent stiffeners have smaller radius and larger thickness. The thickness is distributed primarily in the top of the structure, leaving only a slight stiffening near the point load. The compliance is lower when optimising both shape and thickness, compared to only optimising shape, although the difference is not very large when comparing to the

Both optimised cowling examples show very high curvature in some elongated stiffeners. The stiffeners are estimated to have radii of 0.225 m in the most curved areas, which correspond to thickness-to-radius ratios of 4.5 and 2.25 for the shape-only and shape-and-thickness case respectively. These ratios both clearly break with the thin-walled assumption of the shell elements, brining into question the validity of the solved finite element model. While the placement of stiffeners is quite intuitive from a mechanical perspective, correctness of the underlying finite element mode is important to guarantee validity optimised results. Hence, a sufficiently large filter is sometimes needed, to disallow problematic high-curvature sections, if shell elements are to be used for modelling the mechanical response.

6 Discussion

In this work we have presented a consistent formulation for optimisation of shape and thickness of shell structures. Almost all of the components used in this formulation are not novel, but the key contribution is composition.

We are able to define a mathematically consistent optimisation problem by always referencing back to the initial mesh coordinates. Meaning that the optimisation problem shown in eq. (17) does not change during the design history, as the filters and updates do not depend on the design variables. This is not only advantageous in some strict mathematical sense. Optimisation algorithms which incorporate the history, such as quasi-Newton methods, or the used Method of Moving Asymptotes.

The main drawback of the relocation based formulation is that some shapes are not realisable with the regularised parameters. Also, some shapes are not realisable, as the resulting aggregated radius ratio would be too high. This method should primarily be considered for small changes, or corrections, to shell structures which are already close to the desired final design.

Lastly, the presented constraint on the aggregated radius ratio has shown to be a very useful tool for ensuring a sufficiently accurate finite element analysis for the resulting shapes.

Reproducibility

How to access code goes here.

Acknowledgements

The authors would like to thank Dirk P. Munro for his generously given confusion.

The authors would also like to acknowledge the Villum foundation, without whom the Danes would be destitute.

Appendices

A Shell Formulation

This is a summary of the linear shell element. For the derivation, and finite deformations, please refer to Morley (1971); van Keulen (1993); Van Keulen and Booij (1996).

A.1 Local system

We introduce a local coordinate system for each element. The coordinate system is defined by the orthonormal basis vectors $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \bar{\mathbf{e}}_3$, which are constructed from the nodal coordinates $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

The coordinate system is constructed such that $\bar{\mathbf{e}}_3$ denotes the out-of-plane direction. This is achieved by defining the basis as the unit normal for the plane of the triangle.

$$\bar{\mathbf{e}}_3 = \frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{||(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)||_2} \tag{22}$$

The first two basis vectors are chosen, somewhat arbitrarily, as

$$\bar{\mathbf{e}}_1 = \frac{\mathbf{x}_2 - \mathbf{x}_1}{||\mathbf{x}_2 - \mathbf{x}_1||_2}, \quad \bar{\mathbf{e}}_2 = \bar{\mathbf{e}}_3 \times \bar{\mathbf{e}}_1 \tag{23}$$

Which ensures an orthonormal basis by construction. .

Using the basis-vectors, the local rotation matrix for a point $\mathbf{T}_{\text{point}}$ can be found by

$$\mathbf{T}_{\text{point}} = \begin{bmatrix} \bar{\mathbf{e}}_1 & \bar{\mathbf{e}}_2 & \bar{\mathbf{e}}_3 \end{bmatrix} \tag{24}$$

This matrix can be used to rotate the nodal displacements from the global reference u_i, v_i, w_i to the local reference $\bar{u}_i, \bar{v}_i, \bar{w}_i$.

$$\begin{bmatrix} \bar{u}_i \\ \bar{v}_i \\ \bar{w}_i \end{bmatrix} = \mathbf{T}_{\text{point}} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}, \quad i \in \{1, 2, 3\} \tag{25}$$

The transformation of all element degrees-of-freedom can now be defined by the element matrix $\mathbf{T}$. The matrix is constructed from entries of $\mathbf{T}_{\text{point}}$ to couple the displacements locally in each node, and the identity matrix to transfer the edge rotations $\phi_{12}, \phi_{23}, \phi_{31}$ without change.

$$\begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \\ \bar{w}_1 \\ \bar{w}_2 \\ \bar{w}_3 \\ \phi_{12} \\ \phi_{23} \\ \phi_{31} \end{bmatrix} = \mathbf{T} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ v_1 \\ v_2 \\ v_3 \\ w_1 \\ w_2 \\ w_3 \\ \phi_{12} \\ \phi_{23} \\ \phi_{31} \end{bmatrix} \tag{26}$$

A.2 Membrane formulation

The membrane aspect is modelled using a standard constant strain triangle (CST) formulation. The element is defined for the in-plane deformations in the plane spanned by $\bar{\mathbf{e}}_1$ and $\bar{\mathbf{e}}_2$.

The considered triangle can be parametrised by the coefficients a_i and b_i, where the sides of the triangle are specified by

$$\mathbf{s}_i = a_i \bar{\mathbf{e}}_1 - b_i \bar{\mathbf{e}}_2, \quad i \in \{1, 2, 3\} \tag{27}$$

As the triangle sides vectors $\mathbf{s}_i$ can also be found from the nodal coordinates, as shown in fig. 30, this gives a way to compute the triangle coefficients.

$$a_i = \mathbf{s}_i^\top \bar{\mathbf{e}}_1, \quad b_i = -\mathbf{s}_i^\top \bar{\mathbf{e}}_2, \quad i \in \{1, 2, 3\} \tag{28}$$

These coordinate coefficients can be used to redefine the triangular problem in a basis defined by the edge normals.

The area shape functions are introduced

$$\xi_i = \frac{A_i}{A_e}, \quad i \in \{1, 2, 3\} \tag{29}$$

Where A_i denote the areas of the subtriangles shown in fig. 30, and A_e denotes the area of the entire triangle. Taking the spatial derivatives of the area coordinates result in the following

$$\begin{aligned} \mathbf{d}_1^\top &= \frac{1}{2A_e} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}, \\ \mathbf{d}_2^\top &= \frac{1}{2A_e} \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \end{aligned} \tag{30}$$

Which can be used to define the membrane strain-interpolation matrix $\mathbf{B}_{\text{m}}$

$$\epsilon_{\text{m}} = \mathbf{B}_{\text{m}} \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{bmatrix}, \quad \mathbf{B}_{\text{m}} = \begin{bmatrix} \mathbf{d}_1^\top & 0 \\ 0 & \mathbf{d}_2^\top \\ \mathbf{d}_2^\top & \mathbf{d}_1^\top \end{bmatrix} \tag{31}$$

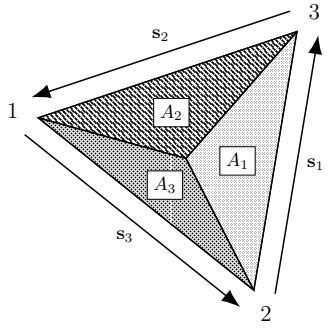


Figure 30: Area coordinates and element side vectors.

Finally, we note that the linear constitutive law for membranes is given by

$$\mathbf{C}_\mathrm{m} = \frac{Eh}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{32}$$

A.3 Bending formulation

The bending is modelled using a constant bending-plate element and derived in Morley (1971); Van Keulen and Booij (1996). We summarise the resulting formulation, but do not provide the full derivation for brevity.

The deformation is modelled using out-of-plane displacements $\bar{w}_1, \bar{w}_2, \bar{w}_3$ at the nodes along with rotations about the element edges $\phi_{12}, \phi_{23}, \phi_{31}$. These variables are used to define relative rotations, which in turn are used to compute the change in curvatures κ_b. The relation is given by the matrix $\mathbf{B}_\mathrm{b}$

$$\kappa_\mathrm{b} = \mathbf{B}_\mathrm{b}\begin{bmatrix} \bar{w}_1 \\ \bar{w}_2 \\ \bar{w}_3 \\ \phi_{12} \\ \phi_{23} \\ \phi_{31} \end{bmatrix} \tag{33}$$

Where it should be noted that the edge rotations are defined about s_1, s_2, s_3. In order to ensure correctness during the assembly of local matrices, the local definitions must be changed, such that triangles sharing an edge define the rotation in the same direction. The direction is changed by changing the sign of the appropriate column of $\mathbf{B}_\mathrm{b}$, and can be done based on the node numbers associated with the edge.

The constitutive law for bending is given by

$$\mathbf{C}_\mathrm{b} = \frac{Eh^3}{12(1-\nu^2)}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{34}$$

The side-lengths of the triangle can be found by

$$\begin{aligned} s_{12}^2 &= b_3^2 + a_3^2, \\ s_{23}^2 &= b_1^2 + a_1^2, \\ s_{31}^2 &= b_2^2 + a_2^2 \end{aligned} \tag{35}$$

The sidelengths are used in the final expression for the curvature interpolation $\mathbf{B}_\mathrm{b}$ along with the triangle coefficients from the previous section.

$$\mathbf{B}_\mathrm{b}^\top = \begin{bmatrix} \left(\frac{b_2 a_2}{s_{31}^2} - \frac{b_3 a_3}{s_{12}^2}\right) & \left(\frac{b_3 a_3}{s_{12}^2} - \frac{b_2 a_2}{s_{31}^2}\right) & 2\left(\frac{b_2^2}{s_{31}^2} - \frac{b_3^2}{s_{12}^2}\right) \\ \left(\frac{b_3 a_3}{s_{12}^2} - \frac{b_1 a_1}{s_{23}^2}\right) & \left(\frac{b_1 a_1}{s_{23}^2} - \frac{b_3 a_3}{s_{12}^2}\right) & 2\left(\frac{b_3^2}{s_{12}^2} - \frac{b_1^2}{s_{23}^2}\right) \\ \left(\frac{b_1 a_1}{s_{23}^2} - \frac{b_2 a_2}{s_{31}^2}\right) & \left(\frac{b_2 a_2}{s_{31}^2} - \frac{b_1 a_1}{s_{23}^2}\right) & 2\left(\frac{b_1^2}{s_{23}^2} - \frac{b_2^2}{s_{31}^2}\right) \\ -\frac{b_3^2}{s_{12}^2} & -\frac{a_3^2}{s_{12}^2} & \frac{2b_3 a_3}{s_{12}^2} \\ -\frac{b_1^2}{s_{23}^2} & -\frac{a_1^2}{s_{23}^2} & \frac{2b_1 a_1}{s_{23}^2} \\ -\frac{b_2^2}{s_{31}^2} & -\frac{a_2^2}{s_{31}^2} & \frac{2b_2 a_2}{s_{31}^2} \end{bmatrix} \tag{36}$$

A.4 Element formulation

The resulting finite element formulation applies the membrane and bending systems separately for the deformations in the local coordinate system. The resulting stiffness matrix can be found as

$$\mathbf{k}_e = \int_{V_e} \mathbf{T}^\top \begin{bmatrix} \mathbf{B}_\mathrm{m}^\top \mathbf{C}_\mathrm{m} \mathbf{B}_\mathrm{m} & 0 \\ 0 & \mathbf{B}_\mathrm{b}^\top \mathbf{C}_\mathrm{b} \mathbf{B}_\mathrm{b} \end{bmatrix} \mathbf{T} dV_e \tag{37}$$

As expression is constant in the element, the integration can be simplified to

$$\mathbf{k}_e = A_e \mathbf{T}^\top \begin{bmatrix} \mathbf{B}_\mathrm{m}^\top \mathbf{C}_\mathrm{m} \mathbf{B}_\mathrm{m} & 0 \\ 0 & \mathbf{B}_\mathrm{b}^\top \mathbf{C}_\mathrm{b} \mathbf{B}_\mathrm{b} \end{bmatrix} \mathbf{T} \tag{38}$$

Where A_e denotes the area of the element.

B Filter Matrices

The finite element formulation of the filter for linear triangular elements is presented here. The area coordinates from appendix A are used as the shape functions.

$$\mathbf{N}_\mathrm{f} = \begin{bmatrix} \xi_1 & \xi_2 & \xi_3 \end{bmatrix} \tag{39}$$

The derivatives of the shape functions can be found by using eq. (30).

$$\mathbf{B}_\mathrm{f} = \frac{1}{2A_e}\begin{bmatrix} b_1 & b_2 & b_3 \\ a_1 & a_2 & a_3 \end{bmatrix} \tag{40}$$

Using the shape-functions the mass matrix of this scalar problem discretised by a linear triangle is

$$\mathbf{M}_e = \int_{V_e} \mathbf{N}_\mathrm{f}^\top \mathbf{N}_\mathrm{f} dV_e = \frac{A_e}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{41}$$

And the discretised Laplace operator for the triangle is

$$\mathbf{L}_e = \int_{V_e} \mathbf{B}_\mathrm{f}^\top \mathbf{B}_\mathrm{f} dV_e = A_e \mathbf{B}_\mathrm{f}^\top \mathbf{B}_\mathrm{f} \tag{42}$$

Note that both matrices are only constructed for the initial mesh configuration, i.e. only $\mathbf{x}_0$ is used to compute the shape function derivatives and areas.

C Example parameters

See table 1 for all parameters.

D Compliance of membrane

We start with a cylindrical shell modelling a quarter of a cylinder with radius $r = 1\,\mathrm{m}$ and thickness $h_o = 0.01\,\mathrm{m}$. This is then transformed to a flat plate of identical volume with sidelengths $d = 2\,\mathrm{m}$ and $L = \sqrt{2}\mathrm{m}$. First, we wish to find the thickness of the resulting flat plate, denoted h. We note that from the volume conservation we have that

$$v = \frac{1}{4} 2\pi r d = h d L \tag{43}$$

From this we find the resulting thickness of the plate as.

$$h = \frac{\frac{1}{4}2\pi r d}{dL} = \frac{\pi}{dL} h_o \approx 0.011\,10\,\mathrm{m} \tag{44}$$

The resulting force is found by integrating the line load of $\sqrt{2}\mathrm{N/m}$.

$$f = \int_l \sqrt{2}\mathrm{N/m} dl = 2\sqrt{2}\mathrm{N} \approx 2.83\,\mathrm{N} \tag{45}$$

Recalling the Young's modulus $E = 1\,\mathrm{Pa}$, we can now find an equivalent spring stiffness, under linear elastic assumptions.

$$k = E\frac{A}{L} = E\frac{dh}{L} \tag{46}$$

Using the spring stiffness we find the resulting displacement u.

$$u = k^{-1} f = \frac{2}{h}\mathrm{m} \approx 180.18\,\mathrm{m} \tag{47}$$

we realise that this is well beyond linear elastic assumptions, to say the least, but we continue the example nevertheless.

Finally, we find the resulting compliance. We recall that since the plate is flat, and we pull only in the membrane direction we have that $\mathcal{C}_b = 0$. The resulting compliance is found as:

$$\mathcal{C} = \mathcal{C}_m = uf \approx 509.6\,\mathrm{J} \tag{48}$$

References

I. Antonau, S. Warnakulasuriya, K.-U. Bletzinger, F. M. Bluhm, M. Hojjat, and R. Wüchner. Latest developments in node-based shape optimization using vertex morphing parameterization. *Structural and Multidisciplinary Optimization*, 65(7):198, Jul 2022. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-022-03279-w.

S. Arnout, M. Firl, and K.-U. Bletzinger. Parameter free shape and thickness optimisation considering stress response. *Structural and Multidisciplinary Optimization*, 45(6):801–814, Jun 2012. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-011-0742-8.

R. N. Asl and K.-U. Bletzinger. *Implicit bulk-surface filtering method for node-based shape optimization and comparison of explicit and implicit filtering techniques*. Number arXiv:2208.00700. Aug 2022. URL `http://arxiv.org/abs/2208.00700`. arXiv:2208.00700 [cs, math].

M. P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Engineering online library. Springer, Berlin Heidelberg, 2. ed., corrected printing edition, 2004. ISBN 978-3-662-05086-6.

K.-U. Bletzinger. A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Structural and Multidisciplinary Optimization*, 49(6):873–895, Jun 2014. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-013-1031-5.

M. E. Botkin. Shape optimization of plate and shell structures. *AIAA Journal*, 20(2): 268–273, Feb 1982. ISSN 0001-1452, 1533-385X. doi:10.2514/3.51074.

Example	Sanity Check Figure 4	Pressure plate Figure 16a	Pressure sphere Figure 16b	Cylinder Figure 12	Cantilever Figure 3	Half-cylinder Figure 22	Cowling Figure 27
Initial thickness	0.01	0.0005	0.001	0.01	0.01	0.01	0.05
Thk. filter radius $r_{h\ \mathrm{supp}}$	0.2	0	0	0.005	0	-	0.2
Thk. bounds $h_{\mathrm{upp}}, h_{\mathrm{low}}$	0.02, 0.001	0.0011,0.0001	0.0011,0.0001	0.02, 10^{-6}	0.02, 10^{-6}	-	0.1, 0.001
Shape filter radius r_{supp}	0.5	2	2	0.5	0.5	2	5
Shape bounds $x_{\mathrm{upp}}, x_{\mathrm{low}}$	± 1	± 10	± 10	± 1	± 1	± 0.4	± 2.5
Average element size	0.2	0.05	0.01	0.01	0.01	-	0.1mm
Radius Ratio ρ_{lim}	1.1	1.1	1.1	1.1	1.1	1.1	1.1
Volume fraction V^*	1	-	-	1	1	1	1.01
Young's Module E	1	1	1	1	1	206.9×10^9	210×10^9
Poissons Ratio ν	0.3	0.3	0.3	0.3	0.3	0.29	0.3
Used iterations	500	500	150	300	300	500	400

Table 1: Used parameters for all examples.

H. Eschenauer and M. Weinert. *Structural Optimization Techniques as a Mathematical Tool for finding Optimal Shapes of Complex Shell Structures*, pages 135–153. Gordon and Breach Science Publishers, 1992. ISBN 978-2-88124-878-8.

A. Gates and M. Accorsi. Automatic shape optimization of three-dimensional shell structures with large shape changes. *Computers & Structures*, 49:167–178, Oct 1993. ISSN 00457949. doi:10.1016/0045-7949(93)90135-Z.

A. Ghantasala, R. Najian Asl, A. Geiser, A. Brodie, E. Papoutsis, and K.-U. Bletzinger. Realization of a framework for simulation-based large-scale shape optimization using vertex morphing. *Journal of Optimization Theory and Applications*, 189(1):164–189, Apr 2021. ISSN 0022-3239, 1573-2878. doi:10.1007/s10957-021-01826-x.

M. Hojjat, E. Stavropoulou, and K.-U. Bletzinger. The vertex morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268:494–513, Jan 2014. ISSN 00457825. doi:10.1016/j.cma.2013.10.015.

B. S. Lazarov and O. Sigmund. Filters in topology optimization based on helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering*, 86(6):765–781, May 2011. ISSN 00295981. doi:10.1002/nme.3072.

B. S. Lazarov, F. Wang, and O. Sigmund. Length scale and manufacturability in density-based topology optimization. *Archive of Applied Mechanics*, 86(1–2):189–218, Jan 2016. ISSN 0939-1533, 1432-0681. doi:10.1007/s00419-015-1106-4.

A. M. M. Leal. autodiff, a modern, fast and expressive C++ library for automatic differentiation. `https://autodiff.github.io`, 2018. URL `https://autodiff.github.io`.

K. Maute and E. Ramm. Adaptive topology optimization of shell structures. *AIAA Journal*, 35(11):1767–1773, Nov 1997. ISSN 0001-1452, 1533-385X. doi:10.2514/2.25.

A. Michell. Lviii. the limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 8(47):589–597, Nov 1904. ISSN 1941-5982, 1941-5990. doi:10.1080/14786440409463229.

G. Mohr. Design of shell shape using finite elements. *Computers and Structures*, 10(5):745 – 749, 1979. doi:10.1016/0045-7949(79)90038-5.

L. S. D. Morley. The constant-moment plate-bending element. *Journal of Strain Analysis*, 6(1):20–24, Jan 1971. ISSN 0022-4758. doi:10.1243/03093247V061020.

P. P. Pébay and T. J. Baker. Analysis of triangle quality measures. *Mathematics of Computation*, 72(244):1817–1840, Jan 2003. ISSN 0025-5718. doi:10.1090/S0025-5718-03-01485-6.

C. Ramakrishnan and A. Francavilla. Structural shape optimization using penalty functions. *Journal of Structural Mechanics*, 3 (4):403–422, Jan 1974. ISSN 0360-1218. doi:10.1080/03601217408907275.

E. Ramm, K.-U. Bletzinger, and R. Reitinger. Shape optimization of shell structures. *Revue Européenne des Éléments Finis*, 2(3):377–398, Jan 1993. ISSN 1250-6559. doi:10.1080/12506559.1993.10511083.

O. Sigmund, N. Aage, and E. Andreassen. On the (non-)optimality of michell structures. *Structural and Multidisciplinary Optimization*, 54(2): 361–373, Aug 2016. ISSN 1615-147X, 1615-1488. doi:10.1007/s00158-016-1420-7.

K. Svanberg. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, Feb 1987. ISSN 0029-5981, 1097-0207. doi:10.1002/nme.1620240207.

F. van Keulen. *On Refined Trinagular Plate and Shell Elements*. PhD thesis, Technische Universiteit Delft, 1993.

F. Van Keulen and J. Booij. Refined consistent formulation of a curved triangular finite rotation shell element. *International Journal for Numerical Methods in Engineering*, 39(16):2803–2820, Aug 1996. ISSN 0029-5981, 1097-0207. doi:10.1002/(SICI)1097-0207(19960830)39:16<2803::AID-NME977>3.0.CO;2-2.

O. Zienkiewicz, R. Taylor, and J. Zhu. *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*. Butterworth-Heinemann, Oxford, seventh edition edition, 2013. ISBN 978-1-85617-633-0.

DTU Construct
Section of Solid Mechanics
Technical University of Denmark

Koppels Allé, Bld. 404
DK-2800 Kgs. Lyngby
Denmark
Tlf.: +45 4525 4250
Fax: +45 4525 1961

www.construct.dtu.dk

April 2023

ISBN: 978-87-7475-724-5

DCAMM
Danish Center for Applied Mathematics and Mechanics

Koppels Allé, Bld. 404
DK-2800 Kgs. Lyngby
Denmark
Phone (+45) 4525 4250
Fax (+45) 4525 1961

www.dcamm.dk

DCAMM Special Report No. S332

ISSN: 0903-1685