Pontifícia Universidade Católica
do Rio de Janeiro

Fernando Vasconcelos da Senhora

Locally stress-constrained topology optimization with continuously varying loading direction and amplitude: Toward large-scale problems

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Engenharia Mecânica, do Departamento de Engenharia Mecânica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Mecânica.

Advisor: Prof. Ivan Fábio Mota de Menezes

Rio de Janeiro
May 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Fernando Vasconcelos da Senhora

Locally stress-constrained topology optimization with continuously varying loading direction and amplitude: Toward large-scale problems

Thesis presented to the Programa de Pós–graduação em Engenharia Mecânica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Mecânica. Approved by the Examination Committee:

Prof. Ivan Fábio Mota de Menezes
Advisor
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Emílio Carlos Nelli Silva
Universidade de São Paulo - USP

Prof. Americo Barbosa da Cunha Junior
Universidade Estadual do Rio de Janeiro - UERJ

Prof. Glaucio Hermogenes Paulino
Georgia Institute of Technology - GATECH

Prof. Anderson Pereira
Departamento de Engenharia Mecânica – PUC-Rio

Rio de Janeiro, May the 13th, 2022

Fernando Vasconcelos da Senhora

Bachelor's in Mechanical Engineering with a minor in Mathematics from the Pontifical Catholic University of Rio de Janeiro. Completed a Master in Mechanical Engineering in the Pontifical Catholic University of Rio de Janeiro.

Bibliographic data

Vasconcelos Senhora, Fernando

Locally stress-constrained topology optimization with continuously varying loading direction and amplitude: Toward large-scale problems / Fernando Vasconcelos da Senhora; advisor: Ivan Fábio Mota de Menezes. – 2022.

131 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2022.

Inclui bibliografia

1. Engenharia Mecanica – Teses. 2. Otimização Topológica. 3. Restrições de Tensão. 4. Carregamento com Direção Variável. 5. Carregamento com Amplitude Variável. 6. Problemas de Grande Escala. I. Mota de Menezes, Ivan Fabio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. III. Título.

CDD: 621

To all the ones we have lost.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The author also acknowledge the financial support provided by the Brazilian agencies Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ), Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), and Tecgraf/PUC-Rio (Group of Technology in Computer Graphics), Rio de Janeiro, Brazil.

I would like to thank my advisor Prof. Ivan Fábio Mota de Menezes for whom I have nothing but the most profound admiration. Thank for all your support, and for being a wonderful role model throughout these years. More than an advisor, I consider you a dear friend that I hope to keep for the rest of my life. I also want to thank Prof. Glaucio Paulino who also had a central role in advising me through my PhD, your passion and dedication has truly inspired me to become a better version of myself. Thank you for believing in me, and for providing essential guidance in this journey. I would like to thank the members of my committee Prof. Anderson Pereira, Prof. Americo Cunha, and Prof. Emilio Silva for taking the time to read my work, and for their insightful comments, suggestions, and discussions, that helped improve the quality of my work.

Outside of the academic field, but not less important, I want to thank my parents Fernando Rocha da Senhora, and Denise Bezerra Vasconcelos da Senhora, my brother Leonardo Vasconcelos da Senhora, and all my family (which is too extensive to name here). Your love and support is what made this possible. I also want to thank all my friends for brightening up my days, and making life more colorful and cheerful. I hope you all know how much you mean to me.

Abstract

Vasconcelos Senhora, Fernando; Mota de Menezes, Ivan Fabio (Advisor). **Locally stress-constrained topology optimization with continuously varying loading direction and amplitude: Toward large-scale problems**. Rio de Janeiro, 2022. 131p. Tese de Doutorado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

In the field of structural optimization, Topology Optimization (TO) is one of the most general techniques because it is able to generate complex structures with intricate details for a wide range of problems. However, most of the works in TO have focused on compliance-based design that does not consider material strength in the design process leading to structures that do not satisfy material failure requirements. In this work, we focus on the stress-based design approach. We introduce stress constraints in the optimization procedure to guarantee the structural integrity of the final optimized design. This leads to a more natural formulation that addresses a simple engineering question: What is the lightest structure able to withstand its loads? We developed a large-scale GPU-based parallel stress-constrained TO framework considering a continuous range of varying load directions to answer this question and close the gap between TO and practical application. The developed GPU-based C++/CUDA framework efficiently addresses the main challenges of large-scale TO, filtering, optimization algorithm, and the solution of the equilibrium equations, only requiring a moderately affordable GPU hardware. At the same time, we obtain designs that are more suitable for engineering applications by considering a continuous variable range of load directions that more closely resemble real-life service loads using a worst-case analytical approach. We present several numerical results, including 3D problems with over 45 million local constraints providing detailed optimal structures that demonstrate the capabilities of the techniques developed in this work. The large-scale GPU framework, combined with the analytical solutions for continuously varying load cases, has the potential to expand the applications of TO techniques leading to improved engineering designs.

Keywords

Topology Optimization; Stress Constraints; Varying Loading Direction; Varying Loading Amplitude; Large-Scale Problem.

Resumo

Vasconcelos Senhora, Fernando; Mota de Menezes, Ivan Fabio. **Otimização topológica com restrições locais de tensão e variação contínua da direção e amplitude do carregamento: aplicações em problemas de grande escala**. Rio de Janeiro, 2022. 131p. Tese de Doutorado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Otimização topológica (OT) é uma técnica de otimização estrutural capaz de gerar projetos incrivelmente detalhados para uma grande gama de problemas. No entanto, a maioria dos trabalhos de OT presentes na literatura está focada em problemas de minimização de flexibilidade, que não consideram a resistência dos materiais durante o processo de otimização, levando a soluções que não satisfazem limites de falha do material. Neste trabalho, focamos em problemas de OT baseados em tensão no qual introduzimos restrições de tensão no problema de otimização, para garantir a integridade estrutural do projeto final. A formulação de tensão de OT nos leva a um problema de engenharia muito mais natural que nos remete à seguinte pergunta: Qual a estrutura mais leve capaz de suportar as cargas as quais será submetida? Para ajudar a responder essa pergunta e para trazer a OT para mais próximo de aplicações reais, neste trabalho foi desenvolvido um sistema computacional em paralelo, baseado em GPU, considerando uma carga que pode variar a sua direção continuamente e capaz de resolver problemas de larga escala. A implementação em GPU apresenta soluções eficientes para os principais problemas de OT de larga escala, como o filtro, o algoritmo de otimização e a solução das equações de equilíbrio. Ao mesmo tempo, ao considerar uma carga variando continuamente que mais se aproxima das condições reais de carregamento usando uma estratégia de pior cenário, obtém-se soluções mais robustas e mais adequadas a aplicações de engenharia. Várias soluções numéricas são apresentadas, incluindo problemas 3D com mais de 45 milhões de restrições de tensão, que demonstram a efetividade das técnicas desenvolvidas neste trabalho. O sistema de larga escala baseado em GPU combinado com as soluções analíticas para a variação contínua de carga, tem o potencial de expandir o uso da OT na engenharia levando a novas e mais eficientes estruturas.

Palavras-chave

Otimização Topológica; Restrições de Tensão; Carregamento com Direção Variável; Carregamento com Amplitude Variável; Problemas de Grande Escala.

Table of contents

List of figures

List of tables

List of Abreviations

AGD Adaptive Gradient Descent-based

AL Augmented Lagrangian

B&B Branch-and-Bound method

CG Conjugated Gradient

DOF Degrees of Freedom

EbE Element-by-Element approach

FEA Finite Element Analysis

FEM Finite Element Method

FLOPS Floating-Point Operations

GPU Graphic Processing Unit

MMA Method of Moving Asymptotes

MPVCs Mathematical Program with Vanishing Constraints

MPVSC Modified Piecewise Vanishing Stress Constraint

NP Non-Deterministic Polynomial Time

OC Optimality Criteria

PCG Preconditioned Conjugated Gradient

PVSC Piecewise Vanishing Stress Constraint

SAXPY Scalar a$\mathbf{x}$ plus $\mathbf{y}$

SIMP Solid Isotropic Material with Penalization

TO Topology Optimization

Tu te tornas eternamente responsável por aquilo que cativas.

Antoine de Saint-Exupéry, *O pequeno príncipe.*

1
Introduction and Motivation

Throughout human history, engineering design tools, from pen and paper to computer aided design, have mostly played a passive role in the design process. Most recently, however, structural optimization transforms this paradigm by allowing the computer to design optimal structures with minimal human interference. Not only that, structural optimization pushes the limits of structural design achieving an unprecedented level of performance and lightweightness. Among all of the existing techniques inside the structural optimization field, topology optimization (TO) [15, 16] is one of the most general because it is able to generate incredibly complex structures with intricate details for a wide range of applications. With this technique, we can design taller skyscrapers, safer cars, lighter airplanes that consume less fuels, just to cite a few examples of the endless possibilities that TO provides. However, most of the research in the field does not consider local failure criteria in the design process, and/or only accounts for a limited number of load cases in the optimization, which can lead to a structure that will break under normal operating conditions. Furthermore, topology optimization problems are computationally expensive, which prevents its widespread use to solve large-scale problems that are essential for designing engineering structures (see Fig. 1.1 for the main challenges addressed in this work).

Most developments in topology optimization focus on compliance minimization problems, which aim to find the stiffest structure for a given volume constraint. Because no limits on material strength are imposed, structures designed for minimum compliance do not necessarily withstand the applied loads, making some of these designs unfeasible for practical applications. Therefore, from a structural integrity standpoint, a more appropriate topology optimization formulation should aim to find the lightest structure that resists the applied loads without exceeding the material strength at any point. In this work, the material strength limitations are directly imposed in the TO formulation through local stress constraints, while minimizing the total weight of the structure. This stress-constrained weight minimization formulation leads to a more natural engineering design paradigm that generates practical optimized structures.

Furthermore, to make use of the full potential of TO capabilities, it is necessary to have a sufficiently refined mesh to represent the fine details of the optimal structure, which means that the problem size rapidly grows beyond our computational capabilities, specially for 3D problems. This creates the need for efficient large-scale TO techniques. Using parallel processing, the groundbreaking work [6] optimized an airplane wing using a mesh with over 1 billion elements, and by using such refined mesh they obtained a finely detailed micro-structure improved design, which surpasses traditional airplane wing design in weight, possibly saving billions of dollars in airplane fuel per year. Currently, the record for the largest TO problem ever solved is held by [12] with over 2.1 billion elements. However, to achieve such solution they required the use of the Joliet-Curie Supercomputer for over 85 hours, which results in a cost reaching hundredths of thousands of dollars. This high financial cost prevents the implementation of such large-scale solution techniques by the industry, limiting the impact of their research. This highlights the need for an economically viable large-scale approach that can be implemented with far less resources, and that can be widely applied by industry. Furthermore, both [6] and [12] used a compliance-based formulation, which is significantly simpler to solve.

At the same time, engineering structures are subjected to a multitude of loads over their lifespan [99]. For example, aircraft are subjected to constantly varying inertial loads; buildings are subjected to continuously-varying wind and live loads, in addition to the structure's self weight. These myriads of loading conditions cause an equally countless number of stress distributions that should be considered in design process to guarantee structural integrity. Nevertheless, topology optimization typically focuses on structures with a single load case or a small number of load cases, leading to designs that are over-fitted to an artificial single loading condition, and that are not robust under the service loads.

Currently, we lack an efficient implementation that can handle large-scale stress-constrained TO problems accounting for realistic load cases, and this has limited the application of topology optimization by the industry and military alike. In this work, we intent to address this gap in the literature by combining local failure criteria, large-scale TO with continuously varying load conditions (see Fig. 1.1). The combination of these approaches will allow us to design real life structures with practical applications and it will take topology optimization one step closer to the industrial and commercial use.

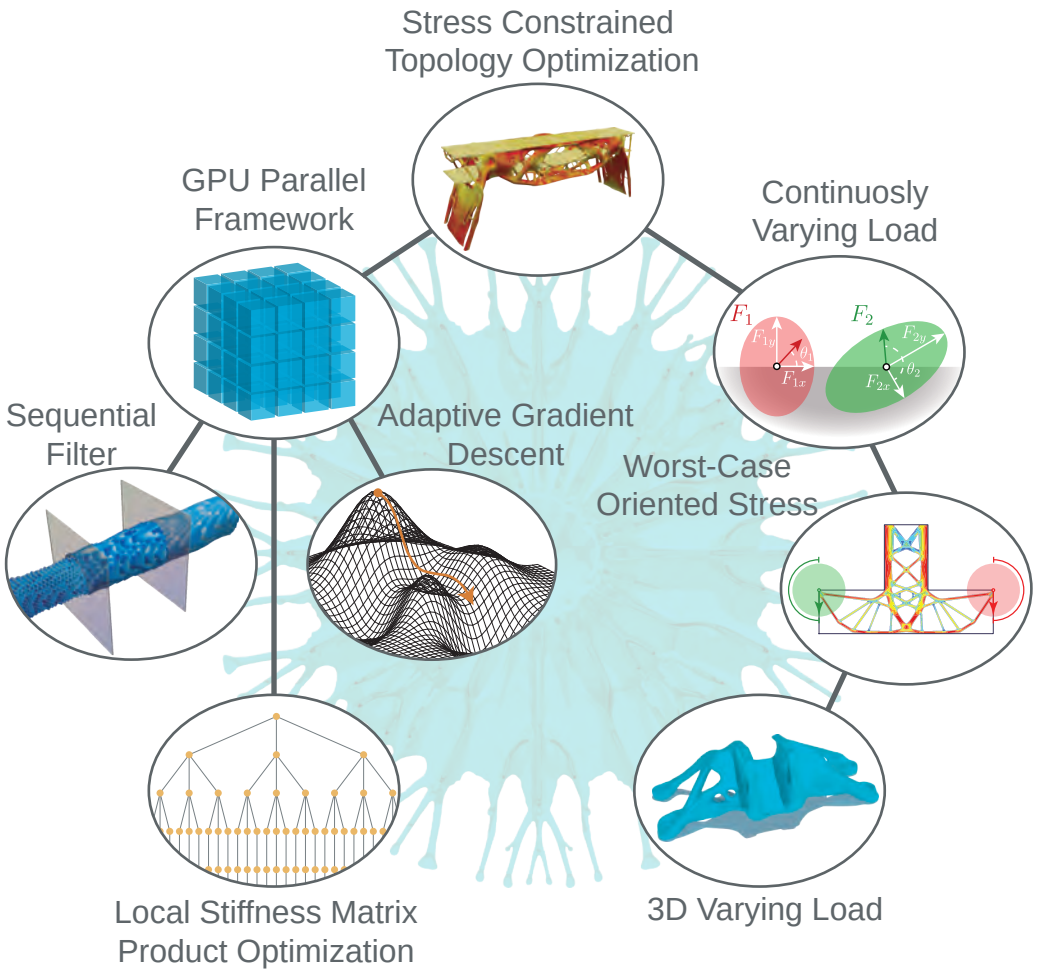


Figure 1.1: Concept map of the main contributions of this work.

1.1 Summary of the Main Contributions of This Work

This work focus on stress constrained TO to solve large-scale problems with realistic loading conditions. To this end, we developed a GPU-based parallel framework considering continuously varying load directions. In the development of this framework, the unifying thread was the optimization techniques and principles, which were used to improve the several aspects of a typical TO procedure, as displayed in Fig. 1.2, e.g. optimization was used to minimize the error of the sequential filter, to reduce the computational cost of the local stiffness matrix product, and to find analytical solutions to the worst-case oriented stress for continuously varying load case. To summarize, the main contributions of this work are:

- a **sequential filtering** technique with a low memory footprint;
- an **adaptive gradient-based optimization algorithm** that can easily be implemented in parallel;

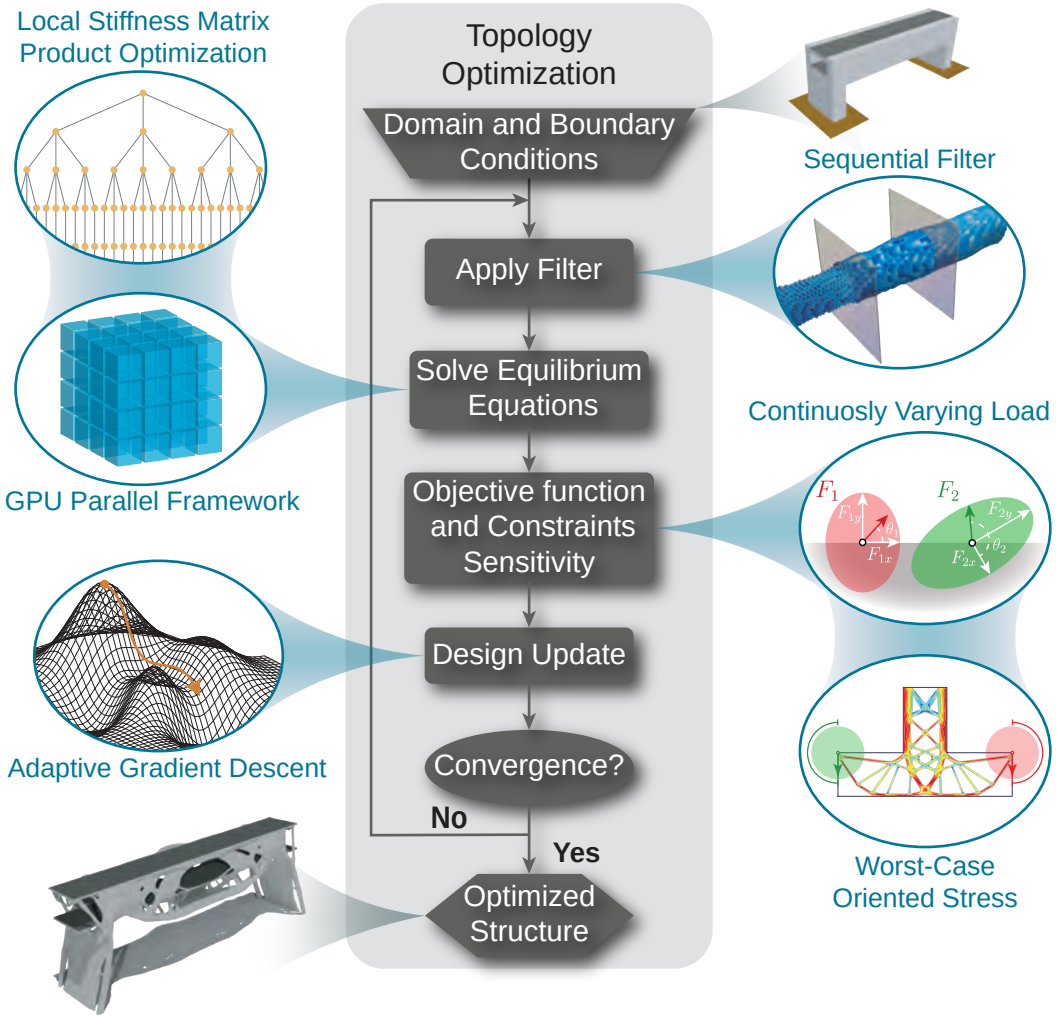


Figure 1.2: Flowchart of a general Topology Optimization framework displaying the contributions of this work on the many aspects of the optimization procedure.

- a **parallel GPU assembly-free preconditioned conjugated gradient solver** for the Finite Element Analysis (FEA) with and optimized local stiffness matrix product that can efficiently handle large-scale problems with unstructured meshes;
- the creation of **the Set Collapsing Problem**, which was mathematically proved to be NP-Hard;
- a **specifically tailored Branch-and-Bound scheme** that lead to a solution, which reduced the number of floating point operations of the local stiffness matrix product by *more than half*;
- a worst-case scenario technique to incorporate **continuously varying load** into stress-constrained TO based on analytical solutions that bound

the maximum stress caused by a range of admissible load angles. The analytical solutions considered:

- a single load varying in a limited range of angles;
- a load varying in direction plus a fixed load;
- two or more loads varying independently in direction;
- a load varying in 3D in a limited range of angles;

1.2 Thesis Outline

The remainder of this thesis is organized as follows. Next Section (Section 1.1) summarizes the main contributions of this work. Chapter 2 reviews the existing literature on stress-constrained topology optimization considering large-scale parallel implementations and multiple load cases. Chapter 3 presents the stress-constrained TO formulation and the AL-based approach used to handle the local stress constraints. Chapter 4 develops the large-scale parallel GPU framework for stress-constrained TO, and provides numerical results that benchmark its efficiency. Chapter 5 describes the multiple load direction framework with detailed derivations of the proposed analytic solutions for worst-case stress, and provides benchmark numerical examples obtained with the proposed method. Finally, Chapter 6 presents the concluding remarks, and discuss potential future works.

2
Literature Review

Stress-constrained TO requires a large number of stress evaluation points, and consequently a large number of stress constraints, to guarantee the structural integrity of the final design, because stress are local by nature. The large number of stress constraints demands high computational resources both for the sensitivity evaluation, and for the optimization problem, which makes the direct problem intractable [41].

To reduce the computational cost, most researchers have used aggregation techniques to combine all the local stress constraints into a single global constraint (e.g., see [42, 76, 77, 103, 122, 124]). The global constraint approximates the value of the maximum stress of the structure. The use of a global measure of stress reduces the computational cost at the expense of losing control over the local behavior of stress [42, 74]. The ability of aggregation functions to represent the local stress constraints depends on the number of constraints, and it can rapidly deteriorate as the number of constraints increases. To circumvent this issue, some researchers have used *clustering techniques*, in which the design domain is first divided in several sub-regions, each called a cluster, and then an aggregation function (e.g., the p-norm) is used to approximate the maximum stress value in each cluster [58, 74, 90]. However, the resulting topologies obtained using the aforementioned clustering techniques strongly depend on the number of clusters and on the way the clusters are defined. In general, it is expected that, as the number of clusters increases, one gains control over the local stress. However, there is no clear relation between the number of clusters and the quality of the optimized results [74].

An attractive approach to efficiently solve the stress-constrained TO problem is the Augmented Lagrangian (AL) method [19, 20]. The AL method directly handles the local stress constraints by adding them to the objective function in the form of a penalty term that is updated at each optimization step. Pereira et al. [91] used the AL method in the context of density-based topology optimization using relaxed local stress constraints [28]. Although promising, the strategy by Pereira et al. [91] appears to have difficulties finding 0/1 solutions at the end of the optimization steps. The AL method was also used by Emmendoerfer Jr. and Fancello, Emmendoerfer Jr. and Fancello

[43, 44] and by James et al. [62] in the context of the level set method. In the approach by Emmendoerfer Jr. and Fancello, Emmendoerfer Jr. and Fancello [43, 44], the AL method is used to treat the von Mises stresses as local quantities, and in the approach by James et al. [62], the AL method is used to enforce volume constraints, while the local von Mises stress values are aggregated using a p-norm aggregation function. The approach by Emmendoerfer Jr. and Fancello, Emmendoerfer Jr. and Fancello [43, 44] produces structures with clear boundaries that satisfy the stress constraints locally, but the algorithmic parameters required for the evolution of the level set may change from one problem to another, which undermines the robustness of their approach. Moreover, da Silva et al. [35] used the AL method for stress-constrained topology optimization considering manufacturing uncertainties via eroded, intermediate and dilated projections [106].

Another challenge in stress-constrained topology optimization problems that has received a lot of attention is related to the phenomenon of singular optima. This phenomenon was first reported by Sved and Ginos [112] in the context of stress-constrained truss optimization. The singular optima means that the stress-constrained optimization problem do not satisfy standard constraint qualification, and Achtziger and Kanzow [7] classified this type of optimization problem as a mathematical program with vanishing constraints (MPVCs). Expading on the work by Achtziger and Kanzow [7], Hoheisel and Kanzow [57] proposed several tailored versions of standard constraint qualification for MPVCs. The phenomenon of singular optima was extensively studied by other researchers [29, 67, 68, 69], and a thorough historical review on the subject can be found in [97]. The issue of singular optima has been alleviated by means of relaxation techniques such as the ε-relaxation [28]. Duysinx and Sigmund [42] modified the ε-relaxed constraint by Cheng and Guo [28] for use in the context of density-based topology optimization. As an alternative methodology to the ε-relaxation approach, Bruggi [25] proposed the so-called qp-relaxation technique. Achtziger et al. [8] proposed a smooth-regularization approach, and proved that this version of the problem satisfies standard constraint qualification.

In this work, we apply a modified version of the approach proposed by Senhora et al. [102], that uses an AL-based method with a modified vanishing-constraint formulation. Their formulation is able to efficiently, and consistently handle millions of constraints, which proved to be essential for the development of this work.

2.1 Large-scale GPU

Topology optimization is generally associated with a high computational cost that stems from the solution of the equilibrium equations imposing the physics of the problem (e.g., linear elasticity). The equilibrium equations are most commonly solved through the finite element method (FEM), which requires the discretization of the domain. For TO, the domain must be sufficiently refined in order to represent small details of the structure, and to accurately capture the physical behavior of the underlying structure. This high discretization requirement dramatically increases the computational cost, specially for 3D cases.

In order to address this high computational cost, the most popular strategy has been the use of parallel computing techniques for TO. However, as noted by Aage and Lazarov [3], the available literature on the subject is scarce, and incomplete. Recently, Mukherjee et al. [87] provided a broad review of the large-scale TO field state-of-the-art. Nonetheless, all the available research follow a similar strategy of employing parallel iterative solvers to obtain the solution of the equilibrium equation. Following this strategy, [3, 6, 12, 22, 37, 70, 82, 100, 105, 117], have implemented parallel frameworks to solve the compliance minimization problem in TO. Meanwhile, Kim et al. [66] solved a buckling-based problem, Aage et al. [5] focused on Stokes flow problems, and Evgrafov et al. [46] solved compliant mechanism problems. While all the previously mentioned works focused on CPU parallel processing, [27, 39, 56, 83, 84, 85, 87, 93, 94, 96, 100, 101, 118, 121, 127] used a GPU implementation achieving higher performance than their CPU counterparts, which demonstrates the promising power of the GPUs for parallel processing.

Expanding on the GPU TO previously mentioned works, Zegard and Paulino [127] provides an introduction to the challenges, and opportunities of GPU based TO system while solving compliance minimization 2D problems. Challis et al. [27] developed a C++/CUDA implementation with the Thrust library [13] to solve the inverse homogenization problem, and maximize the bulk modulus of a unit cell for an isotropic material. Martínez-Frutos and Herrero-Pérez [83] used a multi-GPU scheme to solve robust compliance minimization problems, i.e., compliance expectation minimization with uncertainties in the inputs (e.g., load uncertainty), and used the multiple GPUs for concurrent computations of the numerical approximation of the stochastic variables. Martínez-Frutos and Herrero-Pérez [84] proposed an evolutionary topology optimization framework using a fixed-grid FEM scheme, and analysed the influence of several preconditioners on the solution of the problem,

while Martínez-Frutos et al. [85] developed a similar work for density-based TO to solve compliance, compliant, and heat-sink problems. Ramírez-Gil et al. [94] addressed the multi-physics problem of designing a 3D electrothermomechanical actuator using GPU computing. In a similar multi-physics context, Ramírez-Gil et al. [94] optimized a fluid mixer, and heat exchanger, by using a Lattice Boltzmann method in a multi-GPU setting, to reduce the high computational cost associated to the fluid simulation. Also using a multi-GPU scheme, Herrero-Pérez and Castejón [56] partitioned the domain so that each partition was sent to a different GPU to be computed in parallel, and, with that, they were able to solve problems with over 50 million elements, but the cost of communication between GPUs negatively affected their speedup. On the collaborative spirit of the TO field, and similar to [9, 113], which freely shared educational codes, Schmidt and Schulz [101] developed a short and simple implementation of a GPU-based TO framework. In a similar trend, Duarte et al. [39] implemented a general C++/CUDA TO code that supported both CPU and GPU solutions, using both hexahedral and polygonal elements.

It is worth noting that the previously mentioned implementations are problem specific, and the stress constrained TO problem introduces distinct challenges that are not directly addressed by the previous mentioned works. In fact, the stress constrained large-scale TO literature is extremely scarce. And the only significant work found [75] uses CPU parallel processing to solve problems with over 14 million elements. To address this gap in the literature, we propose to implement a GPU-based parallel processing stress-constrained TO framework, based on the AL method.

2.2 Continuously Varying Load Case

The simplest way to incorporate multiple load cases in a stress-constrained TO formulation is to consider additional constraints for each load case; however, each additional load case substantially increases the computational cost, rendering it impossible to account for continuously-varying loads. Thus, this approach has typically been limited to no more than two fixed load cases [38, 76, 80, 92, 102]. Alternatively, the problem with multiple load cases has been formulated as an uncertainty in the load, which can be solved using stochastic or worst-case oriented approaches [14, 114].

In the stochastic approach, the load is treated as a random variable following a known probability distribution, and the objective and/or constraints of the optimization problem are modeled as statistical moments of random functions or as probabilities. Stochastic models have been widely applied to

compliance-based TO problems [10, 40, 53, 54, 61, 73, 78, 115, 129], but more moderately so for stress-based TO [36, 63, 79, 81]. Kanno and Takewaki [63] studied the effects of multiple load directions with stress constraints in the context of ground structure topology optimization using a probabilistic model to generate a sample load that represents the possible load cases. Lógó et al. [79] worked on the optimally conditions of multiple load cases and proposed a probabilistic model to address the problem with trusses. Luo et al. [81] considered uncertainty in the material and in the load amplitude by using reliability based topology optimization. Stochastic approaches, however, are troublesome in practice because they demand large data samples to obtain accurate results, which leads to high computational costs. The high computational cost can be mitigated using surrogate models [34], but this increases the complexity of the implementation. In addition, stochastic techniques can cause instability in the optimization procedure [130].

On the other hand, worst-case oriented approaches are equivalent to solving an optimization in the set of possible load cases in which the worst-case is identified, and used as the objective and/or constraint function in the TO problem. The solution to the worst-case oriented problem guarantees an upper bound for the objective function (in the case of minimization problems), and/or the satisfaction of the constraints for any load case considered in the set of possible load cases. Thore et al. [114] provides a worst-case oriented general framework for quadratic objective function and quadratic constraints under load uncertainty by recasting the problem as a non-linear, semi-definite programming problem which is then solved numerically. Young et al. [125], and Xie and Steven [123] use an evolutionary structural optimization approach and consider multiple load cases with stress constraints by computing a finite element analysis for each load case. Csébfalvi [33] deals specifically with uncertainty in load directions with an iterative approach. Holmberg et al. [59] proposed a game theory approach to solve the worst-case oriented stress constraints problems considering variation in the load direction; however, because the problems are non-convex, the existence of solution to the game proposed by Holmberg et al. [59] is not guaranteed, and they rely on empirical observation of the numerical results.

In this work, we propose a worst-case oriented approach, but, unlike the previously mentioned approaches, we use the linear state equations and the bilinear properties of the von Mises stress to derive an analytic solution for the worst-case stress caused by continuously-varying loads. We then use these worst-case stresses as the local stress constraints in the TO problem. These analytic solutions, which are one of the main contributions of this work, are

accurate, computationally efficient and guarantee the structural integrity of the final design over the set of continuously varying loading conditions.

3
Stress-Constrained Topology Optimization Formulation

The stress constrained TO problem has two main challenges:

1. Stress is a local quantity that must be satisfied point-wise, and, because of that, one needs to impose a large number of constraints, leading to a prohibitive computational cost;

2. The optimal solution of a stress-constrained problem generally lies on a degenerated low-dimensional region of the solution domain. This phenomenon is called singular optima.

To address these challenges, we employ an AL approach that preserves the local nature of stress, and a variation of the piecewise vanishing constraint [102] that can reach the singular optima. This formulation is efficient because it only requires one adjoint vector computed trough the solution of a linear system for the computation of the sensitivity, which is then used on a gradient-based optimization algorithms. By treating the stress constraints locally, we obtain optimized structures that satisfy all the stress constraints.

3.1
Basic Stress-constrained topology optimization formulation

This section presents a framework for the solution of stress-constrained topology optimization problems based on the AL method. In this work, we focus on the stress constrained mass minimization described by the optimization statement:

$$\begin{aligned} \min_{\mathbf{z}} \quad & m(\mathbf{z}) = \sum_{e=1}^{N_e} \tilde{\rho}_e(\mathbf{z}) v_e \\ \text{s.t.:} \quad & g_j(\mathbf{z}) \leq 0, \quad j = 1, \ldots, N_c \\ & 0 \leq z_e \leq 1, \quad e = 1, \ldots, N_e \\ \text{with:} \quad & \mathbf{K}(\mathbf{z})\mathbf{U} = \mathbf{F} \\ & \tilde{\rho}(\mathbf{z}) = \mathcal{H}(\mathbf{P}\mathbf{z}) \end{aligned} \tag{3-1}$$

where $m(\mathbf{z})$ is the mass (volume) of the structure, $\mathbf{z}$ is the vector of design variables, $\tilde{\rho}_e$ is the density of element e defined using a filter operation and a

smooth Heaviside projection [120], v_e is the area (for 2D problems) or volume (for 3D problems) of element e, $g_j(\mathbf{z})$ is the j-th stress constraint, N_c is the number of stress constraints, N_e is the number of elements in the finite element mesh, and $\mathbf{K}(\mathbf{z})\mathbf{U} = \mathbf{F}$ is the FEM discretization of the linear elastic equilibrium equation, which is solved numerically.

The physical density vector, denoted by $\tilde{\rho}$, is obtained by first applying the density filter operator [23, 126], and then the Heaviside operation [120] to the design variable $\mathbf{z}$. The filter operation is computed by multiplying the design variable vector $\mathbf{z}$ by the matrix $\mathbf{P}$, whose entries P_{ij} are defined as:

$$P_{ij} = \frac{w_{ij}v_j}{\sum_{k=1}^{N_e} w_{ik}v_k}, \quad \text{with} \quad w_{ij} = \max\left[0, 1 - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2}{r}\right]^s, \tag{3-2}$$

$$\boldsymbol{\rho} = \mathbf{P}\mathbf{z} \tag{3-3}$$

where r is the filter radius, v_j is the volume of element j, and $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ represents the distance between the centroids, $\mathbf{x}_i$ and $\mathbf{x}_j$, of elements i and j, respectively. The order of the filter is defined by the filter exponent, s. Note that, when $s = 1$, the polynomial filter reduces to the traditional linear filter [23]. The filtered variables are then defined as $\boldsymbol{\rho}$ for convenient notation. After we apply the filter operation, we perform the smooth Heaviside projection [120]:

$$\tilde{\boldsymbol{\rho}} = \mathcal{H}(\mathbf{P}\mathbf{z}) = \frac{\tanh(\beta\eta) + \tanh(\beta(\boldsymbol{\rho} - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \tag{3-4}$$

where η is the value of the threshold for the Heaviside function (in this work $\eta = 0.5$), and β controls the sharpness of such function. The stiffness matrix, $\mathbf{K}$, is computed through a typical assembly process as:

$$\mathbf{K}(\mathbf{z}) = \mathop{\mathbb{A}}_{e=1}^{N_e} \mathbf{k}_e, \quad \text{with} \quad \mathbf{k}_e = E_e\mathbf{k}_0, \tag{3-5}$$

$$E_e = \epsilon_{min} + (1 - \epsilon_{min})\tilde{\rho}_e^p \tag{3-6}$$

where $\mathbf{k}_e$ are the element stiffness matrices, ϵ_{min} is the Ersatz parameter (in this work $\epsilon_{min} = 10^{-6}$), p is the SIMP penalization factor, $\mathbf{k}_0$ is the stiffness matrix for a solid element, and E_e is the material interpolation stiffness function.

3.1.1 Piecewise vanishing stress constraint

In this work, we use a modified version of the piecewise vanishing stress constraint (PVSC) [102] defined as:

$$g_j(\mathbf{z}) = \begin{cases} \tilde{\rho}_j^p \left[0.1 \left(\dfrac{\sigma_j^v}{\sigma_{\text{lim}}} - 1 \right) + \left(\dfrac{\sigma_j^v}{\sigma_{\text{lim}}} - 1 \right)^2 \right] & \text{, if } \dfrac{\sigma_j^v}{\sigma_{\text{lim}}} > 1 \\ 0.1\, \tilde{\rho}_j^p \left(\dfrac{\sigma_j^v}{\sigma_{\text{lim}}} - 1 \right) & \text{, otherwise,} \end{cases} \tag{3-7}$$

in which the exponent p is the SIMP penalization factor, σ_j^v is the von Mises stress at evaluation point j, and σ_{lim} is the stress limit. This exponent factor p helps to regularize the behavior of the constraint by correlating it with the behavior of the local stiffness matrix as a function of the density. We denote this constraint as the *modified piecewise vanishing constraint* (MPVSC) and we use it in the optimization statement in Eq. (3-1). Figure 3.1 displays the behavior of the MPVSC in comparison to the PVSC as we vary the value of σ^v/σ_{lim}. In this plot, we can see that the PVSC becomes flat, i.e. has a null first derivative, for any value equal to or below 1, which can cause instability in the optimization procedure. Meanwhile, the MPVSC never has a null first derivative, leading to a more well-behaved problem formulation.

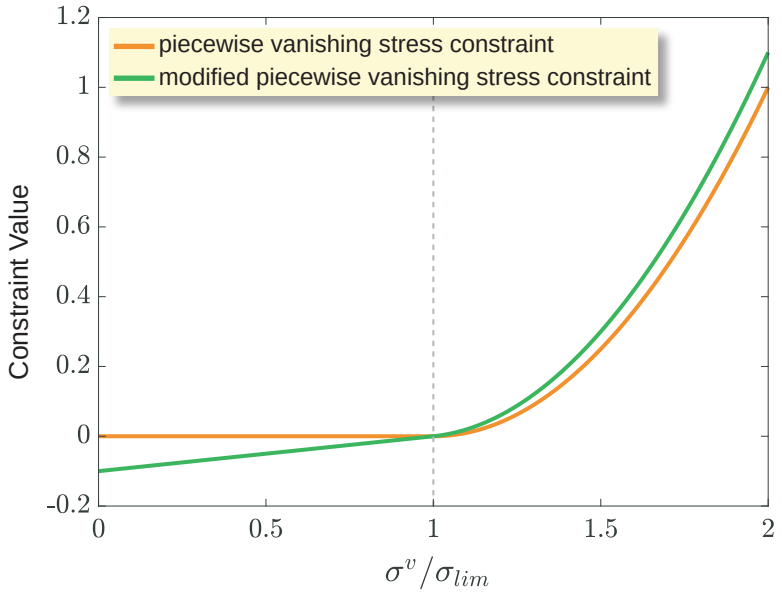


Figure 3.1: Plot comparing the piecewise vanishing stress constraint [102] to the proposed modified piecewise vanishing stress constraint (Eq. (3-7)).

3.1.2 Stress Measure

In order to have a consistent and meaningful stress measure, we define:

$$\tilde{\sigma}^v = \tilde{\rho}^{0.5} \sigma^v, \tag{3-8}$$

which we will use to evaluate the stress on the optimized structures. This stress measure, $\widetilde{\sigma}^v$, was based on the *qp*-relaxation approach [25], and is computed with the elastic properties of the solid material, using the apparent "local" von Mises stress [41, 42].

3.2 Augmented Lagrangian Method (AL)

The AL method [19, 20, 88] is a numerical technique to solve constrained optimization problems. In the AL method, the solution to a constrained optimization problem is achieved by solving a series of unconstrained problems that converge to the original solution. For instance, suppose that we want to use the AL method to solve the optimization problem:

$$\begin{aligned} &\min_{\mathbf{z}\in\mathbb{R}^n} \quad f(\mathbf{z}) \\ &\text{s.t.:} \quad h_j(\mathbf{z}) = 0 \quad \forall j = 1, \ldots, N_c \end{aligned} \tag{3-9}$$

in which $\mathbf{z}$ is the vector of design variables, $f(\mathbf{z})$ is the objective function, $h_j(\mathbf{z})$ are the equality constraints, and N_c is the number of constraints. Using the AL method, we transform the optimization problem in Eq. (3-9), into the unconstrained optimization problem:

$$\min_{\mathbf{z}\in\mathbb{R}^n} \quad J^{(k)}(\mathbf{z}) = f(\mathbf{z}) + \sum_{j=1}^{N_c} \left[\lambda_j^{(k)} h_j(\mathbf{z}) + \frac{\mu^{(k)}}{2} h_j(\mathbf{z})^2 \right], \tag{3-10}$$

where $\lambda_j^{(k)}$ is an estimate of the Lagrange multiplier of $h_j(\mathbf{z})$, $\mu^{(k)}$ is a penalty coefficient, and k indicates the k-th step of the AL method. Both $\lambda_j^{(k)}$ and $\mu^{(k)}$ are updated at every step k. The AL function, $J^{(k)}(\mathbf{z})$, is an approximation of the Lagrangian of the constrained problem (3-9), but with the extra term $\frac{1}{2}\mu^{(k)} h_j(\mathbf{z})^2$.

The solution $\mathbf{z}^{(k)}$ of the approximate sub-problem in Eq. (3-10) converges to the solution $\mathbf{z}^*$ of the original problem (3-9) as $k \to \infty$, given that the original problem satisfies some regularity conditions [88]. Particularly, Bertsekas [20] proved that if both the objective function and the constraints are continuous, the original problem has an optimum, and every sub-problem has an optimum, then the sequence of optima points of the sub-problems converges to an optimum point of the original problem. Assuming that problem (3-10) is well-behaved, the first-order optimality condition states that:

$$\nabla J^{(k)}(\mathbf{z}^{(k)}) = \nabla f(\mathbf{z}^{(k)}) + \sum_{j=1}^{N_c} \left[\lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)}) \right] \nabla h_j(\mathbf{z}^{(k)}) = \mathbf{0}. \tag{3-11}$$

Comparing Eq. (3-11) with the KKT optimality conditions for problem (3-9) implies that:

$$\lambda_j^* \nabla h_j(\mathbf{z}^*) \approx \left[\lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)})\right] \nabla h_j(\mathbf{z}^{(k)}), \tag{3-12}$$

from which we obtain:

$$\lambda_j^* \approx \lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)}) \tag{3-13}$$

or

$$h_j(\mathbf{z}^{(k)}) \approx \frac{\lambda_j^* - \lambda_j^{(k)}}{\mu^{(k)}}. \tag{3-14}$$

Equation (3-13) provides a means for updating the Lagrange multipliers, $\lambda_j^{(k)}$, at every step k, as follows:

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)}), \quad \forall j = 1, \ldots, N_c. \tag{3-15}$$

From Eq. (3-14), we observed that $h_j(\mathbf{z}^{(k)})$ is proportional to $\lambda_j^* - \lambda_j^{(k)}$ and inversely proportional to $\mu^{(k)}$. Thus, a good estimation of the Lagrange multipliers and a large value of $\mu^{(k)}$ improve the convergence of AL methods to a feasible solution (i.e., $h_j(\mathbf{z}) = 0$). In theory, when $\boldsymbol{\lambda}^{(k)}$ is a good estimate of the actual Lagrange multiplier vector, one can obtain a good estimate of $\mathbf{z}^*$ by solving problem (3-10). A proper value for $\mu^{(1)}$ needs to be chosen carefully because a relatively high initial value for this parameter may lead to ill-conditioning of the optimization problem [19, p. 123]. The usual recommendation found in the literature (e.g., see [19, 20, 88]) is to start with a moderate value of $\mu^{(1)}$ and gradually increase it according to:

$$\mu^{(k+1)} = \alpha \mu^{(k)}, \tag{3-16}$$

where $\alpha > 1$ is a constant. The value of α is also problem dependent and may require empirical adjustment. To solve optimization problems using the AL method, one needs to solve the unconstrained optimization statement (3-10) at each step k and update both Lagrange multipliers $\lambda_j^{(k)}$ and penalty term $\mu^{(k)}$, using Eqs. (3-15) and (3-16), respectively. The procedure is repeated until some convergence criterion is satisfied. The AL method presented here is designed for equality constraints, and extended for inequality constraints in Subsection 3.2.1.

3.2.1
Augmented Lagrangian method for inequality constraints

The procedure described in Section 3.2 is designed to solve optimization problems with equality constraints. Here, we expand the AL method for inequality constraints. Consider the following optimization problem with inequality constraints:

$$\begin{aligned} \min_{\mathbf{z}\in\mathbb{R}^n} \quad & f(\mathbf{z}) \\ \text{s.t.:} \quad & g_j(\mathbf{z}) \leq 0 \quad \forall j = 1, \ldots, N_c \\ & \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}, \end{aligned} \tag{3-17}$$

where $\mathbf{l}$ and $\mathbf{u}$ define the lower and upper bounds of the design variables, respectively. Introducing slack variables, constraints $g_j(\mathbf{z}) \leq 0$ are rewritten as:

$$h_j(\mathbf{z}) = g_j(\mathbf{z}) + s_j = 0, \;\; s_j \geq 0, \;\; j = 1, \ldots, N_c. \tag{3-18}$$

Consequently, the approximate sub-problem that needs to be solved at the k-th step of the AL method is:

$$\begin{aligned} \min_{\mathbf{z},\ \mathbf{s}} \quad & J^{(k)}(\mathbf{z}, \mathbf{s}) = f(\mathbf{z}) + \sum_{j=1}^{N_c} \left[\lambda_j^{(k)} \left(g_j(\mathbf{z}) + s_j \right) + \frac{\mu^{(k)}}{2} \left(g_j(\mathbf{z}) + s_j \right)^2 \right] \\ \text{s.t.:} \quad & \mathbf{u} \leq \mathbf{z} \leq \mathbf{l} \\ & s_j \geq 0 \quad \forall j = 1, \ldots, N_c. \end{aligned} \tag{3-19}$$

The minimization of $J^{(k)}(\mathbf{z}, \mathbf{s})$ with respect to the slack variables is obtained explicitly for any fixed $\mathbf{z}$ by solving the optimization problem:

$$\begin{aligned} \min_{s_j} \quad & \left[\lambda_j^{(k)} \left(g_j(\mathbf{z}) + s_j \right) + \frac{\mu^{(k)}}{2} \left(g_j(\mathbf{z}) + s_j \right)^2 \right] \\ \text{s.t.:} \quad & s_j \geq 0. \end{aligned} \tag{3-20}$$

The optimization statement (3-20) is defined in terms of the slack variable, s_j, associated to constraint g_j. As a result, its solution can be found in closed form using the stationary conditions of the Lagrangian of (3-20), which leads to:

$$s_j = \max \left[0, -\left(\frac{\lambda_j^{(k)}}{\mu^{(k)}} + g_j(\mathbf{z}) \right) \right] \quad \forall j = 1, \ldots, N_c. \tag{3-21}$$

Substituting Eq. (3-21) into Eq. (3-18) leads to:

$$h_j(\mathbf{z}) = \max\left[g_j(\mathbf{z}), -\frac{\lambda_j^{(k)}}{\mu^{(k)}}\right] \quad \forall j = 1, \ldots, N_c. \tag{3-22}$$

Using Eq. (3-22), the inequality constraints $g_j(\mathbf{z}) \leq 0$ of (3-17) can be replaced by equality constraints, allowing the problem to be solved using the procedure described for solving the equality-constrained problem (3-9). As inferred from Eq. (3-22), the slack variables do not need to be computed explicitly, facilitating the implementation of the AL method with inequality constraints. One must recall that the Lagrange multiplier estimators, $\lambda_j^{(k)}$, and the penalty factor, $\mu^{(k)}$, remain constant for each AL sub-problem, and thus the AL function is continuously differentiable (with respect to the design variables) at each AL step. Despite the presence of the maximum function, the AL function used with Eq. (3-22) is differentiable even at the points in which $g_j(\mathbf{z}) = -\lambda_j^{(k)}/\mu^{(k)}$ ([20] p. 161).

3.2.2
Modified Augmented Lagrangian method and the Scale Factor η

We apply the AL method to solve the optimization problem in Eq. (3-1). We introduce a slight modification to the AL method by introducing a scale factor η that multiplies the penalty term of the AL function. The AL function for the k-th sub-problem is:

$$J^{(k)}(\mathbf{z}) = \sum_{e=1}^{N_e} \tilde{\rho}_e v_e + \eta \sum_{j=1}^{N_c} \left[\lambda_j^{(k)} h_j(\mathbf{z}) + \frac{\mu^{(k)}}{2} h_j(\mathbf{z})^2\right] \tag{3-23}$$

in which $h_j(\mathbf{z})$ is defined as $\max\left[g_j(\mathbf{z}), -\frac{\lambda_j^{(k)}}{\mu^{(k)}}\right]$, with $g_j(\mathbf{z})$ being the MPVSC defined in Eq. (3-7). The scale factor η is introduced because the progress towards a feasible solution using the traditional AL method depends on the ratio between the original objective function and the penalty term of the AL function. Therefore, the magnitude of the penalty term is highly dependent on the number of constraints. In the context of topology optimization with local stress constraints, the number of constraints increases with the number of elements in the mesh. Through numerical experimentation, we found that if the objective-to-penalty ratio in the AL function is kept approximately constant and independent of the number of constraints, our AL formulation leads to consistent optimization independent of the mesh size. To preserve this ratio for different mesh sizes, we multiply the penalty term by the scale factor η, which is given by:

$$\eta = 1/N_c, \tag{3-24}$$

where N_c is the number of constraints. The proposed scale factor η helped us obtain consistent optimization results for a variety of problems solved in the present study, in which the number of constraints ranged between a few thousands to over 35 million.

3.2.3 Addressing non-convexity

The stress-constrained TO problem is non-convex. Consequently, it is common for optimization algorithms to get trapped in local optima. To mitigate this effect, and possibly escape from unfavorable local optima, we adopt a common strategy with the AL method for non-convex problems that is to restart the values of μ and λ_j when the optimization stagnates[1] [19, 20, 88]. This approach allows us to achieve better results, i.e., designs with lower mass ratio than that obtained when these parameters are not restarted. The effect of restarting these parameters in the quality of the optimization results is demonstrated in [102].

3.3 Sensitivity analysis

The stress-constrained topology optimization problem discussed in this section is solved using gradient-based optimization algorithms. In order to do so, sensitivity information for the AL function (3-23) is required. The sensitivity of the AL function is computed using the chain rule as:

$$\begin{aligned}\frac{\mathrm{d}J^{(k)}}{\mathrm{d}z_j} &= \sum_{i=1}^{N_e} \frac{\partial J^{(k)}}{\partial \tilde{\rho}_i} \frac{\mathrm{d}\tilde{\rho}_i}{\mathrm{d}\rho_i} \frac{\mathrm{d}\rho_i}{\mathrm{d}z_j} = \\ &= \sum_{i=1}^{N_e} \frac{\partial J^{(k)}}{\partial \tilde{\rho}_i} \frac{\tanh(\beta\eta)}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))} \beta \left[1 - \tanh^2(\beta(\rho_i - \eta))\right] P_{ij}\end{aligned} \tag{3-25}$$

The term P_{ij} in Eq. (3-25) is obtained from the relation $\boldsymbol{\rho}(\mathbf{z}) = \mathbf{P}\mathbf{z}$ and the term $\partial J^{(k)}/\partial \rho_i$ is obtained using Eq. (3-23), as follows:

$$\frac{\partial J^{(k)}}{\partial \tilde{\rho}_i} = \frac{\partial}{\partial \tilde{\rho}_i} \sum_{e=1}^{N_e} \tilde{\rho}_e v_e + \eta \frac{\partial}{\partial \tilde{\rho}_i} \sum_{j=1}^{N_c} \left[\lambda_j h_j(\mathbf{z}) + \frac{\mu}{2} h_j(\mathbf{z})^2\right]. \tag{3-26}$$

For simplicity in the notation, we have dropped the superscript k in Eq. (3-26) and in the subsequent equations of this section. The second part of Eq. (3-26), which is related to the penalty term, is computed as:

[1] Stagnation is reached when the average change in the design variables between two consecutive iterations is smaller than a given tolerance.

$$\frac{\partial}{\partial\tilde{\rho}_i}\sum_{j=1}^{N_c}\left[\lambda_j h_j(\mathbf{z})+\frac{\mu}{2}h_j(\mathbf{z})^2\right]=\sum_{j=1}^{N_c}\left[\lambda_j+\mu h_j(\mathbf{z})\right]\frac{\partial h_j(\mathbf{z})}{\partial\tilde{\rho}_i}. \tag{3-27}$$

Using Eq. (3-7) and (3-22), the non-zero part of the sensitivity of constraints h_j is determined as follows:

$$\begin{aligned}\frac{\partial h_j(\mathbf{z})}{\partial\tilde{\rho}_i}=&\frac{\partial}{\partial\tilde{\rho}_i}\tilde{\rho}_j^p\left[(\sigma_j^v/\sigma_{\lim}-1)^2+0.1(\sigma_j^v/\sigma_{\lim}-1)\right]\\ =&p\tilde{\rho}_j^{p-1}\delta_{ij}\left[(\sigma_j^v/\sigma_{\lim}-1)^2+0.1(\sigma_j^v/\sigma_{\lim}-1)\right]+\\ &+\frac{\tilde{\rho}_j^p}{\sigma_{\lim}}\left[2(\sigma_j^v/\sigma_{\lim}-1)+0.1\right]\left(\frac{\partial\sigma_j^v}{\partial\mathbf{U}}\right)^T\frac{\partial\mathbf{U}}{\partial\rho_i}\end{aligned} \tag{3-28}$$

where δ_{ij} is the Kronecker delta operator and $\mathbf{U}$ is the displacement vector obtained from the equilibrium equation $\mathbf{KU}=\mathbf{F}$. The last part of Eq. (3-28) corresponds to the sensitivity of the von Mises stress for the jth stress constraint. The adjoint method is used herein to avoid the expensive computation of $\partial\mathbf{U}/\partial\tilde{\rho}_i$ [17, 31]. Differentiating the aforementioned equilibrium equation with respect to the design variables, and assuming that $\mathbf{f}$ is independent of the design variables, we obtain:

$$\frac{\partial\mathbf{K}}{\partial\tilde{\rho}_i}\mathbf{U}+\mathbf{K}\frac{\partial\mathbf{U}}{\partial\tilde{\rho}_i}=\mathbf{0} \tag{3-29}$$

Substituting Eq. (3-28) into Eq. (3-27) and adding the expression in Eq. (3-29) multiplied by the adjoint vector, $\boldsymbol{\xi}$, leads to:

$$\begin{aligned}&\frac{\partial}{\partial\tilde{\rho}_i}\sum_{j=1}^{N_c}\left[\lambda_j h_j(\mathbf{z})+\frac{\mu}{2}h_j(\mathbf{z})^2\right]=\\ &\sum_{j=1}^{N_c}\left[\lambda_j+\mu h_j(\mathbf{z})\right]p\tilde{\rho}_j^{p-1}\delta_{ij}\left[(\sigma_j^v/\sigma_{\lim}-1)^2+0.1(\sigma_j^v/\sigma_{\lim}-1)\right]+\\ &+\sum_{j=1}^{N_c}\left[\lambda_j+\mu h_j(\mathbf{z})\right]\frac{\tilde{\rho}_j^p}{\sigma_{\lim}}\left[2(\sigma_j^v/\sigma_{\lim}-1)+0.1\right]\left(\frac{\partial\sigma_j^v}{\partial\mathbf{U}}\right)^T\frac{\partial\mathbf{U}}{\partial\tilde{\rho}_i}+\\ &+\boldsymbol{\xi}^T\left(\frac{\partial\mathbf{K}}{\partial\tilde{\rho}_i}\mathbf{U}+\mathbf{K}\frac{\partial\mathbf{U}}{\partial\tilde{\rho}_i}\right),\end{aligned} \tag{3-30}$$

Collecting all terms in Eq. (3-30) that multiply $\partial\mathbf{U}/\partial\tilde{\rho}_i$ and choosing $\boldsymbol{\xi}$ such that these terms vanish from the sensitivity evaluation allows rewriting Eq. (3-30) as:

$$\frac{\partial}{\partial\tilde{\rho}_i}\sum_{j=1}^{N_c}\left[\lambda_j h_j(\mathbf{z})+\frac{\mu}{2}h_j(\mathbf{z})^2\right]=\left[\lambda_i+\mu h_i(\mathbf{z})\right]p\tilde{\rho}_i^{p-1}\left[(\sigma_j^v/\sigma_{\text{lim}}-1)^2+\right.$$
$$\left.+0.1(\sigma_j^v/\sigma_{\text{lim}}-1)\right]+\boldsymbol{\xi}^T\frac{\partial\mathbf{K}}{\partial\tilde{\rho}_i}\mathbf{U}, \tag{3-31}$$

where $\boldsymbol{\xi}$ is the solution to the following adjoint problem:

$$\mathbf{K}\boldsymbol{\xi}=-\sum_{j=1}^{N_c}\left[\lambda_j+\mu h_j(\mathbf{z})\right]\frac{\tilde{\rho}_j^p}{\sigma_{\text{lim}}}\left[2(\sigma_j^v/\sigma_{\text{lim}}-1)+0.1\right]\frac{\partial\sigma_j^{VM}}{\partial\mathbf{U}} \tag{3-32}$$

The last term in Eq. (3-31) is obtained as $\boldsymbol{\xi}^T\frac{\partial\mathbf{K}}{\partial\tilde{\rho}_i}\mathbf{U}=\boldsymbol{\xi}_i^T\frac{\partial\mathbf{k}_i}{\partial\tilde{\rho}_i}\mathbf{U}_i$, where $\boldsymbol{\xi}_i$, $\mathbf{k}_i$, and $\mathbf{U}_i$ refer to element-wise quantities. Because we compute the element stiffness matrices using Eq. (3-5), then:

$$\frac{\partial\mathbf{k}_i}{\partial\tilde{\rho}_i}=p(1-\epsilon)\tilde{\rho}_i^{p-1}\mathbf{k}_0. \tag{3-33}$$

The final expression for the sensitivity of the AL function (3-23) is given by:

$$\frac{\mathrm{d}J^{(k)}}{\mathrm{d}z_j}=\sum_{i=1}^{N_e}\left\{v_i+\eta\left(\lambda_i+\mu h_i(\mathbf{z})\right)p\tilde{\rho}_i^{p-1}\left[(\sigma_j^v/\sigma_{\text{lim}}-1)^2+\right.\right.$$
$$\left.\left.+0.1(\sigma_j^v/\sigma_{\text{lim}}-1)\right]+p(1-\epsilon)\tilde{\rho}_i^{p-1}\boldsymbol{\xi}_i^T\mathbf{k}_0\mathbf{U}_i\right\} \tag{3-34}$$
$$\frac{\tanh(\beta\eta)}{\tanh(\beta\eta)+\tanh(\beta(1-\eta))}\beta\left[1-\tanh^2(\beta(\rho-\eta))\right]P_{ij}$$

Note that using the adjoint vector, $\boldsymbol{\xi}$, obtained from Eq. (3-32) greatly reduces the cost of computing the sensitivity as compared to computing Eq. (3-28) directly, which requires the computation of $\partial\mathbf{U}/\partial\tilde{\rho}_i$ for $i=1,\ldots,N_c$.

4
Toward Large-Scale GPU-based Stress-Constrained Topology Optimization

Large-scale problems are essential for the general application of TO, because they allow for the fully detail design of complex structures with high physical accuracy. Furthermore, large-scale TO has pushed the boundary of material design by enabling the direct optimization of the micro-structure. To solve large-scale TO problems, we will implement parallel computing strategies in a GPU framework. GPUs have thousands of cores that can process extensive amounts of data in parallel, making them powerful devices for the solution of large-scale TO problems.

The GPU-based parallelization of the TO procedure encounters three main challenges: the filter, the optimization algorithm, and the solution of the equilibrium equations by means of the FEM. In the next Sections we will address each of these challenges individually to achieve a parallel GPU scheme capable of solving large-scale stress constrained TO problems.

4.1
Large-Scale Filter in Parallel

The filter [24, 26, 107, 128] is an important part of every TO framework, because it solves two practical problems of TO [18]:

1. **Checkerboard patterns:** The filter prevents checkerboard patterns in the solution. Checkerboard patterns appear due to the numerical discretization of the domain, and the use of low order elements in the FEA, which overestimate the stiffness-to-weigh ratio of such patterns;

2. **Minimum length-scale:** The filter controls the minimum length-scale of the TO result, i.e. the filter prevents structural features smaller than a certain size (determined by the choice of filter radius), which is important from manufacturing point of view, because small structural features are troublesome to manufacture in practice.

The filter solve this problems by restricting the space of admissible density distribution to that of continuous and smooth fields. This restriction to

continuous and smooth fields is accomplished by applying a convolution operation of a continuous function, with local support, over the design variable distribution, ensuring the smoothness of the convoluted output. The convolution operation is described by equation:

$$\rho(\mathbf{x}) = k \star z = \int k(\tau) z(\mathbf{x} - \tau) d\tau \tag{4-1}$$

in which ρ is the filtered density field, k is the kernel function, $\star$ is the convolution operation, z is the design field, $\mathbf{x}$ is the position variable, and τ is the convolution variable. The most common filter kernel used is the linear hat kernel given by the equation:

$$k(\mathbf{x}) = \max\left[0, \left(1 - \frac{\|\mathbf{x}\|_2}{R}\right)\right]. \tag{4-2}$$

The most traditional, and computationally efficient way to apply the filter operation is, as previously described in Section 3.1, pre-computing the discretized convolution parameters given in Eq. 3-2, and writing them in a sparse filter matrix format. The filter operation is then reduced to a simple matrix-vector operation which can be done in $O(n^2)$ time, in which n is the size of the vector (in this case the number of design variables). However, the problem with this approach is the memory requirement to store such sparse filter matrix. Each row "i" of this matrix has to store the distance between element "i" and its neighbors for which such distance is smaller than the filter radius. Considering a 3D case with homogeneous mesh, the average characteristic size[1] of an element follows a $O(1/n^{1/3})$ trend for a given fixed domain. We can then estimate the number of neighbor elements that fall inside the filter radius by diving the volume of a sphere of such radius by the characteristic volume of the elements given by the cube of the characteristic size:

$$O\left(\frac{\frac{4}{3}\pi R^3}{\left(\frac{1}{n^{1/3}}\right)^3}\right) = O\left(n\right). \tag{4-3}$$

Considering that the filter matrix has n rows, the memory requirements, for such matrix, grows with $O(n^2)$. The quadratic growth of the memory requirement rapidly becomes prohibitive, as we deal with large-scale problems, specially for GPUs, which have considerably limited memories. To give an idea of how fast the memory requirement can grow, notice that, a filter that requires only 100 MB with a mesh of 100,000 elements, would require almost 10 GB

[1]Characteristic size here refers to a measure of length-scale that represents the geometry of the element, e.g. the side length of a cube

with a mesh of 1 million elements, and almost 1 TB with a mesh of 10 million elements.

Because of this issue, alternatives to the pre-computation and storage of the filter matrix have been proposed. One approach is to directly compute the convolution operation using discrete Fourier Transform (DFT) algorithms, which can be performed with significantly limited memory. However, such algorithms are not computationally efficient, as they present an asymptotic complexity of $O(n^3 \log^3(n))$ (3D). Other techniques replace the convolutional filter, with a filter based on elliptic PDEs [65, 72, 119], more specifically an Helmholtz-like PDE:

$$-R^2\nabla^2\rho + \rho = z \tag{4-4}$$

In the PDE Filter approach, Eq. 4-4 is solved to obtain the filtered variable ρ. The main advantage of this approach is the limited memory requirement, and the efficiency, which is dictated by the algorithms used to solve the PDE that can be implemented in parallel. However, the PDE filter introduces further complexity to the formulation, both theoretically and implementation-wise. The approach that we propose in Section 4.1.1 combines limited memory requirements and efficiency, while being remarkably simple to implement.

4.1.1 Sequential Filter

The sequential filter combines the idea of PDE filter technique, and the pre-computed filter matrix convolution. Similar to the pre-computed filter matrix, we pre-compute a sparse matrix that represents a discrete convolution of the density field with a continuous locally supported kernel function. The difference is that we choose the kernel function such that its support is restricted to the elements in the immediate vicinity of an element. Because of this limited support of the kernel, we only need to store the neighboring elements of each element. Once we have this local filter matrix, we apply it to the density field repeatedly, similar to a "time" evolution of a time-dependent PDE. This accomplishes an extended smoothing effect that approximates a convolution with a kernel function reaching a larger support region comparable to traditional filtering techniques. However, unlike the traditional filtering techniques, the memory requirements for this technique grows linearly with the problem size, $O(n)$, because we only store a fixed number of neighbours for each element. Mathematically, this sequential filter operation is equivalent to a sequence of convolutions, i.e.:

$$\rho = \overbrace{\mathtt{k} \star \mathtt{k} \star \cdots \star \mathtt{k}}^{r \text{ iterations}} \star z \tag{4-5}$$

in which ρ is the filtered density field, $\mathtt{k}$ is the kernel function, $\star$ is the convolution operation, z is the design field, and r is the number of iterations of the convolution. The associative properties of the convolution operation allow us to compute the term $\mathtt{k}\star\mathtt{k}\star\cdots\star\mathtt{k}$, i.e. the r iterations of the convolution of the kernel $\mathtt{k}$ with itself. This computation is done for analysis purposes only, since pre-computing this term would result in a computational cost comparable to the traditional filter. We propose the following equation to estimate the number of iterations r of the kernel:

$$r = \left\lfloor a \left(\frac{R}{R_{\mathtt{k}}} \right)^{b} \right\rfloor \tag{4-6}$$

where $R_{\mathtt{k}}$ is the radius of action of the local kernel $\mathtt{k}$, R is the radius of effect that we want to achieve, and a and b are numerical parameters. The parameters a and b are determined by minimizing the mean absolute error of the sequential filter in relation to the traditional linear filter that we would like to replicate. In order to evaluate the error we choose discrete values of $R/R_{\mathtt{k}}$ ranging from 2 to 14. We then minimize the mean absolute error of these points using the Nelder-Mead simplex search algorithm [89]. Solving this minimization problem lead us to the following equation for r:

$$r = \left\lfloor 1.07 \left(\frac{R}{R_{\mathtt{k}}} \right)^{2.14} \right\rfloor \tag{4-7}$$

Figure 4.1 displays the mean absolute error obtained using this formulation for different values of $R/R_{\mathtt{k}}$. From this plot we can see that the error decreases with the value of $R/R_{\mathtt{k}}$. This is because the local filter acts like a discretization of the linear filter, and as we increase the value of $R/R_{\mathtt{k}}$, we obtain a finer discretization.

Figure 4.2 illustrates the term $\mathtt{k} \star \mathtt{k} \star \cdots \star \mathtt{k}$ in 3D, in comparison with a linear kernel, considering r iterations computed with Eq. (4-7). The example shown in Fig. 4.2 considers a cube of side length 2, a radius $R = 1$, and a radius for the local kernel $R_{\mathtt{k}} = 0.15$, meaning that this local kernel was applied 44 times.

Algorithm 1 displays the procedure to implement the sequential filter and it has as inputs the matrix representation, $\mathbf{P}_{\mathtt{k}}$, of the local kernel $\mathtt{k}$, the vector of design variables $\mathbf{z}$, the desired equivalent radius of the linear filter R, and the radius of the local kernel $R_{\mathtt{k}}$. Considering that the local kernel radius is proportional to the element characteristic size, and that

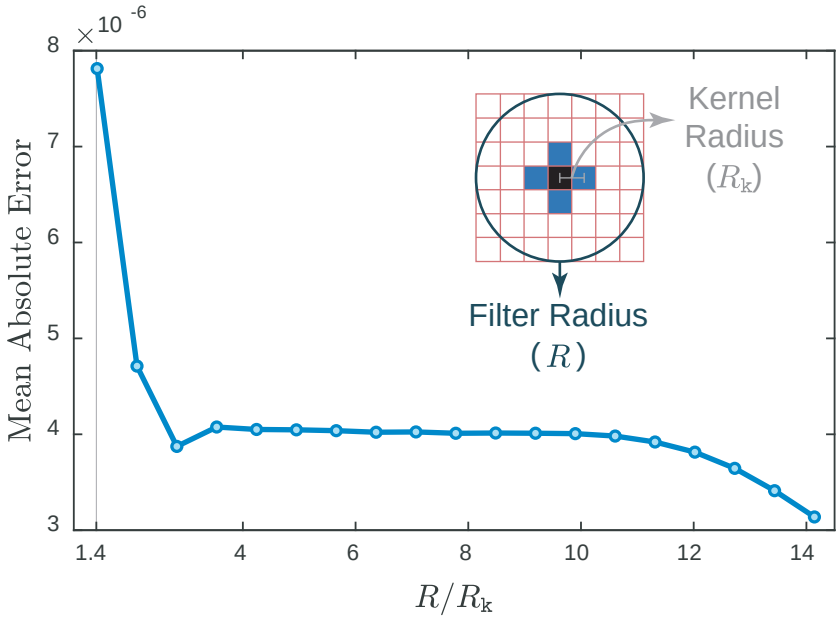


Figure 4.1: Error of the sequential filter in relation to the traditional linear filter as a function of the ratio of the filter radius over the local kernel radius.

the characteristic size of an element is proportional to $1/n^{1/3}$, we have that $R_{\mathtt{k}} \propto 1/n^{1/3}$[2], and, consequently, $r = O(n^{2.14/3})$. Meaning that we have to perform $r = O(n^{2.14/3})$ matrix-vector products of the local filter matrix with the design variables. However, notice that the local filter matrix has a fixed number of non-zero components at each row, meaning that, due to its sparsity, we can compute the matrix-vector product in $O(n)$ operations. Considering this, the asymptotic analysis of the efficiency of the sequential filter algorithm give us an $rO(n) = O(n^{2.14/3}n) = O(n^{5.14/3}) < O(n^2)$ performance. In practice, the cost of the sequential filter is negligible when compared to the cost of the FEA required in each TO iteration. Another advantage of the sequential filter is the easiness to change the filter radius R, which, in the sequential filter algorithm, just required a change in the input of the algorithm, in contrast to traditional filter approaches that would require the full re-computation of the filter matrix.

4.2 Optimization Algorithm

The optimization algorithm is responsible for optimizing the structure by updating the design variables, and it is at the core of any TO implementation. The most traditional optimization algorithms for TO are the Optimality Criteria (OC) [55, 86, 98], and the Method of Moving Asymptotes (MMA)

[2] $\propto$ is the mathematical symbol for *proportional to*.

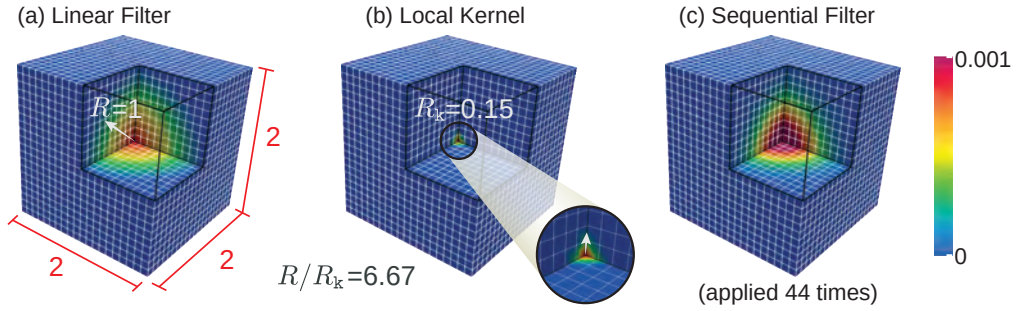


Figure 4.2: Schematic of the effect of the traditional linear filter, the local kernel, and the sequential filter over a 3D mesh.

Algorithm 1 Sequential Filter Algorithm

1: **procedure** SEQUENTIAL_FILTER($\mathbf{P_k}$, $\mathbf{z}$, R, R_k)
2: $\boldsymbol{\rho} \leftarrow \mathbf{z}$
3: **for** $i = 0$ **to** $\left\lfloor (R/R_k)^2 \right\rfloor$ **do**
4: $\boldsymbol{\rho} \leftarrow \mathbf{P}\boldsymbol{\rho}$
5: **end for**
6: return $\boldsymbol{\rho}$
7: **end procedure**

[4, 109, 110]. The OC is very limited, and can only solve optimization problems with one constraint, for which said constraint must be active at the optimum. Furthermore, the OC can be very unstable for problems with high non-linearity, and the optimization path generated by the OC can only transverse a subspace of the design domains in which the constraint is active, making it extremely susceptible to local optima for non-convex problems. Some other OC based algorithms have been developed that can accommodate more than one constraint, such as the ZPR [131], but they suffer from the same shortcomings. Despite this disadvantages, the OC and OC-like methods are still very popular, because of their efficiency, and easiness to implement for compliance problems that tend to be particularly well-behaved.

On the other hand, the MMA is a very powerful and general optimization algorithm that can handle a wide range of different optimization problems. Because of the freely available MMA codes provided by the author of [110], the MMA is widely disseminated in the TO community. However, the general and powerful capabilities of the MMA come at a high computational cost, because the MMA generates a convex approximated optimization problem that needs to be solved iteratively. This iterative inner solve, which can be significantly expensive, determines the update of the design variables. Several MMA-based approaches have been developed, such as the GCMMA [111], which guarantees the convergence of the algorithm, the TRMMA [60], which presents improved

convergence, and the EMMA [21], which incorporate second order information, but none of them address the high computational cost.

Because of the previously mentioned difficulties associated with the OC and the MMA, a different algorithm was necessary for the large-scale solution of stress constrained TO problems. For this reason, we developed a adaptive gradient descent-based (AGD) approach that is efficient, simple to implement, and easily parallelizable. Notice that we handle the constraints of the optimization problem with the Augmented Lagrangian method, and, therefore, we are left with an unconstrained optimization problem. Algorithm 2 describes the main steps of the proposed algorithm.

Algorithm 2 Adaptive Gradient Descent

procedure MODIFIED_GRADIENT_DESCENT($\mathbf{z}$, f, $df/d\mathbf{z}$, f_{old1}, f_{old2}, $\mathbf{z}_{min}$, $\mathbf{z}_{max}$, α, $\texttt{move}$, $\texttt{norm}_0$)
  if $(f_{\text{old1}} - f_{\text{old2}})(f - f_{\text{old1}}) < 0$ **then**
    $\alpha \leftarrow \max(\, 0.25\alpha,\ 10^{-6})$
  else
    $\alpha \leftarrow \min(\, 1.25\alpha,\ 1)$
  end if
  $\mathbf{m}_r \leftarrow \texttt{move}\ (\mathbf{z}_{max} - \mathbf{z}_{min})$
  $\mathbf{d}_f \leftarrow \dfrac{\alpha}{\texttt{norm}_0} \dfrac{df}{d\mathbf{z}}$
  $\mathbf{z} \leftarrow \max(\max(\min(\min(\mathbf{z} - \mathbf{d}_f, \mathbf{z} + \mathbf{m}_r), \mathbf{z}_{max}), \mathbf{z} - \mathbf{m}_r), \mathbf{z}_{min})$
  $f_{\text{old2}} \leftarrow f_{\text{old1}}$
  $f_{\text{old1}} \leftarrow f$
end procedure

In Algorithm 2, $\mathbf{z}$ are the design variables, f is the value of the objective function, $df/d\mathbf{z}$ is the gradient of the objective function in respect to the design variables, f_{old1} and f_{old2} are the values of the objective function in the last two iterations, $\mathbf{z}_{min}$ and $\mathbf{z}_{max}$ are the lower and upper bound of the design variables respectively, α is adaptive parameter that damps oscillation by controlling the size of the step taken by the optimizer, $\texttt{move}$ is move limit (generally set to 0.05), and $\texttt{norm}_0$ is a normalizing parameter for the gradient. The value of this normalizing parameter is set to $\texttt{norm}_0 = \|(df/d\mathbf{z})_0\|_2 / \texttt{move}$, in which $(df/d\mathbf{z})_0$ is the gradient at the first iteration of the optimization. For some problems, it might be beneficial to update the value of $\texttt{norm}_0$ every 50 iterations or so to accelerate convergence.

To compare the performance of the proposed AGD algorithm with the MMA, we will solve the diagonal square problem proposed by [102]. The problem domain is represented in Fig. 4.3(a), in which we have a four element

square mesh fixed at the lower left element, with a load applied at the upper right node. The elements at the anti-diagonal have a fixed density of 1 (i.e. solid passive zone), while the density of the two elements in the main diagonal are the design variables of the problem. The mathematical optimization statement of the diagonal square problem is displayed in Eq. (4-8). Figure 4.3(b) displays the design domain of this problem in which we can see the feasible region, the constraints, the objective function, and the optimization path performed by the AGD (Blue) and the MMA (Gray) algorithms. We can see that the AGD describes a smooth path towards the optimum taking a total of 24 iterations, while the MMA presents a more ragged and oscillatory behavior towards the optimum taking a total of 32 iterations.

$$
\begin{aligned}
\min_{(Z_1,Z_2)} \quad & Z_1 + Z_2 \\
\text{s.t.:} \quad & g_1 = \frac{\sigma_1^V}{\sigma_{\text{lim}}} - 1 \leq 0, \quad Z_1 > 0 \\
& g_2 = \frac{\sigma_2^V}{\sigma_{\text{lim}}} - 1 \leq 0, \quad Z_2 > 0 \\
& 0 \leq \rho_i \leq 1, \quad i = 1, 2 \\
\text{with:} \quad & \mathbf{K}(Z_1, Z_2)\mathbf{U} = \mathbf{F},
\end{aligned}
\tag{4-8}
$$

4.3 Finite Element Analysis

The FEA is the most computationally expensive part of the TO procedure. As it is demonstrated trough numerical experiments in Section 4.5 (see Fig. 4.12(b)), the computational cost of the FEA easily surpass 99% of the total computational time. Not only that, the solution of the FEA also requires a substantial amount of RAM memory, which can be the limiting factor for the size of problems one is able to solve. Therefore, this is the most crucial part of the implementation when dealing with large-scale TO problems.

In this work, we are only considering a linear elastic model for the mechanical behavior of the structure, and therefore, the FEA comes down to the solution of a single linear system per iteration, which is represented by the stiffness matrix.[3] To solve this linear system, the use of direct methods (e.g. Cholesky, LU decomposition) rapidly become prohibitive because of high memory requirements, and, therefore, we will resort to iterative methods, more specifically, the preconditioned conjugated gradient method (PCG) [51, 104, p. 628]. The PCG presents several advantages over other linear system solver:

[3]For a detailed explanation of the FEA method, and the computation of the stiffness matrix see [95].

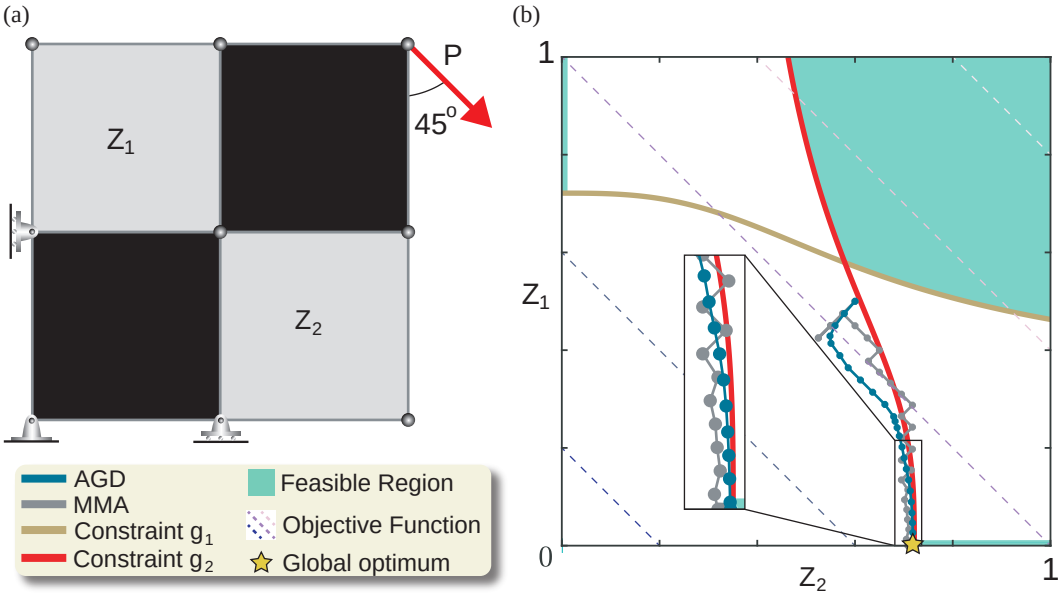


Figure 4.3: Diagonal Square problem proposed by [102]. (a) Problem domain, design variables (Z_1 and Z_2), and boundary conditions. (b) Optimization domain of the problem presented in Eq. (4-8), displaying the feasible region, the constraints, the objective function, the global optimum, and the optimization path performed by the AGD (24 iterations) and the MMA (32 iterations) algorithm. In this plot, we can clearly see the smooth path taken by the AGD, in comparison with the ragged and oscillatory path taken by the MMA.

- PCG is an iterative solver, meaning that we do not need to decompose the stiffness matrix, leading to a lower RAM memory requirement;
- PCG finds the best solution at each step in the subspace that it has explored. That is, for step k, and initial guess $\mathbf{u}_0$, PCG will have the best solution possible for the problem in the Krylov subspace $\mathcal{K}(\mathbf{K}; \mathbf{u}_0) = \text{span}\{\mathbf{u}_0, \mathbf{K}\mathbf{u}_0, \ldots, \mathbf{K}^k\mathbf{u}_0\}$;
- PCG only requires three types of operations, dot product, vector add/subtract, and matrix vector product, which can be easily implemented in parallel.

A disadvantage of the PCG method is that it only guarantees convergence for symmetric positive definite matrices. However, the system that we are trying to solve, generated by linear elastic FEA, are indeed symmetric and positive definite, meaning that this limitation of the PCG is not a issue for us.

4.3.1 Preconditioned Conjugated Gradient (PCG)

In this Section we will provide more details about the Preconditioned Conjugated Gradient method. First we will give a brief introduction to the

Conjugated Gradient method (CG), and then we will discuss the importance of the preconditioner that easily extends the CG method to the PCG method. CG is a linear solver that belongs to the class of iterative Krylov methods. Methods of this class search for the best solution in the Krylov subspace of the problem. The Krylov subspace is generated by the matrix that represents the linear system, and by a initial guess of the solution. To solve the system $\mathbf{KU} = \mathbf{f}$ through the PCG method, we iteratively solve a sequence of optimization problems:

$$\min_{\mathbf{U} \in \mathcal{K}_k(\mathbf{K}, \mathbf{r}_0)} \quad \phi(\mathbf{U}) = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} - \mathbf{U}^T\mathbf{F}, \tag{4-9}$$

in which $\mathbf{r}_0 = \mathbf{K}\mathbf{U}_0 - \mathbf{F}$, $\mathbf{U}_0$ is a initial guess, and $\mathcal{K}_k(\mathbf{K}, \mathbf{r}_0)$ is the Krylov subspace defined as:

$$\mathcal{K}_k(\mathbf{K}, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0,\ \mathbf{K}\mathbf{r}_0,\ \mathbf{K}^2\mathbf{r}_0,\ \mathbf{K}^3\mathbf{r}_0, \cdots,\ \mathbf{K}^{k-1}\mathbf{r}_0\right\}$$

Notice that the only change from the optimization problem (4-9) of iteration k to iteration $k+1$ is the Krylov subspace that defines the solution domain of the problem. Another special property of optimization problem (4-9) is that, if matrix $\mathbf{K}$ is positive-definite, the optimization problem is convex, and therefore has only one solution, because the second derivative of the objective function (i.e. the Hessian of the problem) is the matrix $\mathbf{K}$ itself. This means that any critical point of the objective function $f(\mathbf{U})$ is a solution to the optimization problem. To identify critical points we compute the first derivative of $f(\mathbf{U})$, and set it equal to zero:

$$\frac{\partial \phi(\mathbf{U})}{\partial \mathbf{U}} = \mathbf{K}\mathbf{U} - \mathbf{F} = \mathbf{0}, \tag{4-10}$$

demonstrating that a critical point of $f(\mathbf{U})$ is also a solution to the linear system $\mathbf{KU} = \mathbf{f}$. To solve the optimization problem in Eq. (4-9), we can make use of the fact that the iterations of the solution domains $\mathcal{K}_k(\mathbf{K}, \mathbf{r}_0)$, form a sequence of nested subspaces, i.e.:

$$\mathcal{K}_0(\mathbf{K}, \mathbf{r}_0) \subset \mathcal{K}_1(\mathbf{K}, \mathbf{r}_0) \subset \cdots \subset \mathcal{K}_k(\mathbf{K}, \mathbf{r}_0).$$

Furthermore, we can iteratively define an orthonormal basis for the sequence of nested subspaces. Let $V_{k-1} = \{\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_{k-1}\}$ be and orthonormal basis for $\mathcal{K}_{k-1}(\mathbf{K}, \mathbf{r}_0)$, then an orthonormal basis of $\mathcal{K}_k(\mathbf{K}, \mathbf{r}_0)$ can be created by orthonormalizing $\mathbf{K}^{k-1}\mathbf{r}_0$ with respect to the set $\mathbf{v}_{k-1}$, and then adding the orthonormal resulting vector, $\mathbf{v}_k$, to that set $V_k = V_{k-1} \cup \{\mathbf{v}_k\}$. This sequence of orthonormal basis also gives us a formula to find the optimum of the

optimization problem in Eq. (4-9):

$$\mathbf{U}_{\star}^{(k)} = \mathbf{U}_{\star}^{(k-1)} - \mu_k \mathbf{v}_k \tag{4-11}$$

To find the value of μ_k, we compute:

$$\frac{\partial \phi(\mathbf{U}_{\star}^{(k)})}{\partial \mu} = -\left(\mathbf{v}_k\right)^T \mathbf{K} \left(\mathbf{U}_{\star}^{(k-1)} - \mu_k \mathbf{v}_k\right) + \left(\mathbf{v}_k\right)^T \mathbf{F} = 0 \tag{4-12}$$

$$\mu_k = \frac{\left(\mathbf{v}_k\right)^T \left(\mathbf{K}\mathbf{U}_{\star}^{(k-1)} - \mathbf{F}\right)}{\left(\mathbf{v}_k\right)^T \mathbf{K} \left(\mathbf{v}_k\right)} = \frac{\left(\mathbf{v}_k\right)^T \mathbf{v}_k}{\left(\mathbf{v}_k\right)^T \mathbf{K} \left(\mathbf{v}_k\right)} \tag{4-13}$$

To prove that the optimum of the optimization problem in Eq. (4-9) is of the form displayed in Eq. (4-11), suppose that there exist a point $\widehat{\mathbf{U}}_{\star}^{(k)} = \mathbf{U}_{\star}^{(k)} - \alpha \mathbf{u}$, with $\mathbf{u} \in \mathcal{K}_{k-1}$, such that $\phi\left(\widehat{\mathbf{U}}_{\star}^{(k)}\right) < \phi\left(\mathbf{U}_{\star}^{(k)}\right)$. Then we can compute the optimal value of α in the same way that we computed the optimal value of μ_k:

$$\frac{\partial \phi(\widehat{\mathbf{U}}_{\star}^{(k)})}{\partial \alpha} = -\left(\mathbf{u}\right)^T \mathbf{K} \left(\mathbf{U}_{\star}^{(k)} - \alpha \mathbf{u}\right) + \left(\mathbf{u}\right)^T \mathbf{F} = 0 \tag{4-14}$$

$$\alpha = \frac{\left(\mathbf{u}\right)^T \left(\mathbf{K}\mathbf{U}_{\star}^{(k)} - \mathbf{F}\right)}{\left(\mathbf{u}\right)^T \mathbf{K}\mathbf{u}} \tag{4-15}$$

The term in the numerator between parenthesis, $r_k = \left(\mathbf{K}\mathbf{U}_{\star}^{(k-1)} - \mathbf{F}\right)$, has the special property of being orthogonal to the subspace $\mathcal{K}_{k-1}$ (see [116, p. 296]). Therefore, if $r_k \perp \mathcal{K}_{k-1}$, and $\mathbf{u} \in \mathcal{K}_{k-1}$, we have that $\alpha = 0$, and $\widehat{\mathbf{U}}_{\star}^{(k)} = \mathbf{U}_{\star}^{(k)}$. Proving that Eq. (4-11) is the optimum. For more details on the conjugated gradient, please refer to [51, 104, p. 628]. The mathematical formulation described before leads to Algorithm 3.

Each iteration of the CG algorithm requires the computation of one matrix-vector product, two dot products, and three `scalar ax plus y` (SAXPY) operations. From all of the previously mentioned operations, the matrix-vector product is the most expensive, and it is responsible for most of the computational time spent in each iteration. Furthermore, the dot product and SAXPY operations are straightforward, therefore, in the next Sections we will focus our efforts in optimizing the matrix-vector operation. For the dot product and SAXPY operations we used the standard and highly efficient cuBLAS library [1][4].

[4]cuBLAS is a highly efficient CUDA library that provides GPU-accelerated linear algebra routines.

Algorithm 3 Conjugated gradient algorithm

procedure CONJUGATED GRADIENT($\mathbf{K}$, $\mathbf{F}$, $\mathbf{U}_0$)
  $\mathbf{r}_0 := \mathbf{F} - \mathbf{K}\mathbf{U}_0$
  $\mathbf{p}_0 := \mathbf{r}_0$
  $k := 0$
  $\omega_0 := \mathbf{r}_k \cdot \mathbf{r}_k$
  while norm($\mathbf{r}_k$) > Tol **do**
    $\mathbf{q}_k := \mathbf{K}\mathbf{p}_k$
    $\mu_k := \omega_k / (\mathbf{p}_k \cdot \mathbf{q}_k)$
    $\mathbf{U}_{k+1} := \mathbf{U}_k + \mu_k \mathbf{p}_k$
    $\mathbf{r}_{k+1} := \mathbf{r}_k - \mu_k \mathbf{q}_k$
    $\omega_{k+1} := \mathbf{r}_{k+1} \cdot \mathbf{r}_{k+1}$
    $\beta_k := \omega_{k+1} / \omega_k$
    $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
    $k := k + 1$
  end while
end procedure

Another matter that deserves attention is the convergence of the CG algorithm, i.e. how the error of the approximate solution changes in each iteration. Fortunately, we have an analytical expression that bounds the error of the solution at each iteration developed by [116, p. 299]:

$$\left\|\mathbf{U}_\star - \mathbf{U}^{(k)}\right\|_{\mathbf{K}} \leq 2 \left\|\mathbf{U}_\star - \mathbf{U}^{(0)}\right\|_{\mathbf{K}} \left(\frac{\{\kappa_2(\mathbf{K})\}^{1/2} - 1}{\{\kappa_2(\mathbf{K})\}^{1/2} + 1} \right)^k \tag{4-16}$$

where k_2 is the condition number of the input matrix. Unfortunately, this upper bound on the error is dependent on κ_2, meaning that ill-conditioned matrices might require a large number of iterations to achieve a satisfactory convergence. To improve the condition number of a matrix, preconditioners are generally applied to the system. The idea of preconditoning is to transform the system $\mathbf{K}\mathbf{U} = \mathbf{f}$ as:

$$\mathbf{M}_1\mathbf{K}\mathbf{M}_2\mathbf{Y} = \mathbf{M}_1\mathbf{F}, \qquad \text{with } \mathbf{M}_2^{-1}\mathbf{U} = \mathbf{Y} \tag{4-17}$$

in which $\mathbf{M}_1$ is the left preconditioner and $\mathbf{M}_2$ is the right preconditioner. If the condition number of the resulting matrix $\mathbf{M}_1\mathbf{K}\mathbf{M}_2$ is better than the condition number of $\mathbf{K}$, we are able to greatly reduce the number of iterations necessary to solve the system. The final solution can then be obtained by solving $\mathbf{M}_2^{-1}\mathbf{U} = \mathbf{Y}$. From this, we can derive a set of requirements for a good preconditioner:

1. The preconditioners $\mathbf{M}_1$ and $\mathbf{M}_2$ must improve the condition number of $\mathbf{K}$;

2. The preconditioner must be easy to compute, that is, the computational cost of applying the preconditioner cannot be greater than the computational cost saved by the reduction of the number of iterations;

3. The preconditioners must be memory efficient to be able to handle large-scale problems.

The GPU framework makes requirements 2 and 3 more complicated, because, not only the preconditioner must be easy to compute, but it must be easy to compute in parallel, and we have to deal with the limited memory available for the GPU. For this reason, we chose to use the simple and efficient Jacobi preconditioner defined by $\mathbf{M}_1 = \mathbf{M}_2 = diag(\mathbf{K})^{-1/2}$, in which $diag(\mathbf{K})$ designates a diagonal matrix composed of the main diagonal of $\mathbf{K}$. To apply the preconditioner, however, it is not necessary to the directly compute $\mathbf{M}_1\mathbf{K}\mathbf{M}_2$. We can incorporate the preconditioner by defining the matrix $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$, as shown in Algorithm 4 (for more details see [51, p. 651]):

Algorithm 4 Preconditioned conjugated gradient algorithm

procedure PRECONDITIONED CONJUGATED GRADIENT($\mathbf{K}$, $\mathbf{F}$, $\mathbf{U}_0$,$\mathbf{M}$)
  $\mathbf{r}_0 := \mathbf{F} - \mathbf{K}\mathbf{U}_0$
  $\mathbf{z}_0 := \mathbf{M}\mathbf{r}_0$
  $\mathbf{p}_0 := \mathbf{r}_0$
  $k := 0$
  $\omega_0 := \mathbf{z}_k \cdot \mathbf{r}_k$
  while norm($\mathbf{r}_k$) > Tol **do**
    $\mathbf{q}_k := \mathbf{K}\mathbf{p}_k$
    $\mu_k := \omega_k / (\mathbf{p}_k \cdot \mathbf{q}_k)$
    $\mathbf{U}_{k+1} := \mathbf{U}_k + \mu_k\mathbf{p}_k$
    $\mathbf{r}_{k+1} := \mathbf{r}_k - \mu_k\mathbf{q}_k$
    $\mathbf{z}_{k+1} := \mathbf{M}\mathbf{r}_{k+1}$
    $\omega_{k+1} := \mathbf{z}_{k+1} \cdot \mathbf{r}_{k+1}$
    $\beta_k := \omega_{k+1}/\omega_k$
    $\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
    $k := k + 1$
  end while
end procedure

4.3.1.1 Matrix-Vector Product and Assembly-free Method

This Section explains the computation of the matrix-vector product necessary for the PCG algorithm. In order to perform such matrix-vector product operation in large-scale problems, we adopt an assembly-free method [11]. An assembly-free method does not require the full assembly of the matrix

representing the linear system. This is beneficial because the assembly of the matrix generally demands a large amount of RAM memory even when state-of-the-art sparse representations are used, and, not only that, the memory requirement grows with $O(n^2)$ which quickly prevents the assembly of the matrix for large-scale problems. This means that methods that require the assembled matrix, such as direct method (e.g. Cholesky, LU decomp), are severely limited for large-scale systems.

To compute a matrix-vector product, or any other operation using an assembly-free method, one must consider such operation as the composition of several smaller operations. For example, if $\mathbf{A} = \mathbf{B} + \mathbf{C} + \mathbf{D} + \mathbf{E}$, one could compute $\mathbf{Ax}$ as $\mathbf{Bx}+\mathbf{Cx}+\mathbf{Dx}+\mathbf{Ex}$, and in this way, the explicit computation of $\mathbf{A} = \mathbf{B} + \mathbf{C} + \mathbf{D} + \mathbf{E}$ is not necessary. In simple cases, the advantage of computing $\mathbf{Bx}+\mathbf{Cx}+\mathbf{Dx}+\mathbf{Ex}$, instead of just summing up the matrices that compose $\mathbf{A}$, and computing $\mathbf{Ax}$ directly might not be apparent. However, when matrix $\mathbf{A}$ presents clear patterns, or formation rules, assembly-free methods can be extremely efficient. Luckily, the stiffness matrix that comes from the linear elastic FEM presents a formation pattern that can be exploited for the computation of the matrix-vector product. The formation pattern can be clearly seen when we examine the global stiffness matrix definition as:

$$\mathbf{K}(\mathbf{z}) = \mathop{\mathbb{A}}_{e=1}^{N_e} \mathbf{k}_e$$

in which the symbol $\mathop{\mathbb{A}}_{e=1}^{N_e}$ represents the assembly process of the global stiffness matrix. In this equation, we see that the stiffness matrix is nothing more than the sum of the contributions of the local stiffness matrix $\mathbf{k}_e$ of each finite element of our mesh. We will use the individual contributions of the local stiffness matrix to compute an the assembly-free matrix-vector product in what is called an Element-by-Element approach (EbE) [39]. Furthermore, if we use a uniform mesh, i.e. all of the mesh elements have the same geometry, we only need to compute one local stiffness matrix, significantly reducing the amount of memory, and computation required for the solution of the linear system. For this reason we chose to restrict our meshes to one single type of element, the hexagonal element (Brick 8) displayed in Fig. 4.4. In the particular case of TO, each element of the mesh is also associated to a optimization z_e variable that controls its stiffness $E_e(\mathbf{z})$. Therefore the assembly process becomes:

$$\mathbf{K}(\mathbf{z}) = \mathop{\mathbb{A}}_{e=1}^{N_e} E_e(\mathbf{z})\mathbf{k}_e,$$

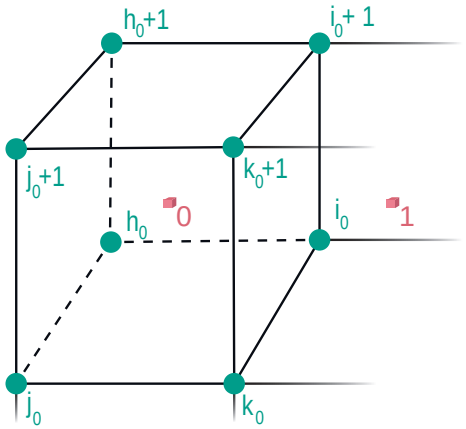
but since $E_e(\mathbf{z})$ is a scalar we can still exploit this formation pattern for

an assembly-free matrix-vector product operation. Notice that, to perform such operation, we need to know the $\mathop{\mathbb{A}}_{e=1}^{N_e}$, i.e. how each local stiffness matrix contributes to the global stiffness matrix. This information tell us how the entries of the matrices $\mathbf{k}_e$ are summed up to form the entries of $\mathbf{K}$, and this information depends on the connections of the elements of the mesh, i.e. which elements share a node, and how many degrees of freedom (DOF) each node has (3 DOF's for each in the linear elastic case). If we are dealing with a structured mesh, i.e. a mesh for which we can derive a simple formation rule that tell us the connection between elements, we do not need to store any connectivity information, and the DOF contribution of each local $\mathbf{k}_e$ can be computed on demand. However, this severely restricts the geometry of the domain that we can optimize. A workaround would be defining passive zones, i.e. elements which we do not account for in the optimization, but this strategy wastes a lot of resources on this passive zones that can, for some geometries, account for more than 5 times the elements in the active zone. Therefore, to extend our capabilities to non-trivial geometric domains, we store the connectivity of the mesh in a data structured that can be easily accessed and used during the assembly-free matrix-vector product. Since we are using an element-based approach, i.e. we compute the contribution of each local stiffness matrix associated with each element, our mesh data structure is also element-based, and for each element we store the nodes associated to it. For the Brick 8 element that we will use to compose our mesh, each element has 8 nodes that we need to store. However, we can reduce the number of nodes that we need to store by half (from 8 to 4) if we use a simple node index numbering rule:

- The indexes of the top nodes of an element is equal to the indexes of the bottom nodes plus one.

This rule can be visualized in Fig. 4.4. With this rule, we only need to store the bottom nodes for each element, and we can easily compute the top nodes. The indexes of the bottom 4 nodes can be stored using a `int4` CUDA data type, which groups the information of 4 regular `int` variables together for fast coalesced memory access.

Using the parallel processing power of the GPU, we can compute the contribution of several local elements to the global matrix in the assembly-free method at the same time, which greatly speeds up the computation of the matrix-vector product. However, when we compute the contribution of each element in parallel, two different processors might try to add the contribution of their respective elements to the same DOF at the same time. That will



Mesh Data Structure

Element Index	Node Index	Int4 Array Index
0	h_0 i_0 j_0 k_0	0
1	h_1 i_1 j_1 k_1	1
⋮	⋮	⋮

Figure 4.4: Node ordering and mesh data structure for efficient GPU storage and access.

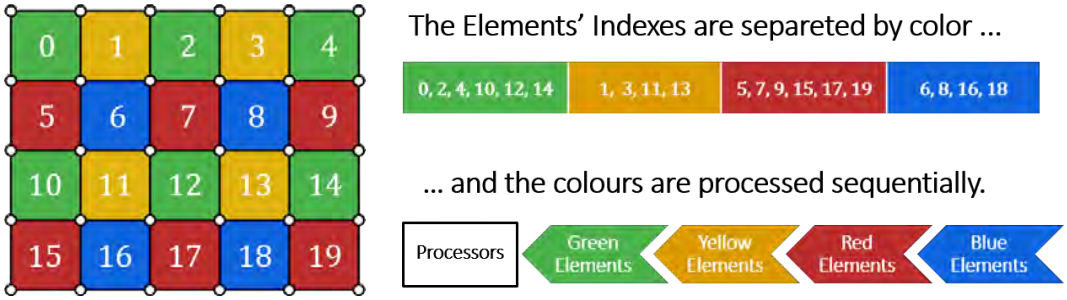


Figure 4.5: Coloring scheme for assembly-free parallel matrix-vector product.

happen when the elements being computed by two different processors share a node. When two processors try to write to the same memory space (a.k.a. the same DOF) at the same time, we have what is called a race condition [52], which can severely slow down computation, because one of the processors has to wait for the other to finish writing to that memory space so that it can then itself use that memory space as well. To avoid this race condition we use a mesh coloring scheme [39]. In this mesh coloring scheme, we color the mesh such that elements with the same color do not share a node. After the mesh is colored, we can send the elements of each color to be processed in parallel, one color at the time, and with this, we avoid the race condition, as exemplified in Fig. 4.5.

4.3.1.2 Optimized Local Stiffness Matrix Product

The essence of the assembly-free global stiffness matrix product operation is the local stiffness matrix product, which is computed in parallel. Therefore,

the efficiency of this local stiffness matrix computation is directly related to the efficiency of the assembly-free operation. Our goal in this Section is to make the local stiffness matrix product as efficient as possible, and to explain that, we are going to use a simple example. Imagine we have the following matrix-vector product that we are trying to compute:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} a & a & a & -b & -b \\ -a & -a & c & c & c \\ c & c & a & a & c \\ a & -a & c & b & b \\ -a & -a & -a & b & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \tag{4-18}$$

Notice that this matrix has only 3 distinct terms a, b and c, and we can use this to our advantage. Let us focus on the first row of the matrix, which represents the following Eq.:

$$y_1 = a\,x_1 + a\,x_2 + a\,x_3 - b\,x_4 - b\,x_5 \tag{4-19}$$

In this Eq. (4-19), we have to perform 5 multiplications, and four additions to compute the value of y_1, which sums up to a total of 9 floating-point operations (FLOPS). However, if we reorganize this Eq. (4-19) making use of the distributive property of multiplication:

$$y_1 = a\,(x_1 + x_2 + x_3) - b\,(x_4 + x_5), \tag{4-20}$$

we can reduce the number of multiplication required to compute the value of y_1 from 5 to only 2, which reduces the total number of FLOPS to 6. If we do the same for all the rows in the Eq. we get:

$$\begin{aligned} y_1 &= a\,(x_1 + x_2 + x_3) - b\,(x_4 + x_5) \\ y_2 &= -a\,(x_1 + x_2) + c\,(x_3 + x_4 + x_5) \\ y_3 &= c\,(x_1 + x_2 + x_5) + a\,(x_3 + x_4) \\ y_4 &= a\,(x_1 - x_2) + c\,x_3 + b\,(x_4 + x_5) \\ y_5 &= -a\,(x_1 + x_2 + x_3) + b\,(x_4 + x_5), \end{aligned} \tag{4-21}$$

which reduces the total number of FLOPS of this matrix-vector product from 45 to 31. So far we only reduced the number of operations row-wise, but we have several repeated terms across different rows that we can pre-compute to reduce the number FLOPS even further. Let us define the following terms:

$$
\begin{aligned}
t_1 = (x_1 + x_2 + x_3) \qquad t_2 = (x_4 + x_5) \qquad t_3 = (x_1 + x_2) \\
t_4 = (x_3 + x_4 + x_5) \qquad t_5 = (x_1 + x_2 + x_5) \\
t_6 = (x_3 + x_4) \qquad t_7 = (x_1 - x_2) \qquad t_8 = (x_3),
\end{aligned}
\tag{4-22}
$$

then we can redefine Eqs. (4-21) as:

$$
\begin{aligned}
y_1 &= a\, t_1 - b\, t_2 \\
y_2 &= -a\, t_3 + c\, t_4 \\
y_3 &= c\, t_5 + a\, t_6 \\
y_4 &= a\, t_7 + c\, t_8 + b\, t_2 \\
y_5 &= -a\, t_1 + b\, t_2
\end{aligned}
\tag{4-23}
$$

The computation of the t_* terms cost us 10 FLOPS, and to compute Eqs. (4-23) cost us an extra 17 FLOPS, which give us a total of 27 FLOPS. This simple example demonstrates the idea of how to optimize a matrix-vector product. We can go a step further and optimize the formation of the t_* terms. To see this, notice that we can write the formation of these terms as a matrix-vector product as well, i.e.:

$$
\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix} =
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
-1 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
1 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix},
\tag{4-24}
$$

and we can optimize this computation in a similar manner by defining the following terms:

$$
s_1 = x_1 + x_2 \qquad s_2 = s_1 + x_3 \qquad s_3 = x_3 + x_4 \tag{4-25}
$$

With the terms defined above in Eq. (4-25), we can compute the terms from Eq. (4-22) as:

$$
\begin{aligned}
t_1 = s_2 \qquad t_2 = x_4 + x_5 \qquad t_3 = s_1 \qquad t_4 = x_5 + s_3 \\
t_5 = x_5 + s_1 \qquad t_6 = s_3 \qquad t_7 = x_1 - x_2 \qquad t_8 = x_3
\end{aligned}
\tag{4-26}
$$

By using the terms of Eq. (4-25), to compute the terms of Eq. (4-26), we can reduce the total number of FLOPS to 25. Therefore, by simply rearranging

the operations we can reduce the matrix-vector product from 45 FLOPS to 25 FLOPS, which yields a 45% reduction in computational cost.

This 5X5 matrix example is simple enough that we can analyse it by hand, and figure out an efficient arrangement to compute the output of the matrix-vector product. However for the 24X24 local stiffness matrix, or other more complex matrix-vector product, we need a systematic procedure to find the best way to compute the matrix-vector product. In other words, we want to find a set of additions, and multiplications that minimizes the number of FLOPS necessary to compute the output of a given matrix-vector product. In this general form, we could have terms, such as $t_1 = x_1 + x_2 + x_3$, that are formed by the addition of three values. However, in terms of number of FLOPS, this is equivalent to computing $t_{0.5} = x_1 + x_2$, and $t_1 = t_{0.5} + x_3$. As a matter of fact, any term that is composed by the addition of more than two terms can be decomposed in a series of terms formed by adding two numbers. These terms that are composed by adding only two terms are, therefore, the "pairs" of our problems, which we can use to form any other term. This means that we can reduce our problem of finding a set of additions, to that of finding a set of "pairs" that minimizes the number of FLOPS.

The input of our problem is, therefore, a set of sets, that represent the matrix of the matrix-vector product, a comparison function, and a cost function. The set of sets is generated from the matrix such that, for each row, all the components of the input vector that are multiplied by the same value form a set, e.g. for the matrix in Eq. (4-18), the first row would give us the sets $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$, the second row would give us the sets $\{x_1, x_2\}$ and $\{x_3, x_4, x_5\}$, and so on, until we get the whole set of sets:

$$\mathcal{G} = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_1, x_2\}, \{x_3, x_4, x_5\}, \{x_1, x_2, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}, \{x_3\}\} \tag{4-27}$$

The comparison function takes as inputs two "pairs", and outputs a logical `True` if they are equal according to a specified rule, and `False` otherwise. In the specific problem that we are interested, i.e. minimize the number of FLOPS spent in addition, the comparison function is:

$$\mathcal{C}\left(\{x_\alpha, x_\beta\}\ \{x_\gamma, x_\theta\}\right) = \left(x_\alpha == x_\gamma \wedge x_\beta == x_\theta\right) \vee \left(x_\alpha == -x_\gamma \wedge x_\beta == -x_\theta\right) \tag{4-28}$$

in which $\wedge$ and $\vee$ are the symbols for the logical operations "`and`" and "`or`", respectively. To understand why the need for such comparison function see the two Eqs. in the following example:

$$\begin{aligned} y_1 &= x_1 + a\ (x_2 + x_3) \\ y_2 &= x_1 + a\ (-x_2 - x_3) \end{aligned} \tag{4-29}$$

The two Eqs. are different, because the second terms $(x_2 + x_3)$, and $(-x_2 - x_3)$, however, we can rewrite the second Eq. as:

$$y_2 = x_1 - a\ (x_2 + x_3)\,, \tag{4-30}$$

which preserves the number of operations (FLOPS) necessary to compute y_2, but show us that we can re-utilize the term $(x_2 + x_3)$ computed in the first equation. The cost function, L give us the cost of forming a specific "pair", and for this case $L\left(\left\{x_\alpha, x_\beta\right\}\right) = 1$, is constant for any $\left\{x_\alpha, x_\beta\right\}$, because the cost of adding two terms in the "pair" is always the same. We use definition of a cost function here, instead of simply instating a cost of 1 to each pair for the sake of generality, and the possibility to expand this scheme to other types of operation cost minimization. Once these inputs are specified, we can define a series of algorithmic operations in the input $\mathcal{G}$ presented in the sequence of instructions below:

1. Initialize a counter $p = i+1$ such that i is the largest index of x_i contained in all the sets in $\mathcal{G}$;
2. If possible, choose a "pair", $\left\{x_\alpha, x_\beta\right\}$, from a set of $\mathcal{G}$. If is not possible to choose a pair, terminate;
3. Search all the sets in $\mathcal{G}$ for the pair $\left\{x_\alpha, x_\beta\right\}$ using the comparison function. If the comparison function returns `True`, remove the "pair" $\left\{x_\alpha, x_\beta\right\}$ and replace it by the term x_p. We will call this operation *collapsing* the "pair" $\left\{x_\alpha, x_\beta\right\}$ into x_p;
4. Remove from $\mathcal{G}$, all sets with one or less elements, and increase the counter $p = p + 1$.
5. If $\mathcal{G}$ is not empty return to step 2, otherwise terminate.

These set of instructions will always terminate, because each iteration collapses at least one "pair", and the number of "pair" in the input is finite. We are going to demonstrate this sequence of instructions on the example input of Eq. (4-27), that represent the matrix-vector product of Eq. (4-18), i.e.:

Iteration 0: Initialize $p = 6$

$$\mathcal{G}^{(0)} = \{\{x_1, x_2, x_3\}\,,\ \{x_4, x_5\}\,,\ \{x_1, x_2\}\,,\ \{x_3, x_4, x_5\}\,,\ \{x_1, x_2, x_5\}\,,\ \{x_3, x_4\}\,,\ \{x_1, -x_2\}\,,\ \{x_3\}\}$$

Iteration 1: Chosen "pair" $\{x_1, x_2\}$, $p = 6$

$$\mathcal{G}^{(1)} = \{\{x_6, x_3\}, \{x_4, x_5\}, \{x_3, x_4, x_5\}, \{x_6, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}\}$$

Iteration 2: Chosen "pair" $\{x_6, x_3\}$, $p = 7$

$$\mathcal{G}^{(2)} = \{\{x_4, x_5\}, \{x_3, x_4, x_5\}, \{x_6, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}\}$$

Iteration 3: Chosen "pair" $\{x_4, x_5\}$, $p = 8$

$$\mathcal{G}^{(3)} = \{\{x_3, x_8\}, \{x_6, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}\}$$

Iteration 4: Chosen "pair" $\{x_3, x_8\}$, $p = 9$

$$\mathcal{G}^{(3)} = \{\{x_6, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}\}$$

Iteration 5: Chosen "pair" $\{x_6, x_5\}$, $p = 10$

$$\mathcal{G}^{(3)} = \{\{x_3, x_4\}, \{x_1, -x_2\}\}$$

Iteration 6: Chosen "pair" $\{x_3, x_4\}$, $p = 11$

$$\mathcal{G}^{(5)} = \{\{x_1, -x_2\}\}$$

Iteration 7: Chosen "pair" $\{x_1, -x_2\}$, $p = 12$

$$\mathcal{G}^{(7)} = \{\} \qquad \textbf{Terminate}$$

This give us the sequence of "pairs":

$$x_6 = \{x_1, x_2\}, \; x_7 = \{x_6, x_3\}, x_8 = \{x_4, x_5\}, \; x_9 = \{x_3, x_8\},$$
$$x_{10} = \{x_6, x_5\}, \; x_{11} = \{x_3, x_4\}, \; x_{12} = \{x_1, -x_2\},$$

or more compactly written, $\mathcal{P} = \{\{x_1, x_2\}, \{x_6, x_3\}, \{x_4, x_5\}, \{x_3, x_8\}, \{x_6, x_5\}, \{x_3, x_4\}, \{x_1, -x_2\}\}$, which coincides with the formation of the terms in Eqs. (4-25) and (4-26). The problem is then to find the sequence of "pairs" $\mathcal{P} = \left\{\left\{x_{\alpha_1}, x_{\beta_1}\right\}, \left\{x_{\alpha_2}, x_{\beta_2}\right\}, \cdots \left\{x_{\alpha_n}, x_{\beta_n}\right\}\right\}$, that terminates the sequence of instructions above, and minimizes the total cost $T(\mathcal{P}) = \sum_{i=1}^{n} L\left(\left\{x_{\alpha_i}, x_{\beta_i}\right\}\right)$. We will call this problem the *Set Collapsing problem*. The "pairs" of this problem's solution are the terms $t_\star$ that we need to form to minimize the number of FLOPS of the additions necessary to compute the matrix-vector product of the input.

4.3.1.3 NP-Hardness

Unfortunately, this problem is NP-Hard [48], which means that a polynomial time solution algorithm is unlikely to exist. To proof that this problem is NP-Hard, we are going to reduce the *Vertex Cover problem* [64], which is known to be NP-Hard, into the *Set Collapsing problem*, outlined above, in polynomial time. In the *Vertex Cover problem*, we have as an input a Graph $G = (V, E)$, and the problem is to find the smallest subset, $\overline{V}$, of vertex of this graph, such that for every edge $e_{ij} \in E$, we have that either the vertex v_i, or the vertex v_j (or both), are in the subset $\overline{V}$, i.e. every edge must have at least one of the vertices on its extremes in the subset $\overline{V}$. The reduction of the *Vertex Cover problem* into the *Set Collapsing problem* is outlined in the steps below:

1. As input we have a graph $G = (V, E)$, with $V = \{v_1, v_2, \cdots, v_n\}$, and $E = \left\{e_{i_1 j_1}, e_{i_2 j_2}, \cdots, e_{i_m j_m}\right\}$ in which the term e_{ij} represents an edge between vertices v_i, and v_j;

2. We perform the reduction to the *Set Collapsing problem* by creating m sets g_k, one for each edge, such that $g_k = \left\{x_0, x_{i_k}, x_{j_k}\right\}$, in which the indexes i_k, and j_k come from the edge $e_{i_k j_k}$[5]. This will be the sets of the $\mathcal{G} = \{g_1, g_2, \cdots, g_m\}$ that will serve as input to the *Set Collapsing problem*. The comparison and loss function used will be the same as define in the previous Section;

3. We solve the *Set Collapsing problem* with the input above. We now interpret the solution of the *Set Collapsing problem* $\mathcal{P} = \left\{\left\{x_{i_1}, x_{j_1}\right\}, \left\{x_{i_2}, x_{j_2}\right\}, \cdots, \left\{x_{i_p}, x_{j_p}\right\}\right\}$, to obtain a solution to the original *Vertex Cover problem*. To do this, first discard all the pairs $\left\{x_{i_K}, x_{j_K}\right\}$ in $\mathcal{P}$ for which one of the elements has its index (either i_k or j_k) greater than the number of vertices n. This will discard any pairs that has an element that was created by collapsing a pair of existing elements. From each of the remaining pairs $\left\{x_{i_K}, x_{j_K}\right\}$ choose the greatest index among the two of them, $q = \max(i_K, j_K)$, and add the corresponding vertex x_q to the covering subset $\overline{V}$. The subset $\overline{V}$ formed this way is a minimum vertex cover.

Now we will prove that this procedure give us the solution to *Vertex Cover problem*, and that the subset $\overline{V}$ formed this way is a minimum vertex

[5]Notice that the element x_0 has no vertex counterpart, and it is exclusively used in the reduction.

cover. First, see that each element of the input $\mathcal{G}$ has 3 elements, and that two different sets have, at most, two elements in common, x_0, and possibly another element x_i [6]. This means that after a set is collapsed, the remaining set has two elements, and this new set is not the same as any other set, current or possibly generated in future collapses. To see this more clearly, we can separate the collapse in four cases, and for any arbitrary two sets we have:

Case 1: $g_1 = \left\{x_0, x_{i_1}, x_{j_1}\right\}$ and $g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$, with $x_{i_1} = x_{i_2}$ (the cases in which $x_{j_1} = x_{j_2}$, or $x_{i_1} = x_{j_2}$, or $x_{j_1} = x_{i_2}$ are analogues), and we are collapsing $x_{n+1} = \{x_0, x_{i_1}\}$. From this we get the new sets:

$$g_1 = \left\{x_{n+1}, x_{j_1}\right\}, \text{ and } \; g_2 = \left\{x_{n+1}, x_{j_2}\right\}$$

Since $x_{j_1} \neq x_{j_2}$, we have that $g_1 \neq g_2$.

Case 2: $g_1 = \left\{x_0, x_{i_1}, x_{j_1}\right\}$ and $g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$, with $x_{i_1} = x_{i_2}$, and we are collapsing $x_{n+1} = \left\{x_0, x_{j_1}\right\}$. From this we get the new sets:

$$g_1 = \{x_{n+1}, x_{i_1}\}, \text{ and } \; g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$$

The two sets are clearly different because they have different number of elements, but not only that, any future collapse can only replace pairs with a new element x_{n+r} with $r > 1$, meaning that $x_{n+1} \neq x_{n+r}$. Therefore $g_1 \neq g_2$.

Case 3: $g_1 = \left\{x_0, x_{i_1}, x_{j_1}\right\}$ and $g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$, with $x_{i_1} = x_{i_2}$, and we are collapsing $x_{n+1} = \left\{x_{i_1}, x_{j_1}\right\}$. From this we get the new sets:

$$g_1 = \{x_{n+1}, x_0\}, \text{ and } \; g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$$

The collapse of the pair in the first set cannot cause a collapse in the second set because $x_{j_1} = x_{j_2}$, and for the same argument stated in the previous case regarding the formation of new element, the two sets will never be equal. Therefore $g_1 \neq g_2$.

Case 4: $g_1 = \left\{x_0, x_{i_1}, x_{j_1}\right\}$ and $g_2 = \left\{x_0, x_{i_2}, x_{j_2}\right\}$, with $x_{i_1} \neq x_{i_2}$, and we are collapsing a pair in the first set. This case is trivial, and for the same argument stated in case 2 regarding the formation of new element, the two sets will never be equal. Therefore $g_1 \neq g_2$.

[6] If two sets had all 3 of the elements they would represent the same edge counted twice.

This tell us that the solution, $\mathcal{P}$ of the *Set Collapsing problem* is composed by $\overline{u} + m$ pairs, m being the number of edges, and $\overline{u}$ being the number of elements in the vertex cover subset $\overline{V}$, i.e. s collapses that reduce the 3-element sets in $\mathcal{G}$ to m 2-element sets, and those m 2-element sets which are themselves pairs. Notice that all of these new m 2-element sets will have a new element x_{n+r} with $r > 1$ and, therefore, will not be a part of the solution to the *Vertex Cover Problem*. Imagine now that there exist a subset cover, $\hat{V}$ of size $\hat{u}$, that is smallest than the solution , $\overline{V}$ of size $\overline{u}$ ($\hat{u} < \overline{u}$), obtained through the reduction procedure outlined. From this new solution, $\hat{V}$, we can create a new solution $\hat{\mathcal{P}}$ to the *Set Collapsing problem* by considering the pairs $\{x_0, x_k\}$ for each $x_k \in \hat{V}$. These pairs would collapse all the 3-element sets in $\mathcal{G}$, because the 3-element sets are formed based on the n edges, and the subset $\hat{V}$ is a vertex cover. We then add the remaining collapsed 2-element sets to this solution $\hat{\mathcal{P}}$, which will give us a total number of pairs equal to $\hat{u}+m < \overline{u}+m$ contradicting the assumption that $\mathcal{P}$ is the solution to the *Set Collapsing problem*, proving that $\overline{V}$ is the solution to the *Vertex Cover Problem*.

4.3.1.4 Branch-and-Bound Solution

The fact that the *Set Collapsing problem* is NP-Hard tell us that a polynomial time algorithm is unlikely to exist. However, we can still solve the problem using non-polynomial time algorithms, more specifically we will use the Branch-and-Bound method (B&B) [32, 71] to obtain the optimal solution. In the B&B algorithm, we create a tree that systematically represent all the possible solutions of the problem. We then search the branches of this tree for the best solution, and while we are searching, we prune unfavorable branches, i.e. we limit our search to favorable solutions, by using appropriate upper and lower bounds. The pruning of the tree can greatly reduce the number of possible solutions that need to be evaluated, and this becomes important once we analyse the asymptotic number of possible solutions. The asymptotic number of possible solution to this problem, for an input $\mathcal{G} = \{g_1, g_2, \cdots, g_n\}$, is given by:

$$O(\mathcal{G}) = \prod_{i=1}^{n} \prod_{j=2}^{|g_i|} \frac{j!}{2!(j-2)!} = \prod_{i=1}^{n} \prod_{j=2}^{|g_i|} \frac{j(j-1)}{2} = O((|g_{max}|!(|g_{max}|-1)!)^n) \quad (4\text{-}31)$$

where $|g_i|$ indicates the number of elements of g_i, and $|g_{max}|$ is the number of elements of the largest set in $\mathcal{G}$. This indicates that the number of solutions grows incredibly fast, e.g. the asymptotic number of solutions for the example problem in Eq. (4-27) is 486, but if you duplicate the number of sets (keeping

the same number of elements for each duplicated set) it grows to 236196, if instead, we add one set with 8 elements the number of possible solutions grows to 771573600. From this, it is evident that checking every possible solution becomes impractical extremely fast, and pruning the tree plays a critical role in finding an optimal solution. Now we will describe the details of the B&B algorithm that we will use to solve this problem. The first part is to define the tree that represents all the possible solutions. To build this tree we will use a recursive formation rule:

1. Start with a root node. For the first set of the input $\mathcal{G}$ with two or more elements compute all possible pairs. Each possible pair will be a new node of the tree connected to the root, and will represent the choice to include the respective pair in the solution;

2. For each new node, collapse the pair related to that node, add that pair to the solution set $\mathcal{P}$ and remove any set in $\mathcal{G}$ with less than two elements. If $\mathcal{G}$ is empty, $\mathcal{P}$ represents a solution, otherwise, for each node, return to the first step with the new $\mathcal{G}$ and the current node as root.

The resulting tree allow us to systematically search the solution space for the optimal solution. A schematic of such tree is displayed in Fig. 4.6. Each node of the tree amounts to solving a sub-problem generated by collapsing a pair, leading to a recursive description of the problem. Searching every branch of the tree, however, would amount to a brute force approach, which, according to the previous discussion of the complexity of problem, would be impractical even for small-sized problems. The second part of the B&B algorithm, pruning, significantly reduces the number of branches that we need to check to find the optimal solution. In order to prune we need to conceive a function, $b(\mathcal{G})$, such that $b(\mathcal{G}) \leq T(\mathcal{P})$ for all possible solutions, $\mathcal{P}$, of $\mathcal{G}$, i.e. this function $b(\mathcal{G})$ is a lower bound on the objective function to all possible solutions of $\mathcal{G}$.

To use this bound to prune the tree, observe that each node i has a partial solution $\mathcal{P}_i$, obtained by systematically adding the pairs represented by the nodes that lead to the current node i, and a sub-problem $\mathcal{G}_i$ generated by collapsing all the pairs in said partial solution. By adding the cost of the partial solution of a node to the lower bound of the sub-problem associated to said node, $l_i = T(\mathcal{P}_i) + b(\mathcal{G}_i)$, we get a lower bound on the objective function of all possible solutions emanating from that node. If we compare this value to the best solution that we found thus far, $\mathcal{P}_{opt^\star}$, we can estimate how favorable a branch of tree might be, moreover, if $T(\mathcal{P}_{opt^\star}) \leq l_i$, we can guarantee that any solution emanating from that node will not be better than our current

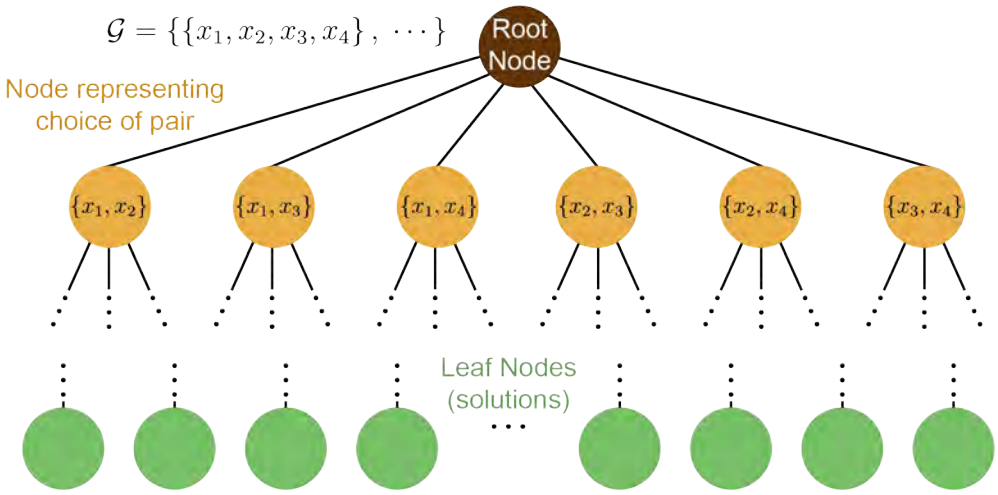


Figure 4.6: Schematic of the decision tree graph representing the solutions of the Set Collapsing Problem.

best solution. Therefore, we do not need to search any edge going out of that node, effectively pruning the tree of irrelevant branches.

This highlights the importance of choosing a good lower bound function. The tighter the bound, i.e. the closest $b(\mathcal{G})$ is to the minimal value of $T(\mathcal{P})$, the more branches we will be able to prune, reducing significantly the computational cost. The bounding function chosen for our problem is the number of unique sets in $\mathcal{G}$. To demonstrate that this is a lower bound, note that any unique set will have at least one pair that can be associated to it. More precisely, if there existed a solution with less pairs than unique sets, it would mean that a set was completely collapsed by pairs contained in other sets. In order for that to happen, either the collapsed set, g_c, must be equal to another set, which is not possible by assumption, or, g_c is contained in another set, g_i, larger than itself ($\|g_c\| < \|g_i\|$). We regard that even if g_c is the combination of two or more smaller sets, we can still associate an unmatched pair to it, because the collapsed pairs will generate new elements such that the pairing of these new elements will, eventually, not be in any smaller set. Returning to the case $g_c \subsetneq g_i$, we can associate g_c to the pair, which will completely collapse (i.e. the pair that will make g_c collapse to a single element), and associate to g_i, the pair that will completely collapse it, which will be different because $\|g_c\| < \|g_i\|$, and this concludes the proof.

To exemplify the effectiveness of this bounding function, we display, in Fig. 4.7, the solution tree of the example problem in Eq. (4-27), highlighting the checked solutions in comparison to all existing solutions.

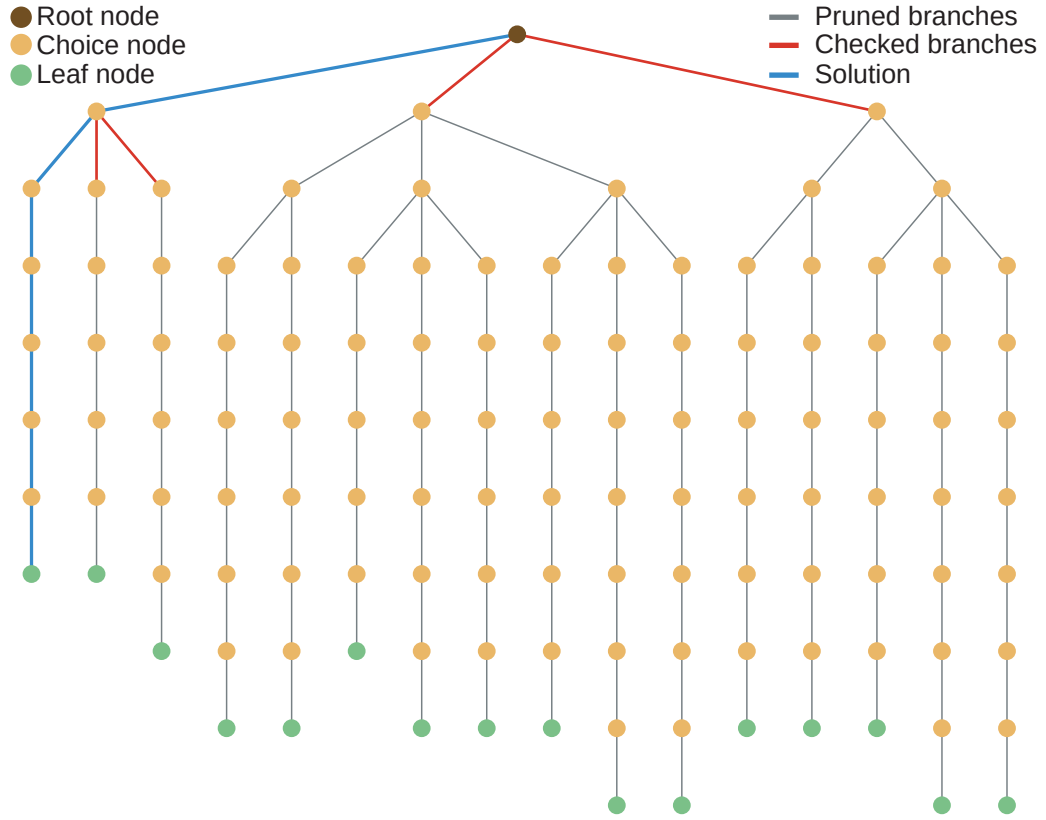


Figure 4.7: Decision tree graph representing the solutions, and the Branch-and-Bound procedure of problem in Eq. (4-27).

4.3.1.5 Optimizing FLOPS Spent in Post-Multiplication Addition

We have effectively optimized the number of FLOPS spent in pre-multiplication additions in the matrix-vector product. Similarly, we can reduce the number of FLOPS spent in the post-multiplication addition operations, and we can do this using the same framework. To do this, we simply write each multiplied $t_\star$ term as a new input to a matrix-vector product, e.g. for Eqs. (4-23), we can write:

$$\begin{aligned} m_1 &= a\, t_1 \qquad m_2 = b\, t_2 \qquad m_3 = a\, t_3 \qquad m_4 = c\, t_4 \\ m_5 &= c\, t_5 \qquad m_6 = a\, t_6 \qquad m_7 = a\, t_7 \qquad m_8 = c\, t_8, \end{aligned} \tag{4-32}$$

and then redefine the computation of the output $\mathbf{y}$ vector as the matrix-vector product:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \end{pmatrix} \tag{4-33}$$

The first thing we highlight is that we can reuse the multiplied terms m_1 and m_2, that appears in more than one row. More than that, if we perform the same optimization procedure on this matrix, we arrive at the following terms:

$$n_1 = m_1 - m_2 \qquad n_2 = -m_3 + m_4 \qquad n_3 = m_5 + m_6 \qquad n_4 = m_2 + m_7 + m_8, \tag{4-34}$$

reducing the computation of the Eqs. (4-23) to:

$$\begin{aligned} y_1 &= n_1 \\ y_2 &= n_2 \\ y_3 &= n_3 \\ y_4 &= n_4 \\ y_5 &= -n_1 \end{aligned} \tag{4-35}$$

With this step of optimizing the post-multiplication addition, we can reduce the total number of FLOPS to 20, which corresponds to a 55% reduction of the number of operations.

4.3.1.6 FLOP Optimization of the BRICK8 Element Local Stiffness Matrix

The local stiffness matrix is defined by the geometry of the elements of the mesh, and the constitutive matrix of the material considered in the optimization. In order to save memory, and increase computational efficiency we restrict ourselves to a single element geometry, a regular hexahedron (Brick8 element). The regular hexahedron element presents several advantages: the regular hexahedron tessellates the 3D space, i.e. it can fill up a volume, it is stable for linear elastic FEM, it presents a higher accuracy computation of stress over tetrahedral elements, and presents several symmetries that we can explore to reduce the cost of computing the local stiffness matrix-vector product. The formula to compute such local stiffness matrix is given by Eq. 4-36:

$$\mathbf{K}_{\text{local}} = l\frac{E}{(1+\nu)(1-2\nu)}\left(\nu\mathbb{K}_{\nu} + \mathbb{K}_{c}\right) \tag{4-36}$$

where l is the side length of the element, E is the Young's modulus, ν is the Poisson ratio, and the two matrices, $\mathbb{K}_{\nu}$ and $\mathbb{K}_{c}$, are displayed in Eqs. A-6 and A-7. Notice that the local stiffness matrix $\mathbf{K}_{\text{local}}$ has a linear dependency on E, and l, and, therefore, these two variables have no influence over the optimization of the local stiffness matrix product. If we analyse the contribution of the term $(\nu\mathbb{K}_{\nu} + \mathbb{K}_{c})$, we notice that this term only has 10 distinct absolute values, which are:

$$\begin{gathered} k_1 = 0.021 \qquad k_2 = 0.028 \qquad k_3 = 0.042 \qquad k_4 = 0.056 \\ k_5 = 0.028 - 0.083\nu \qquad k_6 = 0.021 - 0.083\nu \qquad k_7 = 0.042 - 0.17\nu \\ k_8 = 0.056 - 0.083\nu \qquad k_9 = 0.069 - 0.083\nu \qquad k_{10} = 0.22 - 0.33\nu \end{gathered} \tag{4-37}$$

If we compare the number of distinct values to the total number of components in the local stiffness matrix, $24 \cdot 24 = 576$, we realize that the number of distinct value is incredible low. This low number of distinct values is due to the symmetries of the geometry of the element Brick8, the isotropic behavior of the material, and the regularity of the linear elastic model. This possibly indicates that we can significantly reduce the number of FLOPS spent in the matrix-vector product of this matrix. The FLOP optimized solution of the BRICK8 element local stiffness matrix obtained using the B&B algorithm described in Section 4.3.1.4 is presented in Appendix A.

4.4 Numerical Results

This Section presents numerical results obtained using the techniques described in this Chapter through a C++/CUDA implementation of the AL-based stress constrained TO formulation. The implementation was run in a machine with 24 Intel Xeon CPUs, 251 GB of RAM, and a NVIDIA Titan Xp GPU with 12 GB of RAM. The problems presented here, consisting of a benchmark 3D L-Beam, a Double-Decked Bridge, and a Victoria Amazonica plant (a.k.a. Victoria-Regia) inspired domains, verify the effectiveness of the proposed techniques. The computational efficiency of the framework is discussed in the following Section.

4.4.1 L-Beam

The L-Beam problem is arguably the most traditional stress-constraint TO benchmark problem, because of the sharp corner in its domain's geometry that causes a stress concentration, which needs to be avoided in the final design. In this Section, we solve a 3D version of the L-Beam problem to verify the effectiveness of the implementation in avoiding stress concentration. The solutions presented here also provide a base for comparison for past, and future works. The numerical parameters[7] for this problem are displayed in Table 4.1. The domain and boundary conditions of the L-Beam are displayed in Fig. 4.8(a). Figures 4.8(b)-(d) present the optimized structure of the L-Beam for different mesh sizes, and we can see that all the solutions avoid the sharp corner of the L-Beam domain. Interestingly, the solution displayed in Fig. 4.8(c) presents a considerably different geometry. The difference in geometry is due to the non-convexity of the optimization problem that has several local minima, i.e. several different solutions; nonetheless , we can see that the different results are somewhat equivalent because they present similar values for the objective function, i.e. the weight.

Table 4.1: Input parameters for the L-Beam problem.

Parameter	Description	Value
E_0	Young's modulus	200 GPa
ν	Poisson's ratio	0.25
σ_{lim}	Stress limit	350 MPa
F	Applied load	700 kN
r	Filter radius	0.015

4.4.2 Double-Decked Bridge

This Section presents the numerical results for the Double-Decked Bridge problem for which the domain and boundary conditions are described in Fig. 4.9(a). The numerical parameters for this problem are displayed in Table 4.2. The Double-Decked Bridge problem was chosen for two main reasons: First, it is a fully 3D problem, meaning that it cannot be obtained by the extrusion of a 2D domain; and, second, it has several sharp corners that cause stress concentration and have to be avoided in the final design. Furthermore, the solutions of this problem (Fig. 4.9(b)-(d)) present a great amount of complexity, with several small-scale features. These detailed structures require meshes with a

[7]Material properties of general steel taken from `https://www.matweb.com/search/datasheet.aspx?bassnum=MS0001&ckck=1`.

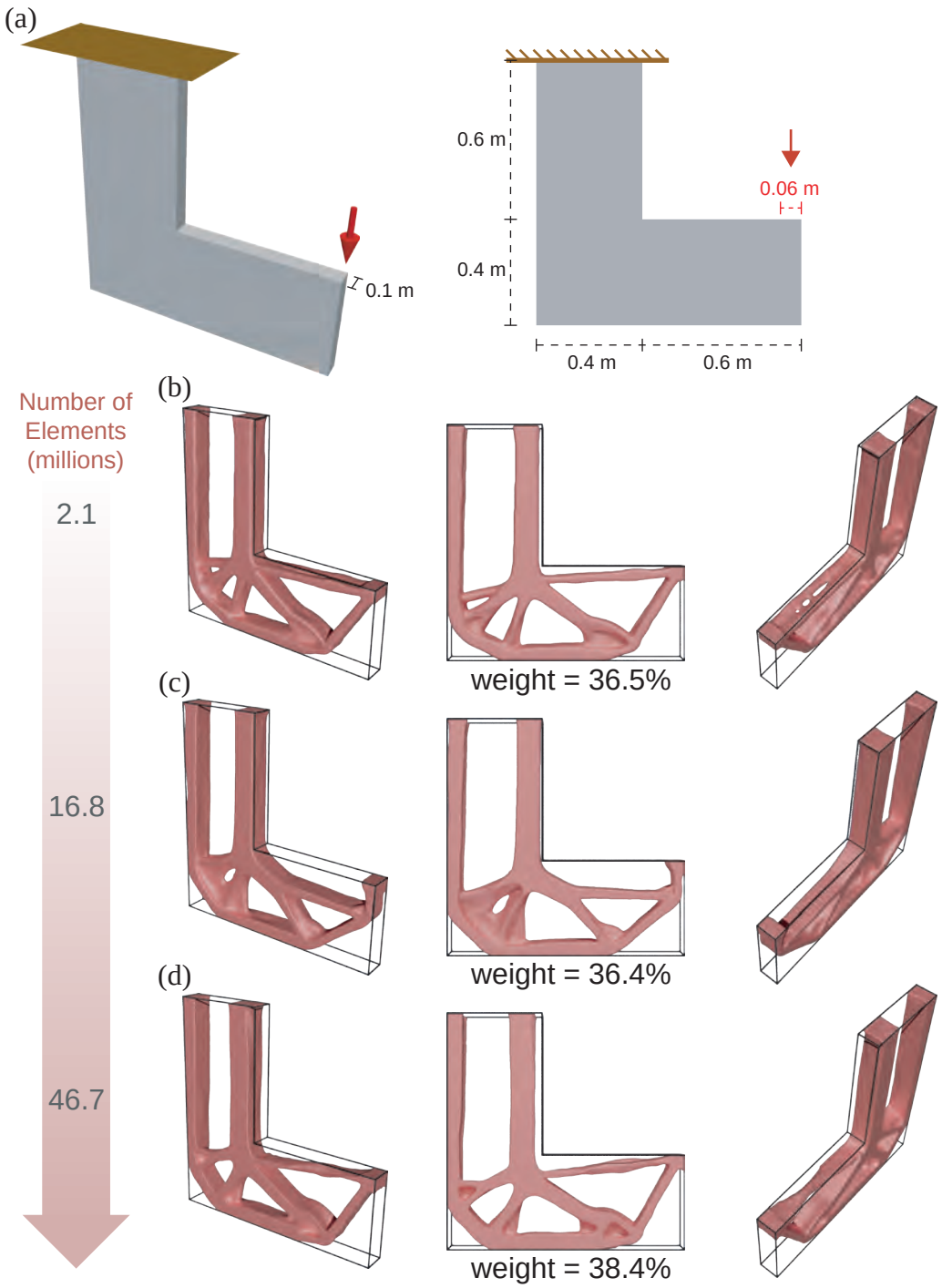


Figure 4.8: L-Beam problem and solutions; (a) Design domain geometry, and boundary conditions, with the supports being represented by the brown patches, and the loads being represented by the red arrow. (b)-(d) Optimized structures for meshes with 2097152, 16777216, and 46656000 elements, and their respective weights as a percentage of the total weight of the domain.

considerable number of elements to be precisely represented, highlighting the importance of large-scale problems. In the optimized structures of Fig. 4.9(b)-(d) we can also notice an increase in complexity, which we believe is caused by the finer meshes increased capabilities of representing fine details of the structure.

Table 4.2: Input parameters for the Double-Decked Bridge problem.

Parameter	**Description**	**Value**
E_0	Young's modulus[8]	200 GPa
ν	Poisson's ratio	0.25
σ_{lim}	Stress limit	350 MPa
F	Applied load	980 MN
r	Filter radius	0.25

4.4.3 Victoria Amazonica

The Victoria Amazonica (a.k.a. Victoria-Regia) is the largest species of water lily in the world, with leaves reaching up to 3 meters in diameter, and supporting up to 40 kilograms in its surface. Their leaves are able to withstand such tremendous weight because of their intricate structure hiding underneath the surface of the water (see Fig. 4.10(a)). Inspired by this evolutionary optimized structure, we proposed the disk like domain, and boundary conditions displayed in Fig. 4.10(b) to try to mimic the Victoria Amazonica underlying structure. Although the whole disk domain is used during the optimization, we impose symmetry in design as indicated by the 3 symmetry planes in Fig. 4.10(b). We also have one element thick solid passive zone in the top layer of the domain (where we apply the loads) to simulate the leaf itself. Loads 1 and 2 increase linearly in magnitude with the radial coordinate, while Load 3 is homogeneously distributed throughout the surface. Each load case is applied separately. Unlike the previous examples that start with homogeneous initial guess of 0.5 for the design variable, this example starts with a heterogeneous random initial guess following a uniform distribution in the range of $[0.3, 0.7]$. This random initial guess breaks the radial symmetry, which promotes the "branching" of the structure, and solutions with lower overall weight.

Figures 4.10(c), and (d) display the optimized structure for two different mesh sizes containing 1.5 million and 16.8 million elements. The optimized structure mimics the radial features of the Victoria Amazonica leaf, but do not present the thin shells that seem to run along its circumference. We theorize that this is because the main function of such circumferential shell is to trap

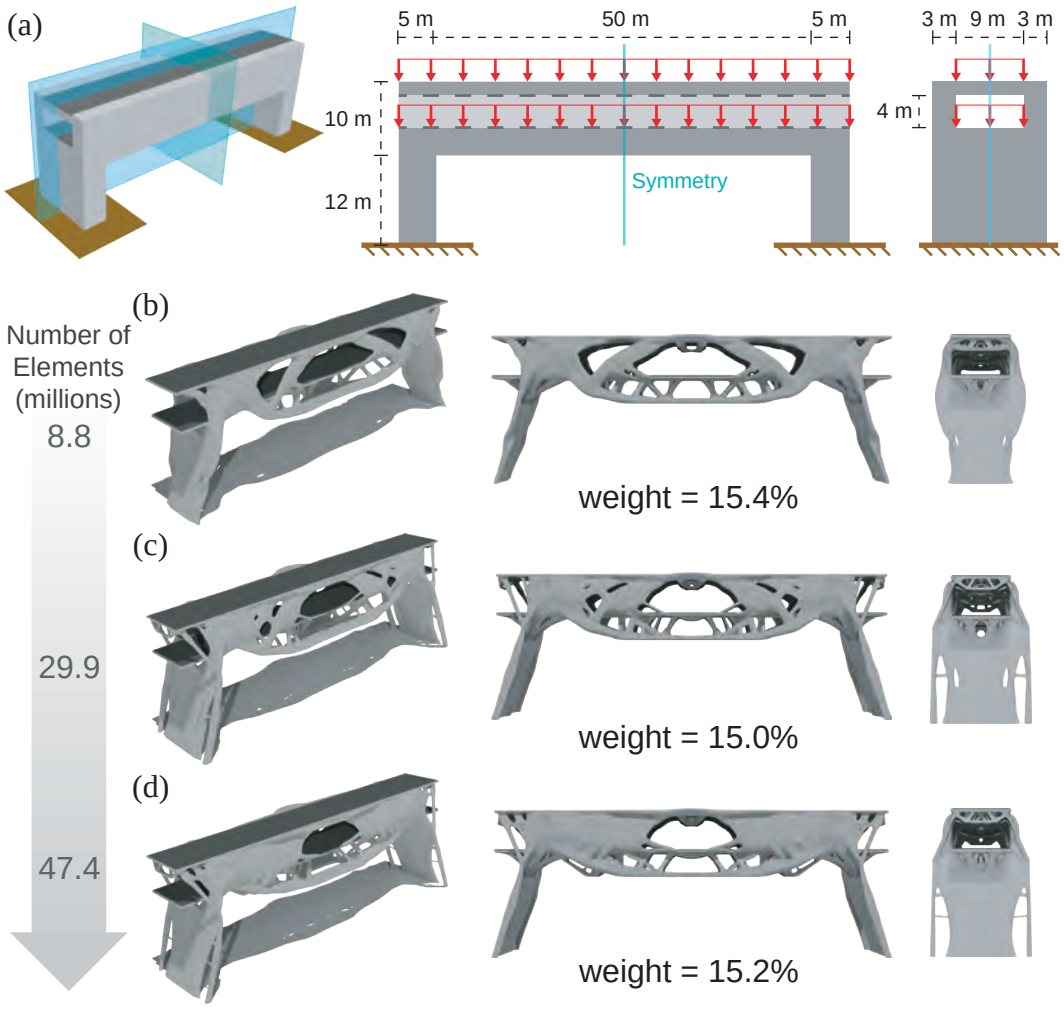


Figure 4.9: Double-Decked Bridge problem and solutions; (a) Design domain geometry, and boundary conditions, with the supports being represented by the brown patches, and the loads being represented by the red arrows. The symmetry planes of the design domain are represented in green and blue. (b)-(d) Optimized structures for meshes with 8847360, 29859840, and 47416320 elements, and their respective weights as a percentage of the total weight of the domain.

air underneath the leaf to help keep it afloat, and do not play a significant structural role. Considering this fact, the resemblance between our optimized structures and the structure of the Victoria Amazonica leafs show us once again the power that evolution plays in the optimization of living organisms, and demonstrates the capabilities of our framework to obtain optimal structures.

Table 4.3: Input parameters for the Victoria Amazonica problem.

Parameter	Description	Value
E_0	Young's modulus	1 Pa
ν	Poisson's ratio	0.25
σ_{lim}	Stress limit	0.75 Pa
F_1	Load 1	100 N
F_2	Load 2	100 N
F_3	Load 3	100 N
r	Filter radius	0.20

4.5 Computational efficiency

In this Section we will evaluate the computational efficiency of the proposed GPU framework, and compare it with other freely-available GPU libraries. All the tests were performed using a machine with 24 Intel Xeon CPUs, 251 GB of RAM, and a NVIDIA Titan Xp GPU with 12 GB of RAM[9]. The computational times displayed are an average of 1000 executions in order to obtain an accurate measure of performance.

We will start by evaluating the efficiency of the implemented matrix-vector product, because this the most essential part of the framework, and the operation with the greatest contribution of this work. The matrix-vector product efficiency is measured using several matrices generated from the FEM ranging from 810,000 to 154,436,544 DOFs (rows/columns). The results are displayed in Fig. 4.11(a) where we compare 4 different frameworks:

1. The proposed EbE optimized local matrix product;
2. The traditional EbE approach using the full local matrix product;
3. An implementation using the cuSPARSE library [2][10];
4. An implementation using the Matlab GPU library.

[9]The computational time is highly dependent on the hardware used to run the implementation, but the same trend is expected for reasonably similar GPU architectures.

[10]cuSPARSE is a highly efficient CUDA library that provides GPU-accelerated linear algebra routines for general sparse matrices

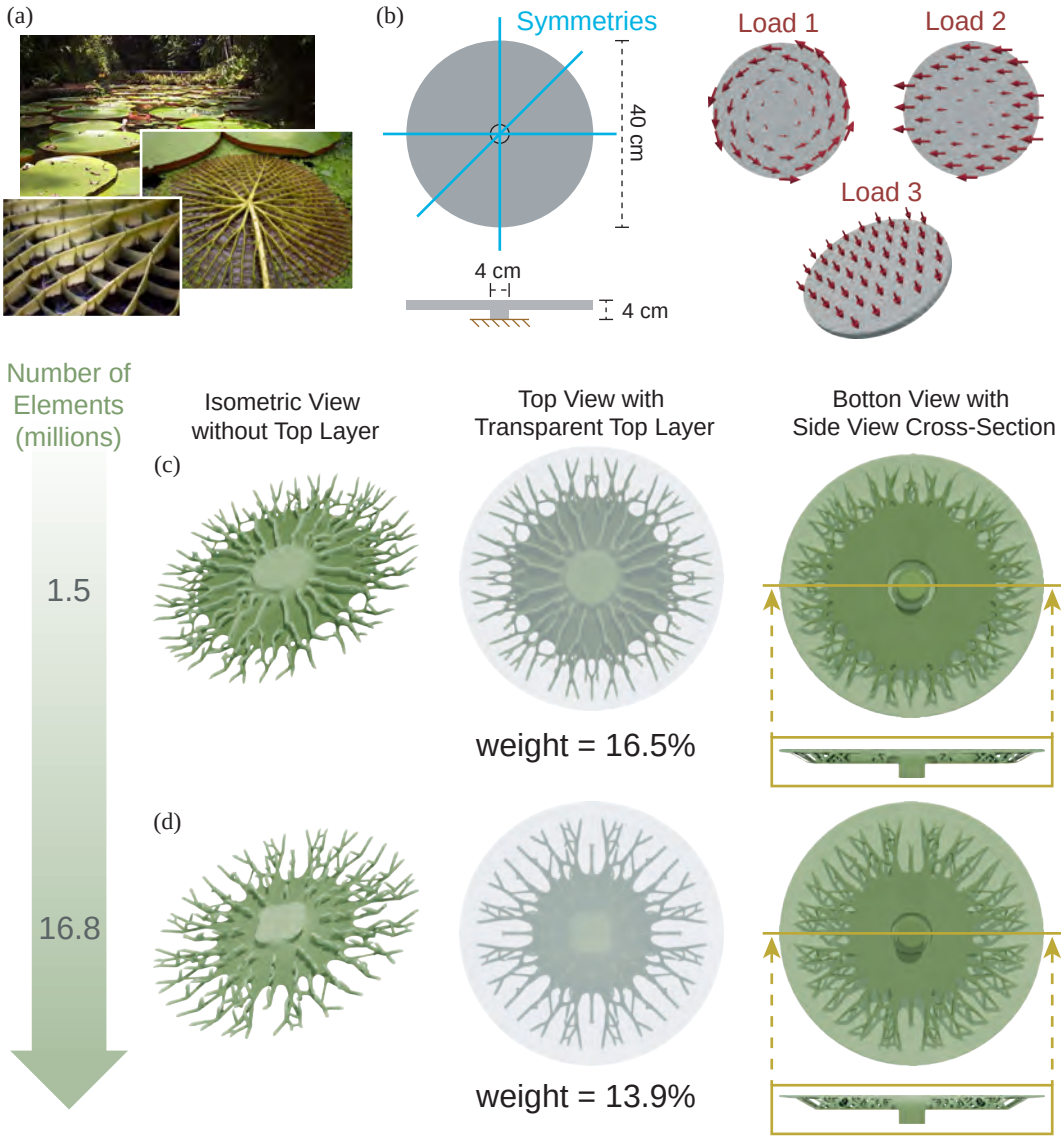


Figure 4.10: Victoria Amazonica problem and solutions; (a) Specimen of the plant Victoria Amazonica in nature that inspired the problem definition, and the complex underlying structure of their leaves ; (b) Design domain geometry, and boundary conditions, with the supports being represented by the brown patches, and the loads being represented by the red arrows. The symmetry planes of the design domain are represented in blue. (c)-(d) Optimized structures for meshes with respective weights as a percentage of the total weight of the domain.

In this results, we can see that, not only the proposed framework **1** provides a considerable speedup when compared to frameworks **2** (X1.6 speed up), **3** (X2.8 speed up), and **4** (X3.3 speedup), but also the EbE approach allow us to solve problems almost 20 times bigger than the cuSPARSE implementation, and almost 50 times bigger than the Matlab implementation. As expected, due to the sparsity pattern of the matrices, we see a linear relation between the number of DOFs and the computational time.

Next, we compare the efficiency of the PCG in each framework (except for framework **2** [11]), and the results are displayed in Fig. 4.11(b). In this results, we can see that the speedup of the proposed framework is even greater, reaching a X3.8 speedup compared to **3**, and a X25.3 speedup compared to **4**.

Finally, we evaluate the performance of the TO procedure using the proposed EbE optimized local matrix product (**1**) for the L-Beam, and the Double-Decked Bridge[12]. The results are displayed in Fig. 4.12(a) in which we can see a non-linear increase in computational time with the number of elements in the mesh. This non-linear increase is due to the higher number of PCG iterations required for convergence of the linear systems solutions. Furthermore, we also see the problem's influence, i.e. L-Beam vs Double-Decked Bridge, on the computational time, because of the geometry of the domains that affects the condition number of the linear systems. Figure 4.12(b) also display a breakdown of the computational time of the TO showing that more than 99% of the computational time is spent on the solution of the linear system. In addition, Figure 4.12(c) displays the breakdown of the PCG algorithm per operation demonstrating that the matrix-vector multiplication accounts for 70% of the total time, which emphasizes the importance of the matrix-vector product performance.

[11] We do not evaluate the efficiency of the PCG using framework **2** because the only difference between framework **1** and **2** is the EbE matrix-vector product.

[12] We do not display the data from the Victoria Amazonica example because it has 3 load cases while the other examples have only one load case, and that would skew the resulting computational time.

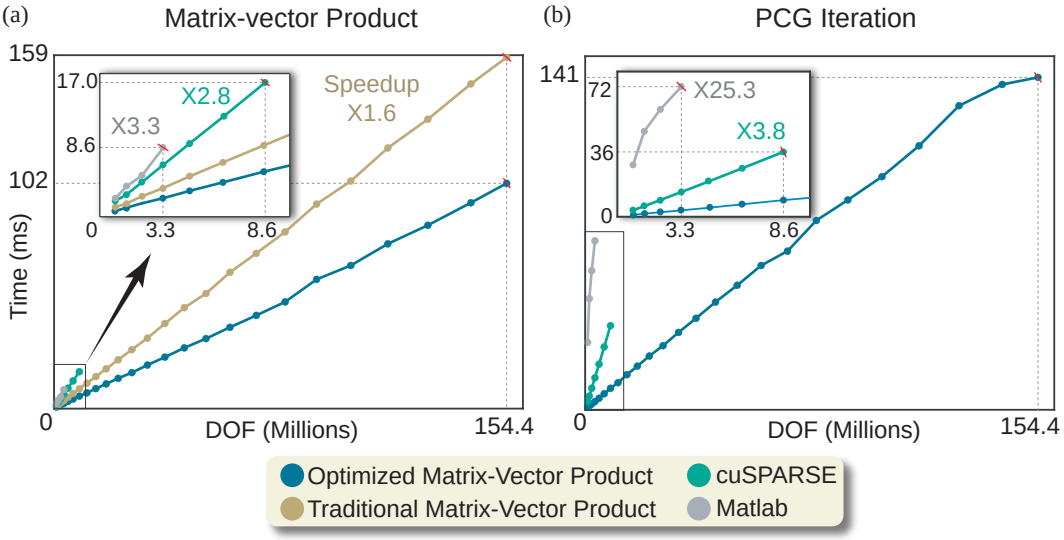


Figure 4.11: Computational efficiency of (a) a matrix-vector operation, and (b) a PCG iteration using the proposed EbE optimized local matrix product compared with a traditional EbE implementation, a cuSPARSE-based implementation, and a Matlab implementation for varying number of DOFs. Each computational time is the average of 1000 executions. The red strike marks the largest number of DOFs that we were able to compute with each approach. We also display the average speedup of the proposed EbE optimized local matrix product in relation to the other approaches.

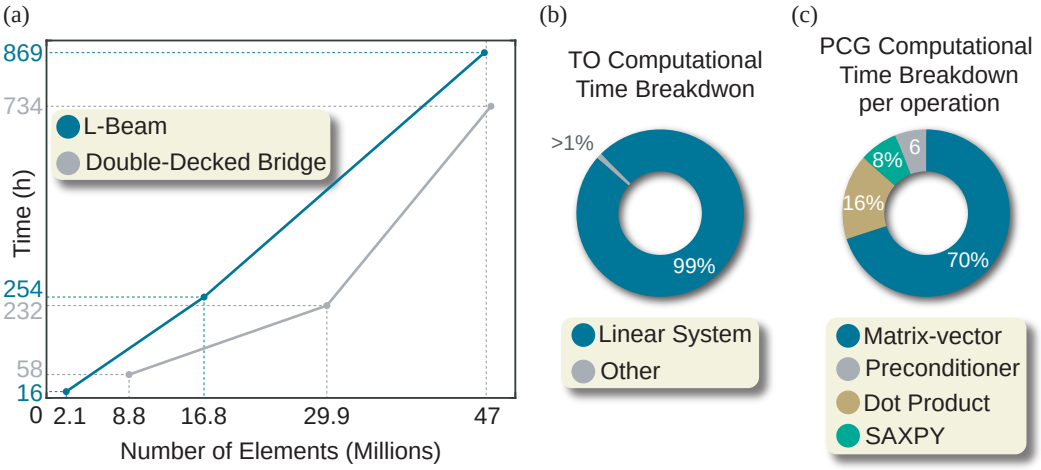


Figure 4.12: Efficiency of the TO procedure; (a) Computational time of the stress constrained TO procedure as a function of the number of elements for the L-Beam and Double-Decked Bridge problem; (b) Computational time breakdown of the TO procedure showing that the linear system accounts for more than 99% of the total computational time; (c) Computational time breakdown of the PCG algorithm per operation.

5
Continuously Varying Load Case

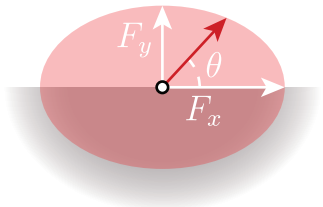
In this chapter, we focus on loads that vary in direction and magnitude, while retaining a fixed location of application. Figure 5.1 exemplifies all the load varying cases contemplated in this work, i.e. loads varying in direction and magnitude (Fig. 5.1(a)), loads varying only in direction (Fig. 5.1(b)), loads varying in a limited range of directions (Fig. 5.1(c)), loads varying in direction combined with fixed loads (Fig. 5.1(d)), two or more loads varying independently (Fig. 5.1(e)), and loads varying in 3D (Fig. 5.1(f)). We formulate the stress constrained topology optimization problem such that the local stress constraints account for the load variability in a worst-case oriented approach. That is, the stress constraints consider the maximum stress generated by the loads in a set of possible load directions. We derive analytic expressions that represent the maximum stress in this set of load directions. The formulation is tailored for linear elasticity (state equation) and we adopt a von Mises stress measure.

To handle the large number of constraints inherent to the local stress constraints problem we use the AL-based approach described in Chapter 3. We highlight that although this work focuses on stress constraints, the analytic expressions derived here can be applied to any topology optimization problem with bilinear functions and linear state equations (see Appendix B for derivations related to compliance minimization).

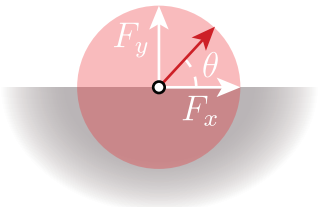
5.1
Multiple load direction

In this work, the direction of the loads applied to the TO domain are controlled by a variable $\boldsymbol{\theta} \in \Gamma$, that represents the angle of the load application, and by the set Γ that represents the set of all possible angles. Each load direction, a.k.a. load angle $\boldsymbol{\theta} \in \Gamma$, induces a distinct stress state in the optimized structure. To prevent structural failure, we have to consider the maximum stress induced by all possible load angles, $\boldsymbol{\theta} \in \Gamma$, i.e. the critical stress:

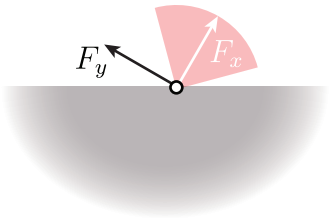
$$\widetilde{\sigma}_j^v = \sup \left\{ \sigma_j^v(\mathbf{z}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Gamma \right\} \tag{5-1}$$



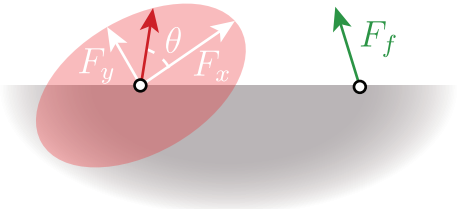
(a) General Case
$\|F_x\| \neq \|F_y\|$



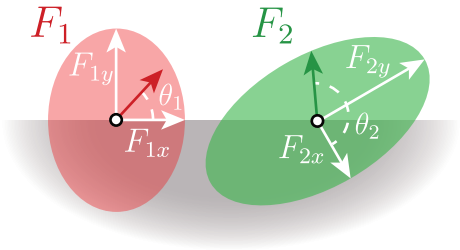
(b) Particular Case
$\|F_x\| = \|F_y\|$



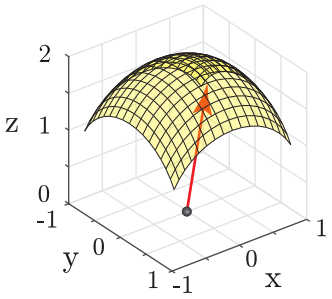
(c) Load varying in a limited range



(d) Varying Load plus a fixed Load



(e) Two loads varying independently



(f) Load varying in 3D

Figure 5.1: Schematic of all the load conditions, with loads varying in direction, and magnitude, contemplated in the proposed formulation. (a) Load varying 360^o degrees forming a ellipsoid domain in which the load varies, not only in direction, but also in magnitude. (b) Load varying 360^o degrees forming a circular domain, in which the load varies only in direction. (c) Load varying in a limited range of admissible directions. (d) Load varying 360^o degrees combined with a fixed load. (e) Two loads varying independently in direction. (f) Load varying in 3D directions.

where $\sigma_j^v(\mathbf{z}, \boldsymbol{\theta})$ is the von Mises stress at the centroid of the element j of the finite element mesh, which depends on the design variables, $\mathbf{z}$, i.e. the structure, and the load direction, $\boldsymbol{\theta}$. We can cast the problem of finding the critical stress generated by $\boldsymbol{\theta}$ as an optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\theta}\in\Gamma} \quad & \sigma_j^v(\mathbf{z}, \boldsymbol{\theta}) \\ \text{with:} \quad & \mathbf{K}(\mathbf{z})\mathbf{U} = \mathbf{F}(\boldsymbol{\theta}) \end{aligned} \tag{5-2}$$

In other words, we are interested in the critical angles, $\boldsymbol{\theta}_j^{cr}$, that cause the highest stress in the structure. Notice that, each stress evaluation point j, and, therefore, each constraint will have a particular critical angle. Using the critical angles, $\theta_j^c r$, $j = 1, ..., N_c$, we can analytically derive a worst-case stress constraint for each constraint, g_j, $j = 1, ..., Nc$, in the TO problem. In the next sections, we derive solutions for the problem in Eq. (5-2) considering different sets of admissible load angles, that is, different domains, Γ.

5.1.1 Case 1: Planar load varying 360^o degrees

We begin by deriving the solution for the problem in Eq. (5-2) considering a single force with direction that can vary 360^o [1], i.e., $\theta \in \Gamma = [-\pi, \pi]$. We start by writing the load as the sum of two vectors weighted by cosine and sine functions:

$$\mathbf{F}(\theta) = \mathbf{F}_x \cos(\theta) + \mathbf{F}_y \sin(\theta) \tag{5-3}$$

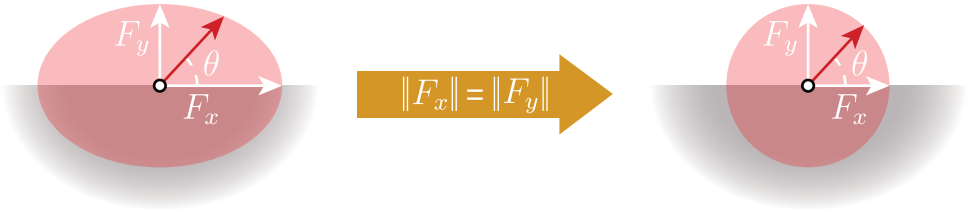
where the load basis vectors, $\mathbf{F}_x$ and $\mathbf{F}_y$, are two linearly independent vectors that compose the space of admissible loads. We highlight that $\mathbf{F}_x$ and $\mathbf{F}_y$ do not need to be aligned with the x-axis or the y-axis, or to have the same magnitude. The load basis vectors just need to be able to span the space of possible load cases. Figure 5.2 displays a schematic of the domain of possible loads, and the load basis vectors.

Due to linearity of the underlying physics (linear elasticity), we can compute the solution to the state equations using the load basis vectors as:

$$\mathbf{U} = \mathbf{K}^{-1}\mathbf{F}(\theta) = \left(\mathbf{K}^{-1}\mathbf{F}_x\right)\cos(\theta) + \left(\mathbf{K}^{-1}\mathbf{F}_y\right)\sin(\theta) \tag{5-4}$$

By defining $\mathbf{U}_x = \mathbf{K}^-1\mathbf{F}_x$ and $\mathbf{U}_y = \mathbf{K}^-1\mathbf{F}_y$, we can compute the stress as:

[1] We choose the domain as $\Gamma = [-\pi, \pi]$ instead of the more obvious $\Gamma = [0, 2\pi]$ because of the definition of the inverse tangent used in this paper, which has an image in the range $\Gamma = [-\pi, \pi]$.



(a) General Case $\|F_x\| \neq \|F_y\|$ (b) Particular Case $\|F_x\| = \|F_y\|$

Figure 5.2: Schematic of the domain of possible load cases in red and the load vector basis in white for case 1, with (a) general load basis vector, i.e. $\|F_x\| \neq \left\|F_y\right\|$ forming a elliptic domain in which the load varies, not only in direction, but also in magnitude, and (b) load basis vectors with the same magnitude, i.e. $\|F_x\| = \left\|F_y\right\|$ forming a circular domain, in which the load varies only in direction.

$$\begin{aligned} \boldsymbol{\sigma} = \mathbf{DBU} = \mathbf{DB}\left(\mathbf{U}_x \cos(\theta) + \mathbf{U}_y \sin(\theta)\right) \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}_x \cos(\theta) + \boldsymbol{\sigma}_y \sin(\theta) \end{aligned} \tag{5-5}$$

where $\mathbf{D}$ and $\mathbf{B}$ are the constitutive and strain-displacement matrices, respectively, in the finite element setting. We defined the stress components as $\boldsymbol{\sigma}_x = \mathbf{DBU}_x$ and $\boldsymbol{\sigma}_y = \mathbf{DBU}_y$, which allow us to compute the von Mises stress as:

$$\begin{aligned} \sigma^v = \left\{\boldsymbol{\sigma}^T \mathbf{V} \boldsymbol{\sigma}\right\}^{1/2} = \left\{\left[\boldsymbol{\sigma}_x \cos(\theta) + \boldsymbol{\sigma}_y \sin(\theta)\right]^T \mathbf{V} \left[\boldsymbol{\sigma}_x \cos(\theta) + \boldsymbol{\sigma}_y \sin(\theta)\right]\right\}^{1/2} \\ \sigma^v = \left\{t_{xx}\ \cos^2(\theta) + t_{yy}\ \sin^2(\theta) + 2\, t_{xy}\ \cos(\theta)\sin(\theta)\right\}^{1/2} \end{aligned} \tag{5-6}$$

where $\mathbf{V}$ is the von Mises matrix, and we defined the quadratic stress terms $t_{xx} = \boldsymbol{\sigma}_x^T \mathbf{V} \boldsymbol{\sigma}_x$, $t_{yy} = \boldsymbol{\sigma}_y^T \mathbf{V} \boldsymbol{\sigma}_y$, and $t_{xy} = \boldsymbol{\sigma}_x^T \mathbf{V} \boldsymbol{\sigma}_y$ to simplify the expression. Notice that t_{xx}, t_{yy} and t_{xy} do not depend on θ. We simplify Eq. (5-6) using trigonometric identities:

$$\sigma^v = \left\{t_{xy} \sin(2\,\theta) + 0.5\left[\left(t_{xx} - t_{yy}\right)\cos(2\,\theta) + t_{xx} + t_{yy}\right]\right\}^{1/2} \tag{5-7}$$

We change the definition of the objective function in Eq.(5-2) to the squared von Mises stress. Squaring the von Mises stress simplifies further computations, and, because the von Mises stress is non-negative, squaring it will not change the solution to the maximization problem. Therefore, the optimization problem for the critical load angle can then be written as:

$$\begin{aligned} &\max_{\theta\in\Gamma} \quad \left[\sigma^v(\mathbf{z},\theta)\right]^2 \\ &\text{with:} \quad \mathbf{K}(\mathbf{z})\mathbf{U}_x = \mathbf{F}_x, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_y = \mathbf{F}_y \end{aligned} \tag{5-8}$$

To find the analytic solution to the optimization problem in Eq. (5-8) we differentiate the objective function with respect to θ, and set it equal to zero to find the critical points, i.e.:

$$\frac{\partial\,(\sigma^v)^2}{\partial\theta} = 2t_{xy}\,\cos(2\,\theta) - \Big(\,t_{xx} - \,t_{yy}\Big)\sin(2\,\theta) = 0 \tag{5-9}$$

Eq. (5-9) solutions are of the form:

$$\theta^{cr} = \begin{cases} \frac{1}{2}\tan^{-1}\Big(\,2\,t_{xy},\;t_{xx} - t_{yy}\,\Big) + k\,\pi \qquad \text{for } k\in\mathbb{Z} \\ \\ \frac{1}{2}\tan^{-1}\Big(\,-2\,t_{xy},\;t_{yy} - t_{xx}\,\Big) + k\,\pi \qquad \text{for } k\in\mathbb{Z} \end{cases} \tag{5-10}$$

where $\tan^{-1}(\,\cdot,\,\cdot\,)$ is the two-value-argument inverse tangent that considers the appropriate quadrant in the computation of the inverse of the tangent. We denote the first and second set of solutions, in Eq. (5-10), as θ^{cr}_{max} and θ^{cr}_{min}, respectively. Notice that all elements of the set θ^{cr}_{max} achieves the same value for the objective function, independently of the value of k, and the same is true for θ^{cr}_{min}. Therefore, from now on, we will assume $k = 0$, meaning that the solution will always lie in the interval $[-\pi,\;\pi]$. To classify these two critical points, θ^{cr}_{max} and θ^{cr}_{min}, we will use the second derivative:

$$\frac{\partial^2\,(\sigma^v)^2}{\partial\theta^2} = -4\;t_{xy}\,\sin(2\;\theta) - 2\;(t_{xx} - t_{yy})\;\cos(2\;\theta) \tag{5-11}$$

We can use the trigonometric identities:

$$\begin{aligned} \sin(\tan^{-1}(a,b)) &= \frac{a}{\{a^2+b^2\}^{1/2}} \\ \cos(\tan^{-1}(a,b)) &= \frac{b}{\{a^2+b^2\}^{1/2}}, \end{aligned} \tag{5-12}$$

to obtain the value of the second derivative at θ^{cr}_{max} and θ^{cr}_{min}:

$$\left.\frac{\partial^2\,(\sigma^v)^2}{\partial\theta^2}\right|_{\theta=\theta^{cr}_{max}} = \frac{-8\;t_{xy}^2 - 2\;(t_{xx}-t_{yy})^2}{\Big\{4t_{xy}^2 + (t_{xx}-t_{yy})^2\Big\}^{1/2}} \tag{5-13}$$

$$\left.\frac{\partial^2\,(\sigma^v)^2}{\partial\theta^2}\right|_{\theta=\theta^{cr}_{min}} = \frac{8\;t_{xy}^2 + 2\;(t_{xx}-t_{yy})^2}{\Big\{4t_{xy}^2 + (t_{xx}-t_{yy})^2\Big\}^{1/2}} \tag{5-14}$$

Notice that the second derivative in θ^{cr}_{max} is always non-positive, while the second derivative in θ^{cr}_{min} is always non-negative, meaning that θ^{cr}_{max} is a local maximum, and θ^{cr}_{min} is a local minimum. Furthermore, the objective function is periodic, with the period equal to half of the domain of the optimization problem, and the objective function is smooth, which means that the global maximum, and global minimum are necessarily critical points. Since θ^{cr}_{max} and θ^{cr}_{min} are the only two critical points in the optimization domain, we have that θ^{cr}_{max} is a global maximum, and θ^{cr}_{min} is a global minimum. Therefore, the solution to optimization problem in Eq. (5-8) is:

$$\theta^{\star} = \theta^{cr}_{max} = \frac{1}{2}\tan^{-1}\Big(\, 2\,t_{xy},\; t_{xx} - t_{yy}\,\Big) \tag{5-15}$$

We substitute the expression of $\theta^{\star}$ into Eq. (5-7) of the von Mises stress to obtain the expression for the critical stress:

$$\widetilde{\sigma}^{v} = \left\{ t_{xy}\sin\left(2\,\theta^{\star}\right) + 0.5\left[\Big(\, t_{xx} - \, t_{yy}\Big)\cos\left(2\,\theta^{\star}\right) + \, t_{xx} + \, t_{yy}\right]\right\}^{1/2} \tag{5-16}$$

This critical stress $\widetilde{\sigma}^{v}$ is used in the stress constraints in Eq. (3-7) to solve the topology optimization problem. This critical stress constraint guarantees that the von Mises stress will be below the stress limit for any load angle in the domain Γ in the optimized structure.

5.1.2 Case 2: Planar load with limited angle

In this section, we focus on the case of a planar load varying in direction within a limited range of angles, $\theta \in \Gamma = \Big[\theta_{low}, \theta_{up}\Big]$, contained inside $[-\pi, \pi]$. This case is of practical interest because the range $[-\pi, \pi]$ might be too conservative for some applications, i.e. it might consider load directions that do not occur in practice leading to an over-design of the structure making it unnecessarily heavier. Therefore, the introduction of an adjustable range of angles for the load allows for more flexibility in the framework.

First, we will consider the special instance where the angle range is centered at $\theta = 0$ described as $[-\theta_r, \theta_r]$, and displayed in Fig. 5.3. Then we will show how to generalize this special instance of angle range to any interval of the form $\Big[\theta_{low}, \theta_{up}\Big]$. To consider this special instance of angle range, we adapt the optimization problem in Eq. (5-8) by including a constraint in the value of θ:

$$\begin{aligned} \max_{\boldsymbol{\theta}\in\Gamma} \quad & \sigma^v(\mathbf{z},\theta) = \left\{ t_{xy}\sin(2\,\theta) + 0.5\left[\left(t_{xx} - t_{yy}\right)\cos(2\,\theta) + t_{xx} + t_{yy}\right]\right\}^{1/2} \\ \text{s.t.:} \quad & \theta^2 - \theta_r^2 \leq 0 \\ \text{with:} \quad & \mathbf{K}(\mathbf{z})\mathbf{U}_x = \mathbf{F}_x, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_y = \mathbf{F}_y \end{aligned} \tag{5-17}$$

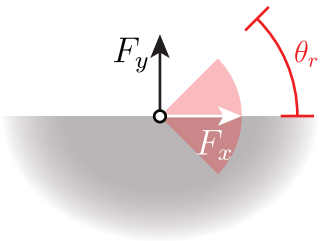


Figure 5.3: Schematic of the domain of possible load cases in red and the load vector basis in white (F_x) and black (F_y), with a limited range of angles (θ_r), for case 2.

We can divide the solution of this problem into two instances, either the critical point θ^{cr}_{max} lies in the interval $[-\theta_r, \theta_r]$, or it lies outside the interval $[-\theta_r, \theta_r]$. If the critical point θ^{cr}_{max} lies in $[-\theta_r, \theta_r]$, then θ^{cr}_{max}, as defined in Eq. (5-15), is the optimum point of this problem.

However, if $\theta^{cr}_{max} \notin [-\theta_r, \theta_r]$, the optimum point is either $-\theta_r$ or θ_r, because we proved that the only other form of critical point of this problem is a global minimum. To determine which of $-\theta_r$ or θ_r is the optimum of the problem, first notice that the objective function is periodic, with a period of π. Therefore, we only have to focus in intervals smaller than $[-\pi/2, \pi/2]$; any larger interval will necessarily contain θ^{cr}_{max} because of the periodicity of the objective function. We can further sub-divide the instance of $\theta^{cr}_{max} \notin [-\theta_r, \theta_r]$ into the cases in which $[-\theta_r, \theta_r]$ lies between θ^{cr}_{min} and θ^{cr}_{max} and the case in which θ^{cr}_{min} lies in the interval $[-\pi/2, \pi/2]$.

In the case $[-\theta_r, \theta_r]$ lies between θ^{cr}_{min} and θ^{cr}_{max}, we can use the fact that the objective function is monotonic in the interval between θ^{cr}_{min} and θ^{cr}_{max} to conclude that the maximum is the point, $-\theta_r$ or θ_r, that is closest to θ^{cr}_{max}. In the case θ^{cr}_{min} is in the interval $[-\theta_r, \theta_r]$, we can use the fact that the objective function is symmetric with respect to θ^{cr}_{min} to conclude that the farthest point from θ^{cr}_{min} inside $[-\theta_r, \theta_r]$ is the maximum, which, consequently, is also the point that is closest to θ^{cr}_{max}. For a clear visualization of the proof outlined above see Fig. 5.4. We can express all of this cases in a simple expression for the solution to the optimization problem in Eq. (5-17) as:

$$\theta^{\star} = \min\left\{ \max\left[\frac{1}{2}\tan^{-1}\left(2\,t_{xy},\ t_{xx} - t_{yy} \right),\ -\theta_r \right],\ \theta_r \right\} \tag{5-18}$$

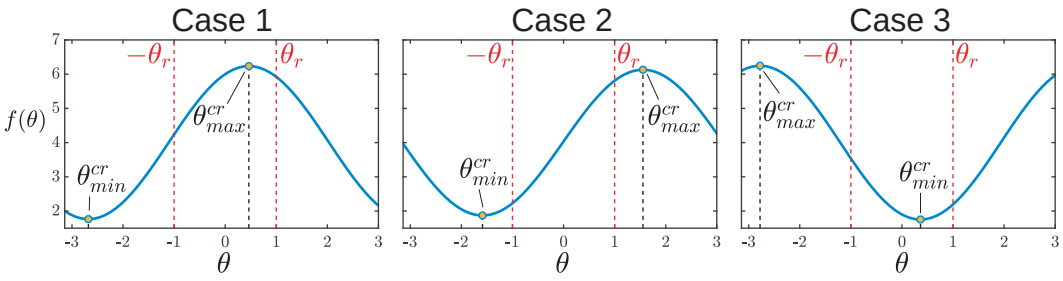


Figure 5.4: Representation of the three cases of the limited angle optimization problem. Case 1, in which the maximum lies between the limited angle range ($[-\theta_r, \theta_r]$). Case 2, in which the maximum and the minimum lie outside the limited angle range. Case 3, in which the maximum lies outside and the minimum lies inside the limited angle range.

To generalize this solution for any value of $\left[\theta_{low}, \theta_{up}\right]$, we take advantage of the freedom that we have to choose the basis vectors F_x and F_y (for more details see section 5.2). We use this freedom to rotate F_x and F_y, by the angle $\dfrac{\theta_{up} + \theta_{low}}{2}$, so that F_x lies in the middle of the interval $\left[\theta_{low}, \theta_{up}\right]$, and we set $\theta_r = \dfrac{\theta_{up} - \theta_{low}}{2}$, so that the interval $[-\theta_r, \theta_r]$ matches the original interval $\left[\theta_{low}, \theta_{up}\right]$, as illustrated in Fig. 5.5. With this change of basis, the problem of finding $\tilde{\sigma}^v$ in the interval $\left[\theta_{low}, \theta_{up}\right]$ is the same as finding $\tilde{\sigma}^v$ in the interval $[-\theta_r, \theta_r]$ with the rotated F_x and F_y.

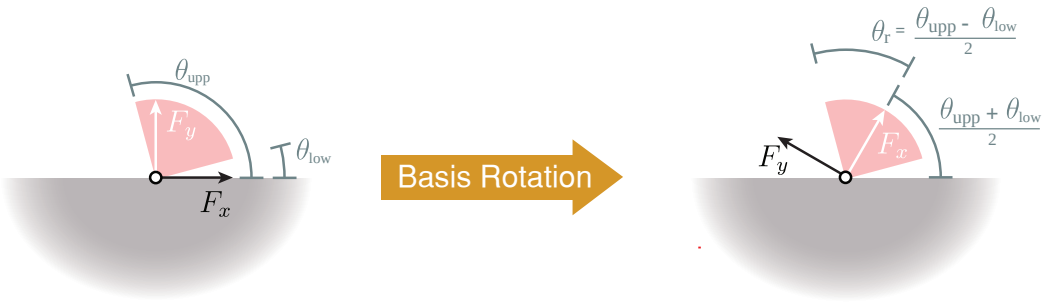


Figure 5.5: Schematic displaying how to rotate the basis vectors (F_x and F_y) to achieve any continuous range of admissible angles desired.

5.1.2.1
Secondary Range of Admissible Angles

If we obtain a solution that satisfy the stress constraints for a range of $[-\theta_r, \theta_r]$, then the solution will also satisfy the stress criteria for $\theta' = \theta + \pi$, in which $\theta \in [-\theta_r, \theta_r]$. This secondary range of admissible load angles occurs because the state equations are linear and the von Mises stress equation is the square root of a bilinear function. This idea comes naturally if one realizes that inverting the direction of the loads generates the same von Mises stress distribution. To see this clearly, notice that, if we replace θ by $\theta' + \pi$ in the objective function of Eq. (5-17), we obtain:

$$\sigma^v(\mathbf{z}, \theta') = \left\{ t_{xy} \sin(2\,\theta + 2\pi) + 0.5 \left[\left(t_{xx} - t_{yy} \right) \cos(2\,\theta + 2\pi) + t_{xx} + t_{yy} \right] \right\}^{1/2}, \quad (5\text{-}19)$$

but $\sin(2\,\theta + 2\pi) = \sin(2\,\theta)$ and $\cos(2\,\theta + 2\pi) = \cos(2\theta)$, which means that the expression in Eq. (5-19) is numerically equivalent to the expression in Eq. (5-17). This secondary range of admissible load directions is illustrated in Fig. 5.6.

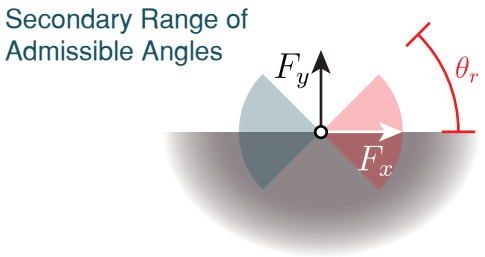


Figure 5.6: Schematic of the secondary range of admissible angles caused by the linearity of the state equations and the quadratic behavior of the von Mises stress.

5.1.3
Case 3: Planar load varying 360^o degrees plus a fixed load

In this section, we discuss the case in which we have a load varying 360^o degrees, plus a fixed load that does not varies in direction. A practical example that demonstrates the utility of this case is a bridge that is subjected to the self-weight load, which is always in the downward direction, plus a simplified load caused by the wind, which can vary in direction. In this case, we can express the loads as:

$$\mathbf{F}(\theta) = \mathbf{F}_x \cos(\theta) + \mathbf{F}_y \sin(\theta) + \mathbf{F}_f \quad (5\text{-}20)$$

where $\mathbf{F}_f$ is the load basis vector associated with the fixed load applied to the structure. Figure 5.7 displays an schematic of this load case. We can similarly define $\mathbf{U}_f = \mathbf{K}^{-1}F_f$, and $\boldsymbol{\sigma}_f = \mathbf{DBU}_f$, where $\mathbf{U}_f$ and $\boldsymbol{\sigma}_f$ are the displacement, and stress component caused by the fixed load, respectively. We can then derive an expression for the von Mises stress (similar to Case 1 in Section 5.1.1), i.e.:

(a) General Case $\|F_x\| \neq \|F_y\|$ (b) Particular Case $\|F_x\| = \|F_y\|$

Figure 5.7: Schematic of the domain of possible load cases (red), the load basis vectors (white) for the varying load and the fixed load (green), for case 3. (a) general load basis vector, i.e. $\|F_x\| \neq \left\|F_y\right\|$ forming an elliptic domain, and (b) load basis vectors with the same magnitude, i.e. $\|F_x\| = \left\|F_y\right\|$ forming a circular domain.

$$\sigma^v = \Big\{t_{xx}\cos^2(\theta) + t_{yy}\sin^2(\theta) + 2\,t_{xy}\cos(\theta)\sin(\theta) + 2\,t_{xf}\cos(\theta) + t_{ff} + 2\,t_{yf}\sin(\theta)\Big\}^{1/2} \tag{5-21}$$

in which we further define the extra quadratic stress terms $t_{ff} = \boldsymbol{\sigma}_f^T\mathbf{V}\boldsymbol{\sigma}_f$, $t_{xf} = \boldsymbol{\sigma}_x^T\mathbf{V}\boldsymbol{\sigma}_f$, and $t_{yf} = \boldsymbol{\sigma}_y^T\mathbf{V}\boldsymbol{\sigma}_f$. In this form, Eq. (5-21) has no clear optima, so we apply the Weierstrass variable substitution [108], also known as half-angle tangent substitution:

$$\sin(\theta) = \frac{2u}{1+u^2} \qquad\qquad \cos(\theta) = \frac{1-u^2}{1+u^2}, \tag{5-22}$$

which leads to the expression:

$$\sigma^v = \left\{\frac{1}{(u^2+1)^2}\Big[(t_{xx} + t_{ff} - 2\,t_{xf})u^4 + 4(t_{yf} - t_{xy})u^3 + 2(t_{ff} + 2\,t_{yy} - t_{xx})u^2 + 4(t_{yf} + t_{xy})u + t_{xx} + 2\,t_{xf} + t_{ff}\Big]\right\}^{1/2} \tag{5-23}$$

Now, we have to solve the related optimization problem:

$$\begin{aligned} &\max_{u\in\mathrm{R}} \quad \left[\sigma^v(\mathbf{z},u)\right]^2 \\ &\text{with:}\quad \mathbf{K}(\mathbf{z})\mathbf{U}_x = \mathbf{F}_x, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_y = \mathbf{F}_y, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_f = \mathbf{F}_f \end{aligned} \tag{5-24}$$

In order to find the solution to the problem in Eq. 5-24, we look for the critical points by differentiating the objective function with respect to the optimization variable and finding the roots of such expression:

$$\begin{aligned}\frac{\partial(\sigma^v(\mathbf{z},u))^2}{\partial u} &= \frac{4(t_{xy}-t_{yf})u^4+8(t_{xx}-t_{yy}-t_{xf})u^3}{(u^2+1)^3}+\\ &+\frac{-24\ t_{xy}u^2+4(t_{yy}-t_{xx}-t_{xf})u+4\ t_{xy}+4\ t_{yf}}{(u^2+1)^3}=0\end{aligned} \tag{5-25}$$

Notice that the denominator is never zero, and, therefore, we can find the roots of the above expression by looking exclusively to the numerator, which is a fourth order polynomial. The roots of a fourth order polynomial have closed expressions based on the rationals of the polynomial. These expressions are too extensive to be displayed here, but can easily be found in literature [45, 47]. Numerical experiments demonstrate that the maximum is attained at different roots depending on the values of the coefficients of the equation. Thus, we add a stress constraint for each root of Eq. (5-25). This means that our optimization problem will have four times the number of constraints of the original problem; however, only the constraints associated with the actual optima of the expression of the von Mises stress will be active at the optimal points, and our numerical experiments demonstrate that the AL formulation is able to accommodate the extra number of constraints without any detriment to the final solution. With this, the value for the critical von Mises stress is:

$$\begin{aligned}\widetilde{\sigma}^v = \Bigg\{ &\frac{1}{((u^\star)^2+1)^2}\Big[(t_{xx}+t_{ff}-2\ t_{xf})(u^\star)^4+4(t_{yf}-t_{xy})(u^\star)^3+\\ &+2(t_{ff}+2\ t_{yy}-t_{xx})(u^\star)^2+4(t_{yf}+t_{xy})(u^\star)+t_{xx}+2\ t_{xf}+t_{ff}\Big]\Bigg\}^{1/2}\end{aligned} \tag{5-26}$$

where $u^\star$ are the roots of the fourth degree polynomial displayed in Eq. (5-25).

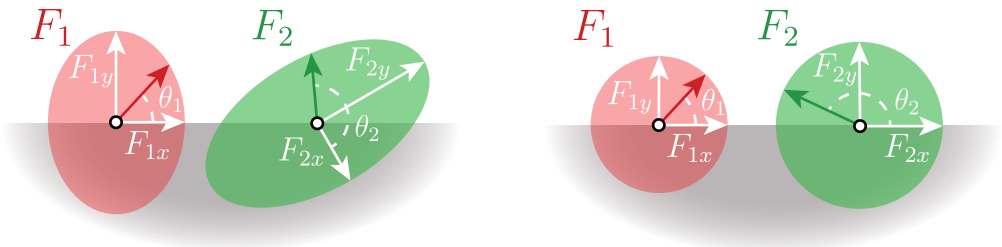
5.1.4 Case 4: Multiple loads varying independently with different angles

In this section, we address the case in which we have several loads varying independently of each other. In this case, the variable that controls the angle of the loads, $\boldsymbol{\theta}$, is a vector in $[-\pi,\pi]^n$, where n is the number of loads, and each component of this vector controls the angle of a different load. First, we develop a solution to this problem based on the simple case of $n=2$, i.e., two loads with two independent angles, and then we generalize the solution for an arbitrary number of loads, n. Let θ_1 and θ_2 be the first and second components of $\boldsymbol{\theta}$. Also, let $\mathbf{F}_1$ and $\mathbf{F}_2$ be the forces that vary in direction with θ_1, and θ_2, respectively. Figure 5.8 displays a schematic of this load case. We can write

each force as the sum of linearly independent components as before:

$$\mathbf{F} = \mathbf{F_1}(\theta_1) + \mathbf{F_2}(\theta_2) \tag{5-27}$$

$$\mathbf{F_1}(\theta_1) = \mathbf{F}_{1x}\cos(\theta_1) + \mathbf{F}_{1y}\sin(\theta_1) \qquad \mathbf{F_2}(\theta_2) = \mathbf{F}_{2x}\cos(\theta_2) + \mathbf{F}_{2y}\sin(\theta_2) \tag{5-28}$$



(a) General Case
$\|F_x\| \neq \|F_y\|$

(b) Particular Case
$\|F_x\| = \|F_y\|$

Figure 5.8: Schematic of the domain of possible load cases in red and green, and the load vector basis in white for case 4. (a) General load basis vector, i.e. $\|F_{1x}\| \neq \left\|F_{1y}\right\|$, and $\|F_{2x}\| \neq \left\|F_{2y}\right\|$ forming elliptic domains, and (b) load basis vectors with the same magnitude, i.e. $\|F_{1x}\| = \left\|F_{1y}\right\|$ and $\|F_{2x}\| = \left\|F_{2y}\right\|$ forming a circular domain.

We can then perform a similar derivation as in Section 5.1.1 to obtain:

$$\mathbf{U}_{1x} = \left(\mathbf{K}^{-1}\mathbf{F}_{1x}\right),\ \mathbf{U}_{1y} = \left(\mathbf{K}^{-1}\mathbf{F}_{1y}\right),\ \mathbf{U}_{2x} = \left(\mathbf{K}^{-1}\mathbf{F}_{2x}\right),\ \mathbf{U}_{2y} = \left(\mathbf{K}^{-1}\mathbf{F}_{2y}\right)$$

and the respective stresses:

$$\boldsymbol{\sigma}_{1x} = \mathbf{DBU}_{1x},\ \boldsymbol{\sigma}_{1y} = \mathbf{DBU}_{1y},\ \boldsymbol{\sigma}_{2x} = \mathbf{DBU}_{2x},\ \boldsymbol{\sigma}_{2y} = \mathbf{DBU}_{2y}.$$

The von Mises stress can then be computed as:

$$\begin{aligned}\sigma^v = \Big\{ & t_{1xx}\cos^2(\theta_1) + t_{1yy}\sin^2(\theta_1) + t_{1xy}\sin(\theta_1)\cos(\theta_1) + \\ & + t_{2xx}\cos^2(\theta_2) + t_{2yy}\sin^2(\theta_2) + t_{2xy}\sin(\theta_2)\cos(\theta_2) + \\ & + 2s_{xx}\cos(\theta_1)\cos(\theta_2) + 2s_{yy}\sin(\theta_1)\sin(\theta_2) + \\ & + 2s_{xy}\cos(\theta_1)\sin(\theta_2) + 2s_{yx}\sin(\theta_1)\cos(\theta_2)\Big\}^{1/2}\end{aligned} \tag{5-29}$$

By defining the quadratic stress terms as:

$$
\begin{aligned}
t_{1xx} &= \boldsymbol{\sigma}_{1x}\mathbf{V}\boldsymbol{\sigma}_{1x} \qquad t_{1yy} = \boldsymbol{\sigma}_{1y}\mathbf{V}\boldsymbol{\sigma}_{1y} \qquad t_{1xy} = \boldsymbol{\sigma}_{1x}\mathbf{V}\boldsymbol{\sigma}_{1y} \\
t_{2xx} &= \boldsymbol{\sigma}_{2x}\mathbf{V}\boldsymbol{\sigma}_{2x} \qquad t_{2yy} = \boldsymbol{\sigma}_{2y}\mathbf{V}\boldsymbol{\sigma}_{2y} \qquad t_{2xy} = \boldsymbol{\sigma}_{2x}\mathbf{V}\boldsymbol{\sigma}_{2y} \\
s_{xx} &= \boldsymbol{\sigma}_{1x}\mathbf{V}\boldsymbol{\sigma}_{2x} \qquad s_{yy} = \boldsymbol{\sigma}_{1y}\mathbf{V}\boldsymbol{\sigma}_{2y} \\
s_{xy} &= \boldsymbol{\sigma}_{1x}\mathbf{V}\boldsymbol{\sigma}_{2y} \qquad s_{yx} = \boldsymbol{\sigma}_{1y}\mathbf{V}\boldsymbol{\sigma}_{2x}
\end{aligned}
\tag{5-30}
$$

we further simplify Eq. (5-29), using trigonometric identities, to obtain:

$$
\begin{aligned}
\sigma^v = \Bigg\{ & t_{1xy}\sin(2\,\theta_1) + 0.5\left[\left(t_{1xx} - t_{1yy}\right)\cos(2\,\theta_1) + t_{1xx} + t_{1yy}\right] + \\
& t_{2xy}\sin(2\,\theta_2) + 0.5\left[\left(t_{2xx} - t_{2yy}\right)\cos(2\,\theta_2) + t_{2xx} + t_{2yy}\right] + \\
& \left(s_{xy} + s_{yx}\right)\sin\left(\frac{\theta_1+\theta_2}{2}\right) + \left(s_{xx} - s_{yy}\right)\cos\left(\frac{\theta_1+\theta_2}{2}\right) + \\
& \left(s_{yx} - s_{xy}\right)\sin\left(\frac{\theta_1-\theta_2}{2}\right) + \left(s_{xx} + s_{yy}\right)\cos\left(\frac{\theta_1-\theta_2}{2}\right)\Bigg\}^{1/2}
\end{aligned}
\tag{5-31}
$$

We then have to solve the related optimization problem:

$$
\begin{aligned}
&\max_{\boldsymbol{\theta}\in\Gamma} \quad \left[\sigma^v(\mathbf{z},\boldsymbol{\theta})\right]^2 \\
&\text{with:} \quad \mathbf{K}(\mathbf{z})\mathbf{U}_{1x} = \mathbf{F}_{1x}, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_{1y} = \mathbf{F}_{1y} \\
&\qquad\qquad \mathbf{K}(\mathbf{z})\mathbf{U}_{2x} = \mathbf{F}_{2x}, \quad \mathbf{K}(\mathbf{z})\mathbf{U}_{2y} = \mathbf{F}_{2y}
\end{aligned}
\tag{5-32}
$$

This optimization problem is more complicated than the previous ones, and we were not able to obtain an analytic solution for its solution; however, we can obtain an analytic upper bound on the objective function. Again, we square the von Mises stress in the objective function to eliminate the square root on the right-hand side, because it does not influence the solution of the optimization problem. We then decompose the expression of the squared von Mises stress into three parts:

$$
\begin{aligned}
\xi_1 &= t_{1xy}\sin(2\,\theta_1) + 0.5\left[\left(t_{1xx} - t_{1yy}\right)\cos(2\,\theta_1) + t_{1xx} + t_{1yy}\right] \\
\xi_2 &= t_{2xy}\sin(2\,\theta_2) + 0.5\left[\left(t_{2xx} - t_{2yy}\right)\cos(2\,\theta_2) + t_{2xx} + t_{2yy}\right] \\
\xi_{12} &= \left(s_{xy} + s_{yx}\right)\sin\left(\frac{\theta_1+\theta_2}{2}\right) + \left(s_{xx} - s_{yy}\right)\cos\left(\frac{\theta_1+\theta_2}{2}\right) + \\
&\quad \left(s_{yx} - s_{xy}\right)\sin\left(\frac{\theta_1-\theta_2}{2}\right) + \left(s_{xx} + s_{yy}\right)\cos\left(\frac{\theta_1-\theta_2}{2}\right)
\end{aligned}
\tag{5-33}
$$

The maximum of the sum of these terms is less than or equal to the sum of the maximum of each term. Therefore, we compute the maximum of each

term and use the sum of the three terms as an upper bound on the critical stress. The expressions for ξ_1 and ξ_2 resemble Eq. (5-8), which we have already derived a solution for. Thus, we have:

$$\begin{aligned}\xi_1^\star &= t_{1xy}\sin\left(2\,\theta_1^\star\right) + 0.5\left[\Big(t_{1xx} - t_{1yy}\Big)\cos\left(2\,\theta_1^\star\right) + t_{1xx} + t_{1yy}\right] \\ \xi_2^\star &= t_{2xy}\sin\left(2\,\theta_2^\star\right) + 0.5\left[\Big(t_{2xx} - t_{2yy}\Big)\cos\left(2\,\theta_2^\star\right) + t_{2xx} + t_{2yy}\right]\end{aligned} \tag{5-34}$$

with:

$$\begin{aligned}\theta_1^\star &= \frac{1}{2}\tan^{-1}\Big(\,2\,t_{1xy},\;t_{1xx} - t_{1yy}\,\Big) \\ \theta_2^\star &= \frac{1}{2}\tan^{-1}\Big(\,2\,t_{2xy},\;t_{2xx} - t_{2yy}\,\Big)\end{aligned} \tag{5-35}$$

To find a solution for the ξ_{12} term, we substitute $u = \dfrac{\theta_1 + \theta_2}{2}$ and $v = \dfrac{\theta_1 - \theta_2}{2}$ into the expression for ξ_{12} in Eq. (5-34), and separate ξ_{12} in two terms, $\xi_{12u} + \xi_{12v}$, each containing only terms with u, or v:

$$\begin{aligned}\xi_{12u} &= \Big(s_{xy} + s_{yx}\Big)\sin\left(u\right) + \Big(s_{xx} - s_{yy}\Big)\cos\left(u\right) \\ \xi_{12v} &= \Big(s_{yx} - s_{xy}\Big)\sin\left(v\right) + \Big(s_{xx} + s_{yy}\Big)\cos\left(v\right)\end{aligned} \tag{5-36}$$

Differentiating the expressions in Eq. (5-36) and setting them equal to zero, we find the following critical points:

$$\begin{aligned}u^{cr} &= \begin{cases}\tan^{-1}\Big(\,s_{xy} + s_{yx},\;s_{xx} - s_{yy}\,\Big) \\ \\ \tan^{-1}\Big(\,-(s_{xy} + s_{yx}),\;-(s_{xx} - s_{yy})\,\Big)\end{cases} \\ v^{cr} &= \begin{cases}\tan^{-1}\Big(\,s_{yx} - s_{xy},\;s_{xx} + s_{yy}\,\Big) \\ \\ \tan^{-1}\Big(\,-(s_{yx} - s_{xy}),\;-(s_{xx} + s_{yy})\,\Big)\end{cases}\end{aligned} \tag{5-37}$$

We denote the first solutions in Eq. (5-37) u^{cr}_{max} and v^{cr}_{max} and the second solutions u^{cr}_{min} and v^{cr}_{min}. By taking second derivatives with respect to u and v and evaluating them at the critical points, we find:

$$\left.\frac{\partial^2\,\xi_{12}}{\partial u^2}\right|_{u=u^{cr}_{max}} = -\left\{\Big(s_{xy} + s_{yx}\Big)^2 + \Big(s_{xx} - s_{yy}\Big)^2\right\}^{1/2} \tag{5-38}$$

$$\left.\frac{\partial^2\,\xi_{12}}{\partial u^2}\right|_{u=u^{cr}_{min}} = \left\{\Big(s_{xy} + s_{yx}\Big)^2 + \Big(s_{xx} - s_{yy}\Big)^2\right\}^{1/2} \tag{5-39}$$

$$\left.\frac{\partial^2 \xi_{12}}{\partial v^2}\right|_{v=v^{cr}_{max}} = -\left\{\left(s_{xy} - s_{yx}\right)^2 + \left(s_{xx} + s_{yy}\right)^2\right\}^{1/2} \tag{5-40}$$

$$\left.\frac{\partial^2 \xi_{12}}{\partial v^2}\right|_{v=v^{cr}_{min}} = \left\{\left(s_{xy} - s_{yx}\right)^2 + \left(s_{xx} + s_{yy}\right)^2\right\}^{1/2} \tag{5-41}$$

The second derivatives of ξ_{12} with respect to u and v are always non-positive at $u = u^{cr}_{max}$ and $v = v^{cr}_{max}$ and always non-negative at $u = u^{cr}_{min}$ and $v = v^{cr}_{min}$, meaning that the point $(u^{cr}_{max}, v^{cr}_{max})$ is a local maximum, $(u^{cr}_{min}, v^{cr}_{min})$ is a local minimum, and $(u^{cr}_{max}, v^{cr}_{min})$ and $(u^{cr}_{min}, v^{cr}_{max})$ are saddle points. Since these are the only critical points in the domain and the function is smooth and periodic, $(u^{cr}_{max}, v^{cr}_{max})$ is the global maximum and $(u^{cr}_{min}, v^{cr}_{min})$ is the global minimum. Therefore, the maximum point of ξ_{12} is:

$$u^\star = \tan^{-1}\left(\, s_{xy} + s_{yx},\ s_{xx} - s_{yy}\,\right) \qquad v^\star = \tan^{-1}\left(\, s_{yx} - s_{xy},\ s_{xx} + s_{yy}\,\right), \tag{5-42}$$

and the maximum is:

$$\begin{aligned}\xi^\star_{12} = &\left(s_{xy} + s_{yx}\right)\sin\left(u^\star\right) + \left(s_{xx} - s_{yy}\right)\cos\left(u^\star\right) \\ &+ \left(s_{yx} - s_{xy}\right)\sin\left(v^\star\right) + \left(s_{xx} + s_{yy}\right)\cos\left(v^\star\right)\end{aligned} \tag{5-43}$$

To obtain the upper bound, $\widehat{\sigma}^v$, for $\widetilde{\sigma}^v$, we sum $\xi^\star_1$, $\xi^\star_2$ and $\xi^\star_{12}$, i.e.:

$$\begin{aligned}\widehat{\sigma}^v = &\left\{\widetilde{\xi}_1 + \widetilde{\xi}_2 + \widetilde{\xi}_{12}\right\}^{1/2} = \\ &\Bigg\{t_{1xy}\sin\left(2\,\theta^\star_1\right) + 0.5\left[\left(\,t_{1xx} - \,t_{1yy}\right)\cos\left(2\,\theta^\star_1\right) + \,t_{1xx} + \,t_{1yy}\right] + \\ &t_{2xy}\sin\left(2\,\theta^\star_2\right) + 0.5\left[\left(\,t_{2xx} - \,t_{2yy}\right)\cos\left(2\,\theta^\star_2\right) + \,t_{2xx} + \,t_{2yy}\right] + \\ &\left(s_{xy} + s_{yx}\right)\sin\left(u^\star\right) + \left(s_{xx} - s_{yy}\right)\cos\left(u^\star\right) + \\ &\left(s_{yx} - s_{xy}\right)\sin\left(v^\star\right) + \left(s_{xx} + s_{yy}\right)\cos\left(v^\star\right)\Bigg\}^{1/2}\end{aligned} \tag{5-44}$$

By using the upper bound, $\widehat{\sigma}^v$, in the stress constraints, we can guarantee that the stress will be below the limit for any of the possible load cases.

5.1.4.1 Error Analysis of Critical Stress Upper Bound

The use of an upper bound on the worst-case stress constraint leads to an overestimation of the actual maximum stress caused on the structure by the loads. This overestimation of the stress can lead to an over design resulting in

structures that have extra unnecessary material. Therefore, the more accurate this upper bound is, i.e. the closer it is to the actual worst-case stress, the less extra unnecessary material the optimal structure will have. To evaluate the accuracy of such upper bound, we generated a sample of 100 million uniformly distributed random stress basis vectors ($\boldsymbol{\sigma}_{1x},\ \boldsymbol{\sigma}_{1y},\ \boldsymbol{\sigma}_{2x},\ \boldsymbol{\sigma}_{2y}$), and computed an percent error (see Eq. (5-45)) comparing the worst-case stress with the proposed upper bound using these random stress basis vectors. The result of such statistic analysis of the error is displayed in Fig. 5.9 in the form of a histogram. In this histogram, we see that the mean error was 3.18% with an 1.8 standard deviation, and that the maximum percent error was 12.8%. We also notice that this probability distribution is skewed to the left, indicating that the error rarely attains values close to this maximum error. With this analysis, we show that the percent error is moderate, and that the proposed upper bound is significantly accurate.

To verify that this sample of 100 million random stress basis vectors is representative of the underlying probability distribution, we performed a numerical convergence analysis by starting with a sample of 10 million and increasing its size, 10 million by 10 million, until we reached the 100 million sample size. We also generated 10 different samples of 100 million elements, which proved to be numerically identical. With this two numerical experiments, we guaranteed that the sample size used was large enough to provide us with an trustworthy representation of the underlying probability distribution.

The use of an upper bound on the worst-case stress constraint brings up the need to evaluate the accuracy of such upper bound. Ideally, the closer such upper bound is to actual worst-case stress the better will, because

$$\text{Percent Error } (\%) \ = \left| \frac{\left\{\xi_1^\star + \xi_2^\star + \xi_{12}^\star\right\}^{1/2} - \sigma_{cr}^v}{\sigma_{cr}^v} \right| \times 100 \qquad (5\text{-}45)$$

in which $\left\{\xi_1^\star + \xi_2^\star + \xi_{12}^\star\right\}^{1/2}$ is the proposed upper bound for the worst-case stress, and σ_{cr}^v is the worst-case stress.

5.1.4.2 Limiting the range of θ_1 and θ_2

If we want to restrict the range of one, or both load angles, we can do so by restricting each expression in Eq. (5-33) individually, and applying the solution presented in Section 5.1.2. Furthermore, we can construct the admissible range for u and v based on θ_{1r} and θ_{2r} as $[-0.5(\theta_{1r}+\theta_{2r}), 0.5(\theta_{1r}+\theta_{2r})]$ for both u and v, in which $[-\theta_{1r}, \theta_{1r}]$ and $[-\theta_{2r}, \theta_{2r}]$ are the range of admissible angles for θ_1 and θ_2, respectively. With this limited range we have:

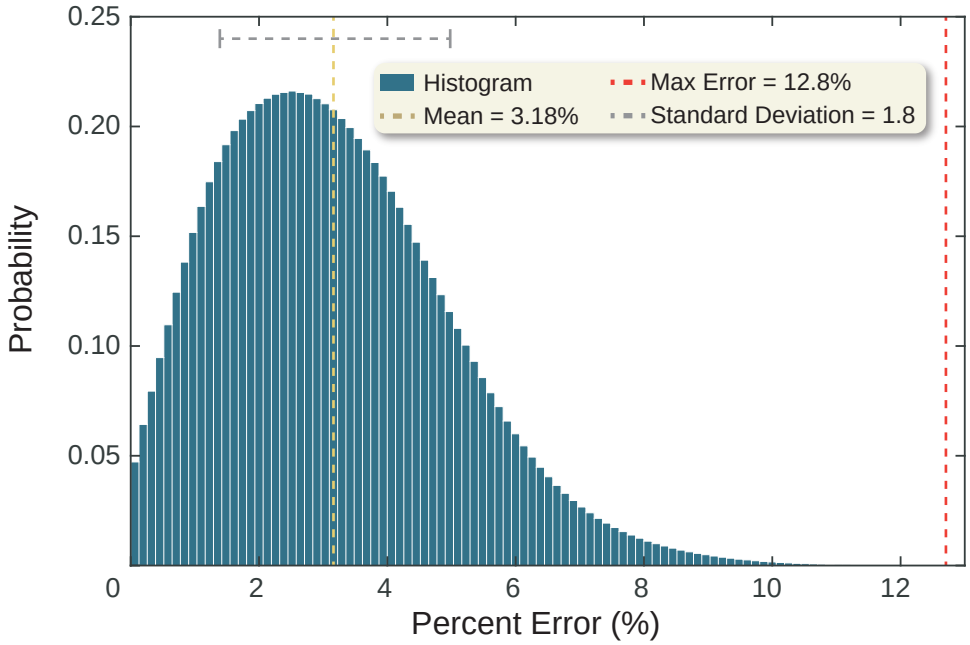


Figure 5.9: Histogram generated using a sample of 100 million uniformly distributed random stress basis vectors representing the underlying probability distribution of the percent error (see Eq. (5-45)) between the proposed upper bound, and the worst-case stress.

$$
\begin{aligned}
\theta_1^\star &= \min\left\{ \max\left[\frac{1}{2}\tan^{-1}\left(2\,t_{1xy},\; t_{1xx} - t_{1yy} \right),\; -\theta_{1r} \right],\; \theta_{1r} \right\} \\
\theta_2^\star &= \min\left\{ \max\left[\frac{1}{2}\tan^{-1}\left(2\,t_{2xy},\; t_{2xx} - t_{2yy} \right),\; -\theta_{2r} \right],\; \theta_{2r} \right\} \\
u^\star &= \min\left\{ \max\left[\tan^{-1}\left(s_{xy} + s_{yx},\; s_{xx} - s_{yy} \right),\; -0.5(\theta_{1r} + \theta_{2r}) \right],\; 0.5(\theta_{1r} + \theta_{2r}) \right\} \\
v^\star &= \min\left\{ \max\left[\tan^{-1}\left(s_{yx} - s_{xy},\; s_{xx} + s_{yy} \right),\; -0.5(\theta_{1r} + \theta_{2r}) \right],\; 0.5(\theta_{1r} + \theta_{2r}) \right\}
\end{aligned}
\tag{5-46}
$$

5.1.4.3 Generalization to more than two independent loads

To generalize this approach for an arbitrary number of loads represented by an arbitrary number of angles θ_i, notice that the interactions between the loads in Eq. (5-29) occurs pairwise, because the expression of the von Mises stress is bilinear. Therefore, we can obtain a similar estimation for the upper bound of the von Mises stress by separating the terms that depend exclusively on θ_i and the cross terms between θ_i and θ_j for $i \neq j$, and then summing them all together. Equation (5-47) displays this approach for a general number n of angles θ_i, in which the terms ξ_i and ξ_{ij} are expressed in Eq. (5-33) by replacing

the appropriate indexes:

$$\widehat{\sigma}^{v} = \left\{ \sum_{i=1}^{n} \left(\xi_i + \sum_{j=i+1}^{n} \xi_{ij} \right) \right\}^{1/2} \tag{5-47}$$

in which the indexes $i, j = 1, ..., n$ refer to the angles θ_i and θ_j that control the loads $\mathbf{F}_i$ and $\mathbf{F}_j$, respectively.

5.1.5 Case 5: Load varying in 3D

The last case presented in this work is for 3D problems in which the loads can vary, not only in a plane, but also in out-of-plane directions, representing a whole surface of possible load directions. To account for these out-of-plane load components, we combine two loads varying independently in direction (with θ_1 and θ_2), with the appropriate basis, using the derivations of case 4. As seen in Fig. 5.10(a), by combining two independent loads with basis that form orthogonal planes, we obtain a 3D load surface that accounts for out-of-plane load components. Furthermore, by combining independent loads of different forms we can obtain different 3D load surfaces (5.10 (b) and (c)). Notice that, by setting one of the basis (F_y) to zero in Fig. 5.10(b) and (c), we obtain a load that only varies in intensity, from $-F_x$ to F_x, in a fixed direction.

5.2 Generalization of load decomposition and varying load intensity

All of the methodology developed in the previous section is based on decomposing the load into load basis vectors that we can use to obtain upper bounds to the worst-case von Mises stress. However, no assumption was made about the direction or magnitude of these basis vectors (other than the basis vectors being linearly independent). Consequently, the previously obtained solutions are valid even if F_x and F_y have different magnitudes, or if F_x and F_y are not aligned with the x and y axis. If we change the magnitude of F_x in relation to F_y, we obtain a domain of load cases that forms an ellipse instead of a circle, as it is displayed in Figures 5.11(a) and (b). We can also rotate this ellipse by rotating the basis vectors, as displayed in Figures 5.11 (c) and (d).[2] This arbitrary choice of load basis vectors allows us to consider a wider set of load domains. This freedom to choose the load basis vector provides extra flexibility to the framework, making it possible for the designer to define load cases in which a specific load direction is more relevant than the others.

[2]We use this property in Case 2 (Section 5.1.2) to shift the range of admissible angles to be centered around the origin.

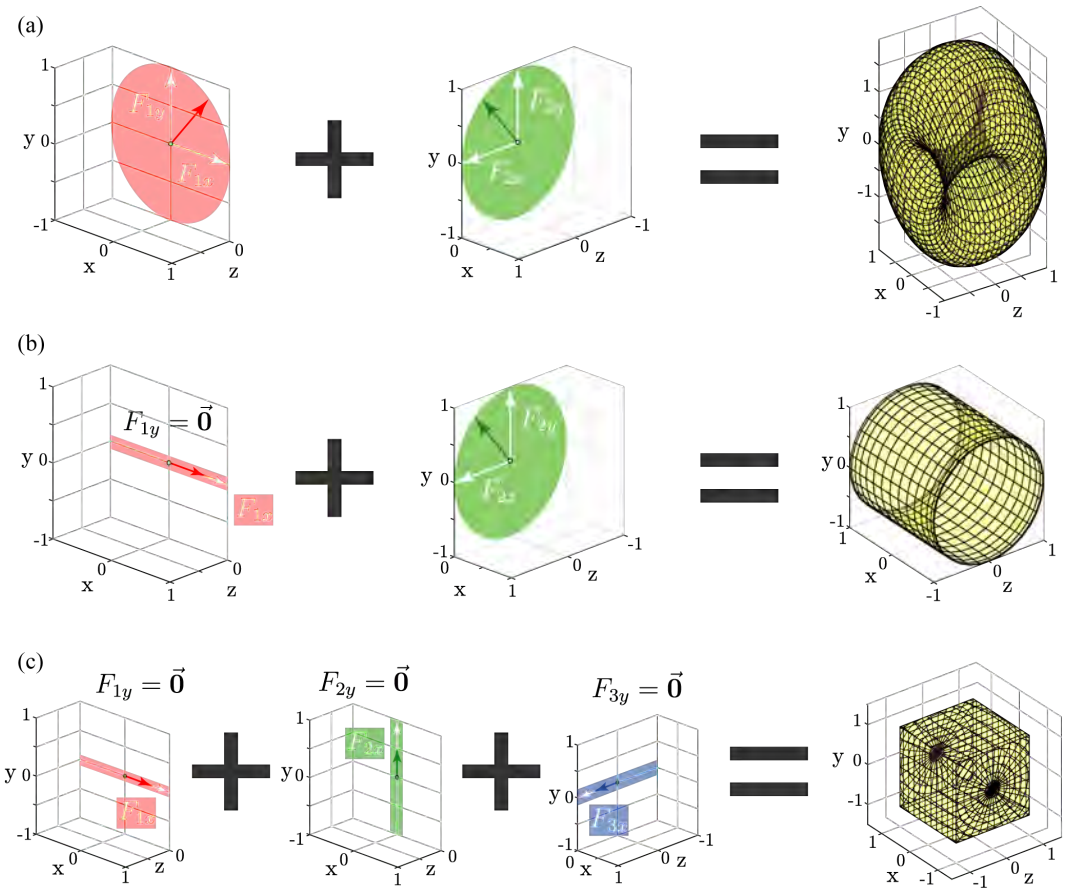


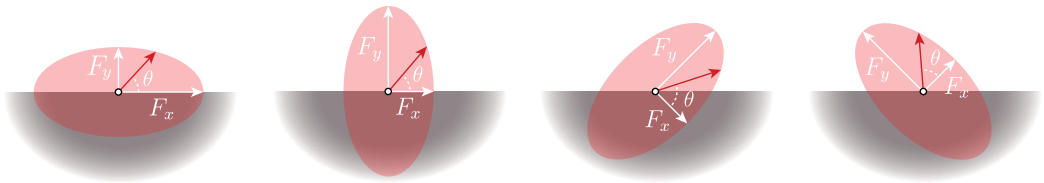
Figure 5.10: Schematic of how to combine loads with varying with independent angles to achieve a load that varies in 3D.

5.3 Critical Stress Sensitivity Analysis

In this section we present the sensitivity analysis in respect to the design variables of the expressions derived in section 5.1 for the worst-case multiple load directions formulation. The sensitivity information is necessary because the topology optimization problem is solved using gradient-based optimization algorithms [30]. We will also provide proof of the differentiability of the worst-case stress analytical expression presented previously. To compute the sensitivity information, we will use the chain rule starting by differentiating the AL function:

$$\frac{\mathrm{d}J^{(k)}}{\mathrm{d}z_j} = \frac{\partial J^{(k)}}{\partial M(\mathbf{z})}\frac{\partial M(\mathbf{z})}{\partial z_j} + \sum_{i=1}^{N_e}\frac{\partial J^{(k)}}{\partial g_i}\frac{\partial g_i}{\partial z_j} = \frac{\mathrm{d}M(\mathbf{z})}{\mathrm{d}z_j} + \sum_{i=1}^{N_e}(\lambda_i^{(k)} + \mu^{(k)} g_i)\frac{\mathrm{d}g_i}{\mathrm{d}z_j} \quad (5\text{-}48)$$

The derivation of the sensitivity analysis for the AL stress constrained formulation without considering continuously varying load cases is presented in Section 3.3, and can be used as support for the following derivations. We will



(a) $\|F_x\| = 2\,\|F_y\|$ (b) $2\,\|F_x\| = \|F_y\|$ (c) $2\,\|F_x\| = \|F_y\|$, rotated -45^o (d) $2\,\|F_x\| = \|F_y\|$, rotated 45^o

Figure 5.11: Load case 1 with different magnitudes and different orientations for the load basis vectors F_x and F_y.

now focus on the derivative of the critical stress constraints, because their formulation is one of the main contribution of this work:

$$\frac{\mathrm{d}g_i}{\mathrm{d}z_j} = \frac{\partial g_i}{\partial z_j} + \frac{\partial g_i}{\partial \widetilde{\sigma}_i^v}\frac{\partial \widetilde{\sigma}_i^v}{\partial z_j} \tag{5-49}$$

The partial derivative of the constraint in respect to the design variable is simple to compute, and it can be written as:

$$\frac{\partial g_i}{\partial z_j} = p\tilde{\rho}_i^{p-1}\left[(\sigma_j^v/\sigma_{\text{lim}} - 1)^2 + 0.1(\sigma_j^v/\sigma_{\text{lim}} - 1)\right]\frac{\partial \rho_i}{\partial z_j} \tag{5-50}$$

Next, if we expand the second term with the derivation of the constraint in respect to the von Mises stress, we obtain:

$$\frac{\partial g_i}{\partial \widetilde{\sigma}_i^v}\frac{\partial \widetilde{\sigma}_i^v}{\partial z_j} = \frac{\rho_j^p}{\sigma_{\text{lim}}}\left[2(\sigma_j^v/\sigma_{\text{lim}} - 1) + 0.1\right]\frac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i}\frac{\partial (\sigma_c)_i}{\partial z_j}, \tag{5-51}$$

where the term $(\sigma_c)_i$ represent the stress components of each sub-case, in which the index c represent the associated stress component, and the index i represent the associated constraint. For each different case displayed in Section 5.1 of the main manuscript, we have a different expression for $\dfrac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i}$, therefore, Sections 5.3.1, 5.3.2, and 5.3.3, will present the derivation of this term for each of the cases. The derivation of the term $\dfrac{\partial (\sigma_c)_i}{\partial z_j}$ is very similar for all cases, and, because of this, its derivation will be presented in the end, in Section 5.3.4.

5.3.1
Sensitivity of case 1 and case 2: Planar load varying in direction

In this Section we will display the sensitivity for the first and second cases. We display the sensitivity for these two cases together because the sensitivity of both cases is practically identical. As a matter of fact, case 1 can be thought of as a sub-case of case 2 for which the limiting angle is $\theta_r = \pi$. The expression for this case is derived in Sections 5.1.1 and 5.1.2, and it is displayed here again to help the reader follow the sensitivity derivations:

$$\widetilde{\sigma}^v = \left\{ t_{xy} \sin\left(2\,\theta^\star\right) + 0.5 \left[\Big(t_{xx} - t_{yy} \Big) \cos\left(2\,\theta^\star\right) + t_{xx} + t_{yy} \right] \right\}^{1/2} \tag{5-52}$$

with:

$$\theta^\star = \min\left\{ \max\left[\theta_1^{cr},\ -\theta_r \right],\ \theta_r \right\} \tag{5-53}$$

We decompose the derivative of $\widetilde{\sigma}^v$ using the chain rule as:

$$\frac{\mathrm{d}\widetilde{\sigma}_i^v}{\mathrm{d}(\sigma_c)_i} = \frac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i} + \frac{\partial \widetilde{\sigma}_i^v}{\partial \theta^\star} \frac{\partial \theta^\star}{\partial (\sigma_c)_i} \tag{5-54}$$

However, notice that $\frac{\partial \widetilde{\sigma}_i^v}{\partial \theta^\star} = 0$, because if $\theta^\star = -\theta_r$ or $\theta^\star = \theta_r$ then $\theta^\star$ does not depend on z_j, and if $\theta^\star = \theta_{max}^{cr}$ then $\theta^\star$ is a critical point of the expression for the von Mises stress then $\frac{\partial \widetilde{\sigma}_i^v}{\partial \theta^\star} = 0$ by the definition of θ_{max}^{cr}. Therefore, we only need to compute:

$$\frac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i} = \frac{1}{2\widetilde{\sigma}_i^v} \left[\sin(2\,\theta^\star) \frac{\partial t_{xy}}{\partial (\sigma_c)_i} + 0.5(1+\cos(\theta^\star)) \frac{\partial t_{xx}}{\partial (\sigma_c)_i} + 0.5(1-\cos(\theta^\star)) \frac{\partial t_{yy}}{\partial (\sigma_c)_i} \right] \tag{5-55}$$

5.3.2
Sensitivity of case 3: Planar load varying in direction plus a fixed load

This Section presents the derivation of the sensitivity of case 3, a planar load varying 360^o degrees in direction plus a fixed load. The expression for this cases is derived in Section 5.1.3. The expression for the critical stress is displayed again here to facilitate the sensitivity derivations:

$$\begin{aligned} \widetilde{\sigma}^v = \Bigg\{ & \frac{1}{\left[(u^\star)^2+1\right]^2} \Big[(t_{xx} + t_{ff} - 2\,t_{xf})(u^\star)^4 + 4(t_{yf} - t_{xy})(u^\star)^3 \\ & + 2(t_{ff} + 2\,t_{yy} - t_{xx})(u^\star)^2 + 4(t_{yf} + t_{xy})(u^\star) + t_{xx} + 2\,t_{xf} + t_{ff} \Big] \Bigg\}^{1/2} \end{aligned} \tag{5-56}$$

where $u^\star$ are the critical points of the expression above in relation to $u^\star$. We decompose the derivative of $\widetilde{\sigma}^v$ using the chain rule:

$$\frac{\mathrm{d}\widetilde{\sigma}_i^v}{\mathrm{d}(\sigma_c)_i} = \frac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i} + \frac{\partial \widetilde{\sigma}_i^v}{\partial u^\star} \frac{\partial u^\star}{\partial (\sigma_c)_i} \tag{5-57}$$

Similar to the previous case, we have that $\frac{\partial \widetilde{\sigma}_i^v}{\partial u^\star} = 0$, because $u^\star$ is a critical point of the expression for the von Mises stress by definition. Therefore, the sensitivity for this case reduces to:

$$\frac{\partial \widetilde{\sigma}_i^v}{\partial (\sigma_c)_i} = \frac{1}{2\widetilde{\sigma}_i^v \left[(u^\star)^2+1\right]^2} \left\{ \left[(u^\star)^4 - 2\,(u^\star)^2 + 1\right] \frac{\partial t_{xx}}{\partial (\sigma_c)_i} + 4\,(u^\star)^2 \frac{\partial t_{yy}}{\partial (\sigma_c)_i} \right.$$
$$+ \left[(u^\star)^4 + 2\,(u^\star)^2 + 1\right] \frac{\partial t_{ff}}{\partial (\sigma_c)_i} - 4\,\left[(u^\star)^3 - \,(u^\star)\right] \frac{\partial t_{xy}}{\partial (\sigma_c)_i}$$
$$\left. -2\,\left[(u^\star)^4 - 1\right] \frac{\partial t_{xf}}{\partial (\sigma_c)_i} + 4\,\left[(u^\star)^3 + (u^\star)\right] \frac{\partial t_{yf}}{\partial (\sigma_c)_i} \right\} \tag{5-58}$$

5.3.3
Sensitivity of case 4: Multiple Planar loads varying independently

This Section presents the derivation of the sensitivity of case 4, displayed in Section 5.1.4, in which we have multiple loads varying independently in direction. In order to simplify the expression, we will split the computation of the sensitivity, and consider the sensitivity of ξ_1, ξ_2 and ξ_{12} separately. The expressions for ξ_1 and ξ_2 are identical to the expression for a single load varying in direction for which the sensitivity has already been computed in Section 5.3.1. Therefore, we refrain from repeating it here and we will focus on the sensitivity of ξ_{12}. We have that:

$$\frac{\mathrm{d}\xi_{12}}{\mathrm{d}(\sigma_c)_i} = \frac{\partial \xi_{12}}{\partial (\sigma_c)_i} + \frac{\partial \xi_{12}}{\partial u^\star}\frac{\partial u^\star}{\partial (\sigma_c)_i} + \frac{\partial \xi_{12}}{\partial v^\star}\frac{\partial v^\star}{\partial (\sigma_c)_i} \tag{5-59}$$

Similar to previous cases, $u^\star$ and $v^\star$ are critical points of the expression for ξ_{12}, and, consequently, $\frac{\partial \xi_{12}}{\partial u^\star} = 0$ and $\frac{\partial \xi_{12}}{\partial v^\star} = 0$. The remaining term reads:

$$\frac{\mathrm{d}\xi_{12}}{\mathrm{d}(\sigma_c)_i} = \frac{\partial \xi_{12}}{\partial (\sigma_c)_i} = \left[\cos(u^\star) + \cos(v^\star)\right] \frac{\partial s_{xx}}{\partial (\sigma_c)_i} + \left[\cos(v^\star) - \cos(u^\star)\right] \frac{\partial s_{yy}}{\partial (\sigma_c)_i} +$$
$$\left[\sin(u^\star) - \sin(v^\star)\right] \frac{\partial s_{xy}}{\partial (\sigma_c)_i} + \left[\sin(u^\star) + \sin(v^\star)\right] \frac{\partial s_{yx}}{\partial (\sigma_c)_i} \tag{5-60}$$

The full expression of the sensitivity for this case is:

$$\frac{\mathrm{d}\widetilde{\sigma}_i^v}{\mathrm{d}(\sigma_c)_i} = \frac{1}{2\widetilde{\sigma}_i^v} \left[\frac{\partial \xi_1}{\partial (\sigma_c)_i} + \frac{\partial \xi_2}{\partial (\sigma_c)_i} + \frac{\partial \xi_{12}}{\partial (\sigma_c)_i} \right] \tag{5-61}$$

where the expressions for $\frac{\partial \xi_1}{\partial (\sigma_c)_i}$ and $\frac{\partial \xi_2}{\partial (\sigma_c)_i}$ are the same as the expression for the sensitivity of case 1.

5.3.4
Sensitivity of the Stress Components

In order to compute the sensitivity of any of the cases displayed in this work, it is necessary to compute the sensitivity of the quadratic stress terms (e.g. $\partial\, t_{xx}/\partial\, \sigma_c$, $\partial\, t_{yy}/\partial\, \sigma_c$ and $\partial\, t_{xy}/\partial\, \sigma_c$ for case 1). The sensitivity of the

quadratic stress terms follows a basic formula, for all the cases displayed in this work. Therefore, the sensitivity for an arbitrary quadratic stress terms is derived in this Section, and it can be used for any of the cases presented. Let Υ_{ab} be the quadratic stress terms generated by $\Upsilon_{ab} = \sigma_a \mathbf{V} \sigma_b$, e.g. in case 1, $\Upsilon_{xy} = t_{xy} = \sigma_x \mathbf{V} \sigma_y$. Then we have that:

$$\frac{\partial \Upsilon_{ab}}{\partial \sigma_c} = \frac{\partial \sigma_a}{\partial \sigma_c} \mathbf{V} \sigma_b + \sigma_a \mathbf{V} \frac{\partial \sigma_b}{\partial \sigma_c} \tag{5-62}$$

in which σ_c is a stress component. Now we focus on the derivation of the stress component in respect to the design variable, written as:

$$\frac{\partial (\sigma_c)_i}{\partial z_j} = \frac{\partial (\sigma_c)_i}{\partial U_{lm}} \frac{\partial U_{lm}}{\partial z_j} \tag{5-63}$$

where the the index c of the variable σ_c represents the stress component associated with that variable, and the outermost index i of $(\sigma_c)_i$ represents the constraint associated with the i-th stress constraint. Furthermore, the first index l of the term U_{lm} represents the displacement generated by the load basis vector F_l, while the second index m represents the degrees of freedom of the displacement vector. The term $\dfrac{\partial U_{lm}}{\partial z_k}$ is computed using the adjoint method. In the adjoint method, we differentiate the equilibrium equations associated to our problem considering all the load basis vectors:

$$\frac{\partial}{\partial z_j} \left(K_{nl} U_{lm} - F_{nm} \right) = \frac{\partial K_{nl}}{\partial z_j} U_{lm} + K_{nl} \frac{\partial U_{lm}}{\partial z_j} = 0 \tag{5-64}$$

Because the above Eq. is equal to zero, we can multiply it by a factor $\boldsymbol{\xi}$, and add it to our sensitivity equation (Eq. 5-48), without altering its value. For simplicity, we only show the relevant terms of Eq. (5-48), i.e. we ignore the terms related to the objective function and the partial derivative $\dfrac{\partial g_i}{\partial z_j}$, and focus only on the terms related to $\dfrac{\partial g_i}{\partial \widetilde{\sigma}_i^v} \dfrac{\partial \widetilde{\sigma}_i^v}{\partial z_j}$. So, we have:

$$\frac{\partial J^{(k)}}{\partial \widetilde{\sigma}_i^v} \frac{\partial \widetilde{\sigma}_i^v}{\partial z_j} = \sum_{i=1}^{N_e} (\lambda_i^{(k)} + \mu^{(k)} g_i) \frac{\partial g_i}{\partial \widetilde{\sigma}_i^v} \frac{\partial \widetilde{\sigma}_i^v}{(\partial \sigma_c)_i} \frac{\partial (\sigma_c)_i}{\partial U_{lm}} \frac{\partial U_{lm}}{\partial z_j} + \boldsymbol{\xi}_{nm} \left(\frac{\partial K_{nl}}{\partial z_j} U_{lm} + K_{nl} \frac{\partial U_{lm}}{\partial z_j} \right) \tag{5-65}$$

If we collect the terms containing $\dfrac{\partial U_{lm}}{\partial z_j}$:

$$\frac{\partial J^{(k)}}{\partial \widetilde{\sigma}_i^v} \frac{\partial \widetilde{\sigma}_i^v}{\partial z_j} = \left[\sum_{i=1}^{N_e} (\lambda_i^{(k)} + \mu^{(k)} g_i) \frac{\partial g_i}{\partial \widetilde{\sigma}_i^v} \frac{\partial \widetilde{\sigma}_i^v}{(\partial \sigma_c)_i} \frac{\partial (\sigma_c)_i}{\partial U_{lm}} + \boldsymbol{\xi}_{nm} K_{nl} \right] \frac{\partial U_{lm}}{\partial z_j} + \boldsymbol{\xi}_{nm} \frac{\partial K_{nl}}{\partial z_j} U_{lm} \tag{5-66}$$

We can then set the value of factor $\boldsymbol{\xi}_{nm}$ so that the term in brackets becomes zero. In order to do that we have to solve the Eq.:

$$K_{nl}\boldsymbol{\xi}_{nm} = \sum_{i=1}^{N_e}(\lambda_i^{(k)} + \mu^{(k)} g_i)\frac{\partial g_i}{\partial \widetilde{\sigma}_i^v}\frac{\partial \widetilde{\sigma}_i^v}{(\partial \sigma_c)_i}\frac{\partial (\sigma_c)_i}{\partial U_{lm}} \tag{5-67}$$

Therefore, we have to solve the system K_{nl} of linear equations for m right-hand sides, where m is the number of load basis vectors, which depends on the load case of interest. Once we have calculated $\boldsymbol{\xi}_{nm}$, we can compute this part of the sensitivity as:

$$\frac{\partial J^{(k)}}{\partial \widetilde{\sigma}_i^v}\frac{\partial \widetilde{\sigma}_i^v}{\partial z_j} = \boldsymbol{\xi}_{nm}\frac{\partial K_{nl}}{\partial z_j}U_{lm} \tag{5-68}$$

5.4 Numerical Results

This Section presents numerical results obtained through an implementation of the formulations to handle continuously varying load cases described in this work[3]. The numerical results presented here, which consist of the 2D double L-bracket, and the 3D bracket from the General Electric (GE) jet engine bracket challenge [4], verify the effectiveness of the proposed formulation.

5.4.1 Double L-bracket

In this Section we present the results obtained for the double L-bracket domain, displayed in Fig. 5.12(a), considering two loads varying simultaneously with the same angle (case 1 and 2, Sections 5.1.1 and 5.1.2), one fixed load and one load varying in angle (case 3, Section 5.1.3), and two loads varying independently of each other (case 4, Section 5.1.4). The numerical parameters for this problem are displayed in Table 5.1. In Fig. 5.12(b), we display the solution obtained for a fixed load with a load angle of $\theta = -90^o$ to serve as a base for comparison with the other load cases. We also use this solution to exemplify the importance of considering multiple load directions in the design. In Fig. 5.12(c), we display this solution's maximum stress as we vary the load angle. Notice that the stress limit is only satisfied for the angle $\theta = -90^o$ that was considered during the optimization, and for the angle $\theta = -90^o + 180^o = 90^o$. [5] Furthermore, the plot shows that small changes in

[3]The 2D numerical results were obtained through a Matlab implementation, and the 3D numerical results were obtained through the C++/CUDA implementation. The Hardware used to run this problems consists of computer with an i7-4930k CPU at 3.40 GHz and 64 GB of RAM and a NVIDIA GEFORCE GTX 1080 Ti GPU running on a 64-bit operating system

[4]`https://grabcad.com/challenges/ge-jet-engine-bracket-challenge`

[5]The angle $\theta = 90^o$ satisfies the stress limit because of the linearity of the equilibrium equations and the symmetry of von Mises stress formulation, as explained in Section 5.1.2.1, which is also the reason for the symmetry of the plot.

the load angle cause a drastic increase in the maximum stress meaning that this structure is highly susceptible to failure due to small deviations of the load direction.

Table 5.1: Input parameters for the 2D Double L-bracket problem.

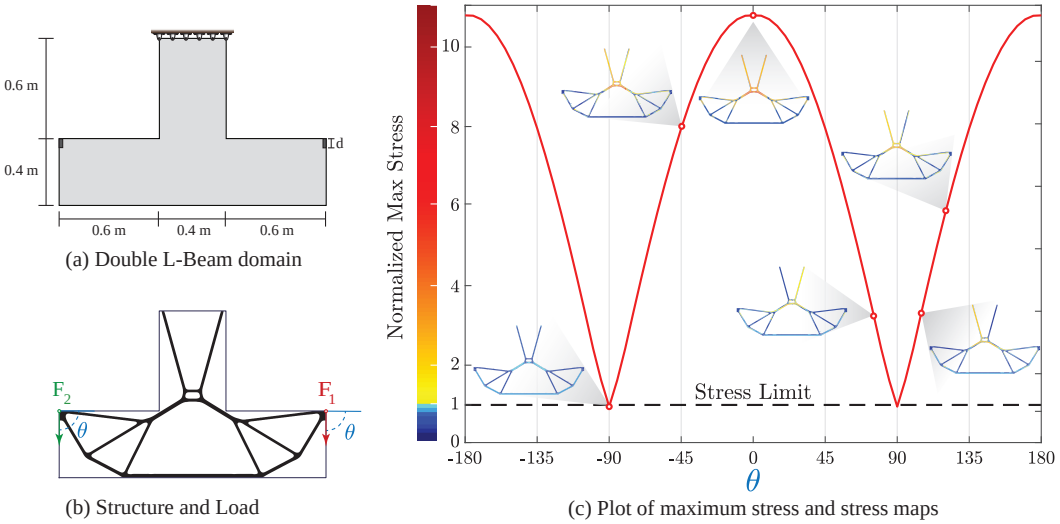
Parameter	Description	Value
E_0	Young's modulus	1 Pa
ν	Poisson's ratio	0.3
σ_{lim}	Stress limit	100 Pa
L	Double L-bracket length	1.6 m
t	Thickness	1 m
F_1	Applied load	1 N
F_2	Applied load	1 N
d	Load distribution length	0.06 m
r	Filter radius	0.015 m
–	Mesh size	220,000 Q4 Elements

5.4.1.1 Double L-beam with two loads varying simultaneously

Figure 5.13 displays the design obtained for the double L-beam considering two loads varying with the same angle θ for different ranges of admissible angle. We can see that, as we increase the range of admissible angles, we obtain designs with more material, and greater complexity. To verify that the structure can withstand the load directions considered in the design, we plot, in Fig. 5.14, the maximum stress of each design as we vary the load direction. We can see that the design that considers a single load direction ($\theta_r = 0$), experiences a drastic increase in the maximum stress as we vary the load angle θ. On the other hand, by imposing a limited angle range as low as 15^o (blue line in plot), we reduce the peak of maximum stress outside the admissible range. We also plot the volume fraction of each design as we increase the range of admissible load angles θ_r, on the top right of Fig. 5.14. As expected, the volume fraction increases as we increase the value of θ_r, because of the extra structural complexity necessary to accommodate the additional load directions.

5.4.1.2 Double L-beam with one fixed load and a load varying in direction

Figure 5.15 displays the design obtained obtained for the double L-beam considering one load varying with angle θ combined with a simultaneous fixed load. We can see that the results present a clear asymmetry regarding the vertical center line, caused by the different load conditions on each side of the double L-beam. To verify the effectiveness of the approach, we plot contours



(a) Double L-Beam domain

(b) Structure and Load

(c) Plot of maximum stress and stress maps

Figure 5.12: Double L-Bracket design considering a single load. (a) Double L-Beam domain geometry; (b) Design optimized for a fixed load angle equal to -90^o for F_1 and F_2 ; (c) Maximum stress of the design in (b) as we vary the load angle θ and the stress map for this structure at selected load angles.

of the maximum stress of each structure as we vary both load directions (Fig. 5.16). In these contour plots, we can see the blue regions, which correspond to regions where the maximum stresses are below the stress limit, align with the load conditions for which we design the structures. We also display the volume fraction of each design in Table 5.2. We choose to display the volume in a table, instead of a plot, as we did for the other cases, because, for this case, we do not have a clear progression of volume, since we vary the fixed load direction, and not the range of admissible angles.

Table 5.2: Volume fractions for the double L-beam designs of Fig. 5.15, considering a load varying in direction and a fixed load with different angles.

	Fixed Angle					
	0^o	30^o	60^o	90^o	120^o	150^o
Volume Fraction	0.235	0.280	0.298	0.302	0.286	0.247

5.4.1.3 Double L-beam with two loads varying independently

Figure 5.17 displays the design obtained obtained for the double L-beam considering two loads varying independently in direction with angles θ_1 and θ_2. This combination of loads generates designs that are completely different from the designs in Fig. 5.13, in which the loads vary simultaneously with the same angle. The difference in design demonstrates the influence of this load case.

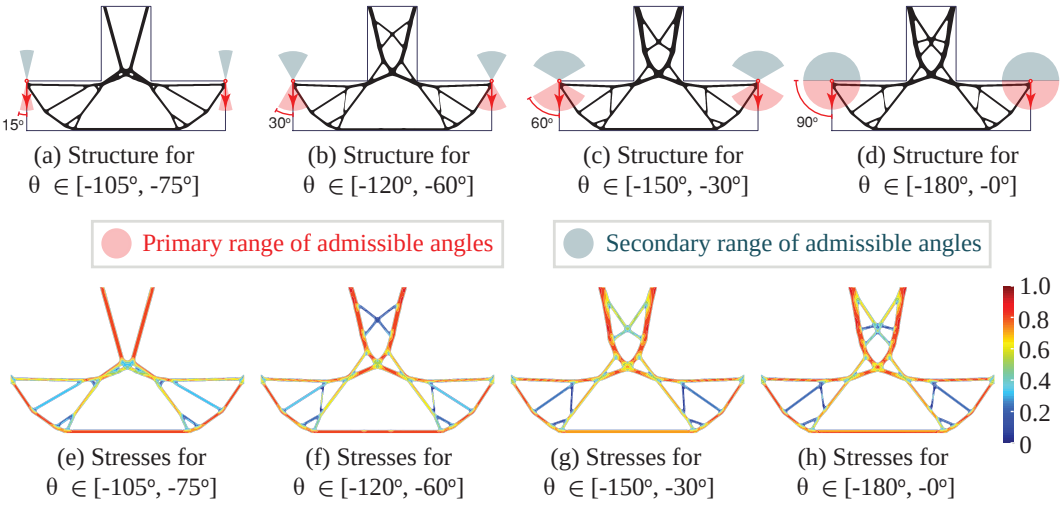


Figure 5.13: Double L-beam solutions with two loads varying simultaneously with an angle θ, with different ranges of admissible angles (a)-(d), and their respective stress map envelope (e)-(h).

To verify the effectiveness of the approach, we plot contours of the maximum stress of each structure as we vary both load directions (Fig. 5.18) in which the white squares in the contour plot represent the set of admissible angles that we consider in the optimization. As mentioned in Section 5.1.4, the approach for loads varying independently is based on an upper bound for the maximum stress, which results in a feasible region (blue region), that is larger than the range of admissible angles considered in the optimization. This is most evident in Fig. 5.18 (f), which displays the maximum stress for the case with a 300^o range of possible load directions. For such case, we can see that the maximum stress is satisfied for every possible combination of load directions. This means that we might possibly design lighter structures that satisfy the stress limit for the range of angles that we considered in the optimization. However, this extra region of feasibility provides a certain degree of safety to the structure. We also display, in Fig. 5.18(h), the volume fraction of the final structure in regards to the range of admissible angles, and we can see a trend of increasing volume fraction as we increase the range of angles considered in the optimization, as expected.

5.4.2 GE Jet Engine Bracket Challenge

In this Section we present the results obtained for the GE Jet Engine Bracket Challenge domain, displayed in Fig. 5.19(a). The domain is subjected to a load case that can vary in 3D, obtained using the combination of two load

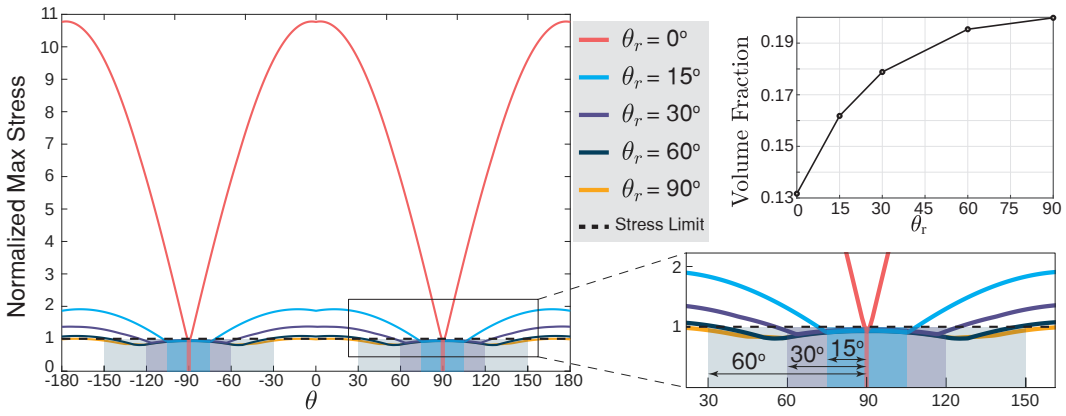


Figure 5.14: Maximum stress of the optimized designs of Fig. 5.13, as we vary the load angle, and the volume fraction of these designs in respect to the range of admissible load angles (θ_r) considered in the optimization.

cases varying independently, contained in perpendicular planes. We proposed this as a Benchmark problem for stress-constrained TO with continuously varying 3D loads. The numerical parameters for this problem are displayed in Table 5.3. Figures 5.12(c)-(g) displays the solutions as we gradually increase the range of admissible load directions from 0^o to 15^o, 30^o, 60^o and 90^o. As we increase the load angle range, we obtain solutions with more volume distributed in a shell-like structure to withstand the extra load directions. The volume fractions of the structures are displayed in Fig. 5.19(b), where we see the expected trend of increasing volume with the increase of the range of admissible angles.

Table 5.3: Input parameters for the 3D GE Jet Engine Bracket Challenge problem.

Parameter	**Description**	**Value**
E_0	Young's modulus	1 Pa
ν	Poisson's ratio	0.25
σ_{lim}	Stress limit	2.5 Pa
F_1	Applied load	100 N
F_2	Applied load	100 N
r	Filter radius	2.5
–	Mesh size	3,727,159 Elements

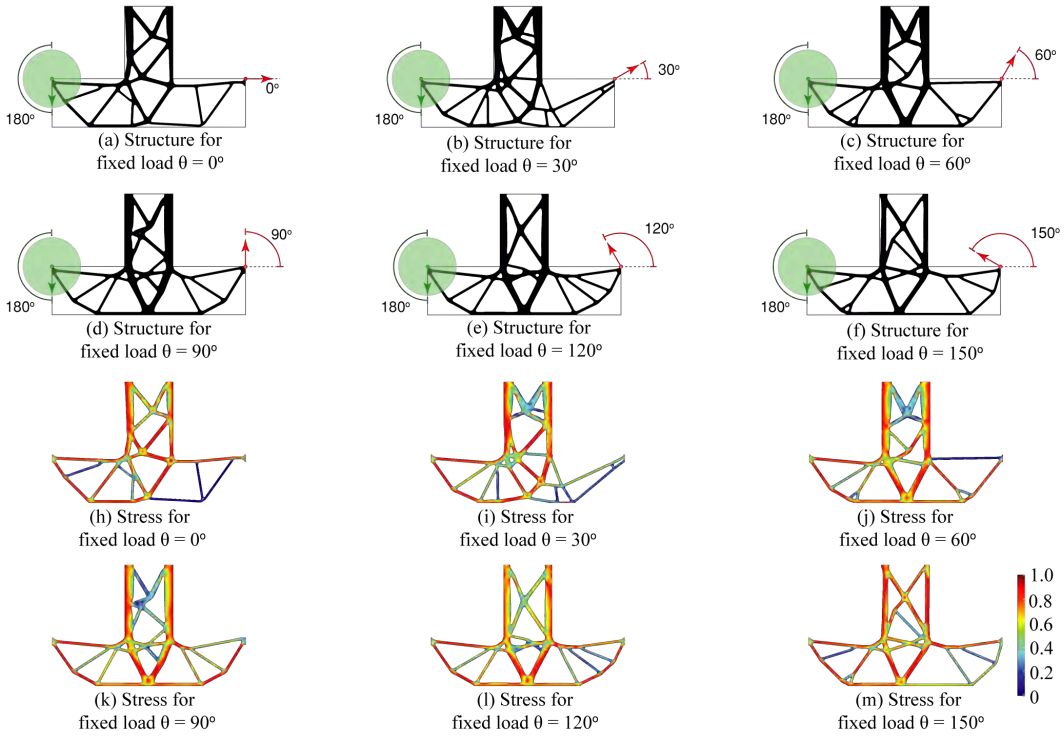
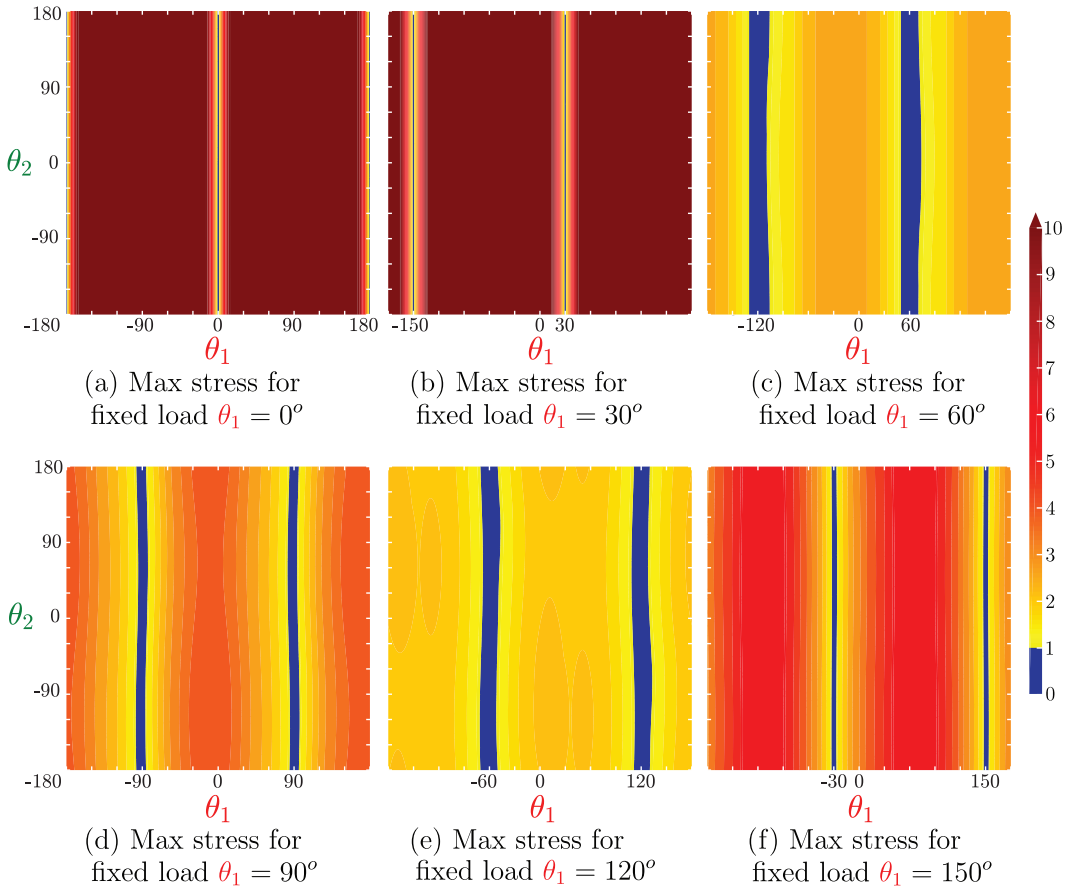


Figure 5.15: Double L-beam solutions with one fixed load (θ_1, red load), and one load varying in direction (θ_2, green load), for different angles of the fixed load (a)-(f), and their respective stress map envelope (g)-(m).



(a) Max stress for fixed load $\theta_1 = 0^o$

(b) Max stress for fixed load $\theta_1 = 30^o$

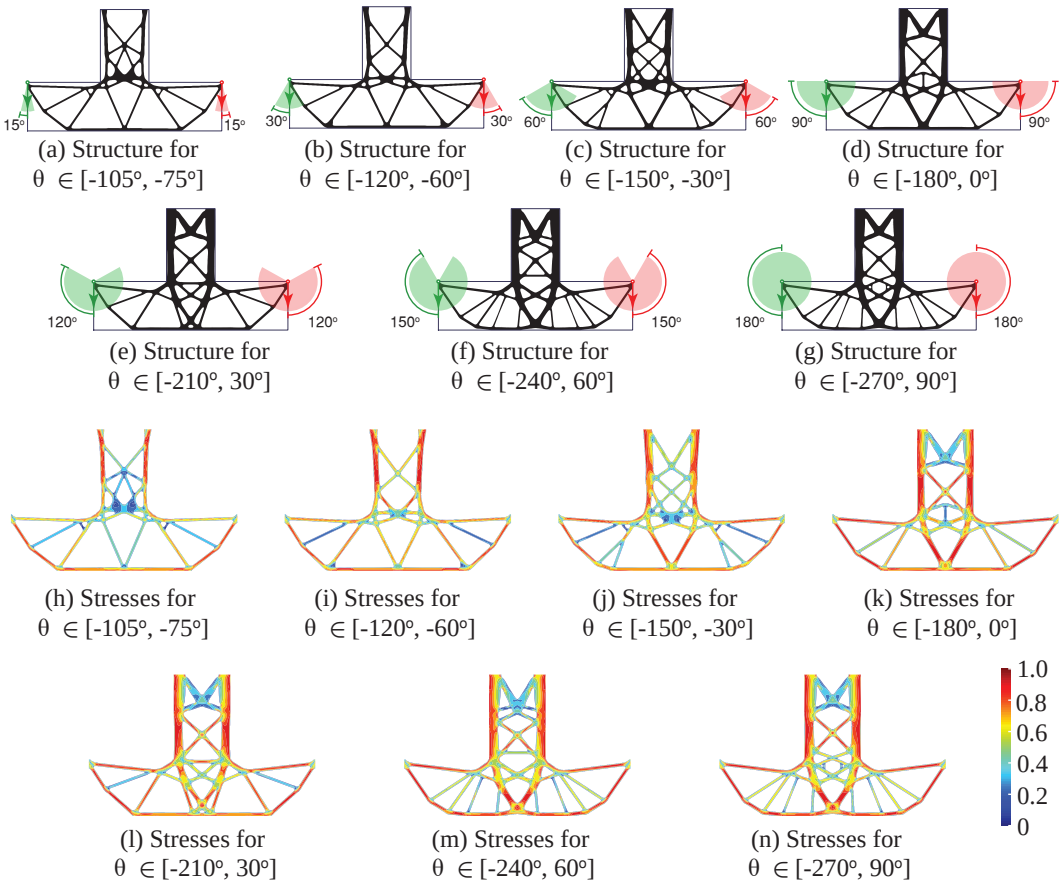
(c) Max stress for fixed load $\theta_1 = 60^o$

(d) Max stress for fixed load $\theta_1 = 90^o$

(e) Max stress for fixed load $\theta_1 = 120^o$

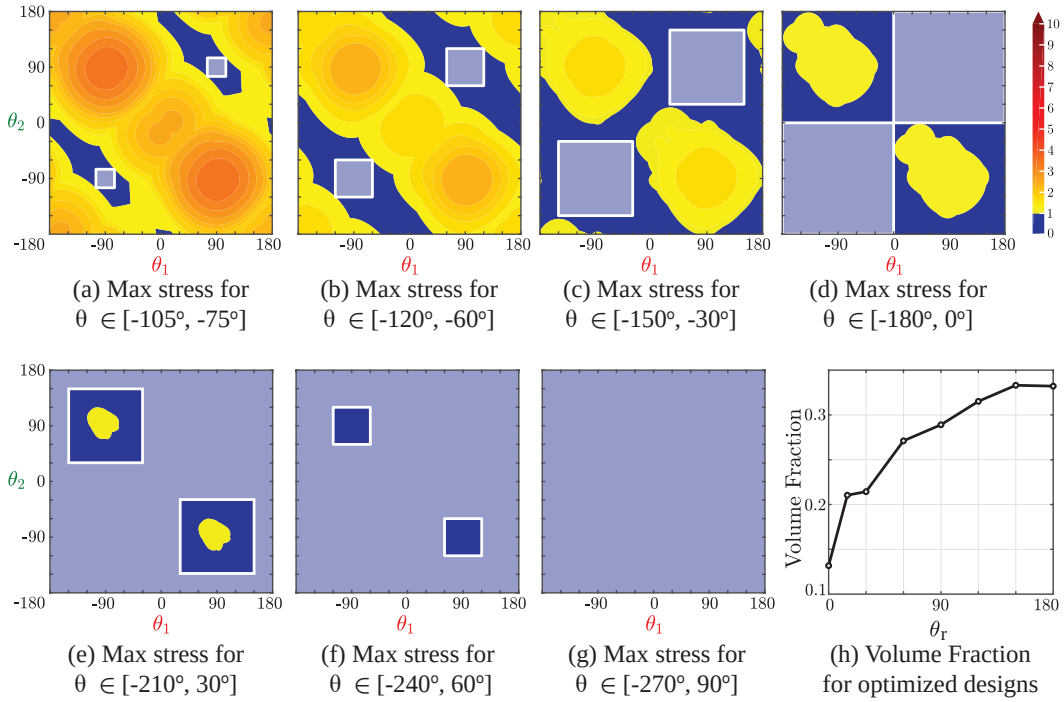
(f) Max stress for fixed load $\theta_1 = 150^o$

Figure 5.16: Maximum stress of the optimized designs of Fig. 5.15, as we vary the angles of both the fixed load (θ_1, red load in Fig. 5.15), and the load varying in direction (θ_2, green load in Fig. 5.15). Blue regions of the contour plot indicate stress below the stress limit.



(a) Structure for θ ∈ [-105°, -75°]
(b) Structure for θ ∈ [-120°, -60°]
(c) Structure for θ ∈ [-150°, -30°]
(d) Structure for θ ∈ [-180°, 0°]
(e) Structure for θ ∈ [-210°, 30°]
(f) Structure for θ ∈ [-240°, 60°]
(g) Structure for θ ∈ [-270°, 90°]
(h) Stresses for θ ∈ [-105°, -75°]
(i) Stresses for θ ∈ [-120°, -60°]
(j) Stresses for θ ∈ [-150°, -30°]
(k) Stresses for θ ∈ [-180°, 0°]
(l) Stresses for θ ∈ [-210°, 30°]
(m) Stresses for θ ∈ [-240°, 60°]
(n) Stresses for θ ∈ [-270°, 90°]

Figure 5.17: Double L-beam solutions with two loads varying independently in direction (θ_1 and θ_2, red and green load, respectively), with different ranges of admissible angles (a)-(g), and their respective stress map envelope (h)-(n).



(a) Max stress for θ ∈ [-105°, -75°]
(b) Max stress for θ ∈ [-120°, -60°]
(c) Max stress for θ ∈ [-150°, -30°]
(d) Max stress for θ ∈ [-180°, 0°]
(e) Max stress for θ ∈ [-210°, 30°]
(f) Max stress for θ ∈ [-240°, 60°]
(g) Max stress for θ ∈ [-270°, 90°]
(h) Volume Fraction for optimized designs

Figure 5.18: Maximum stress of the optimized designs of Fig. 5.17, as we vary the angles of both loads (θ_1 and θ_2, red and green load in Fig. 5.17, respectively). Blue regions of the contour plot indicate stress below the stress limit, and the white squares indicate the range of admissible angles considered in the optimization.

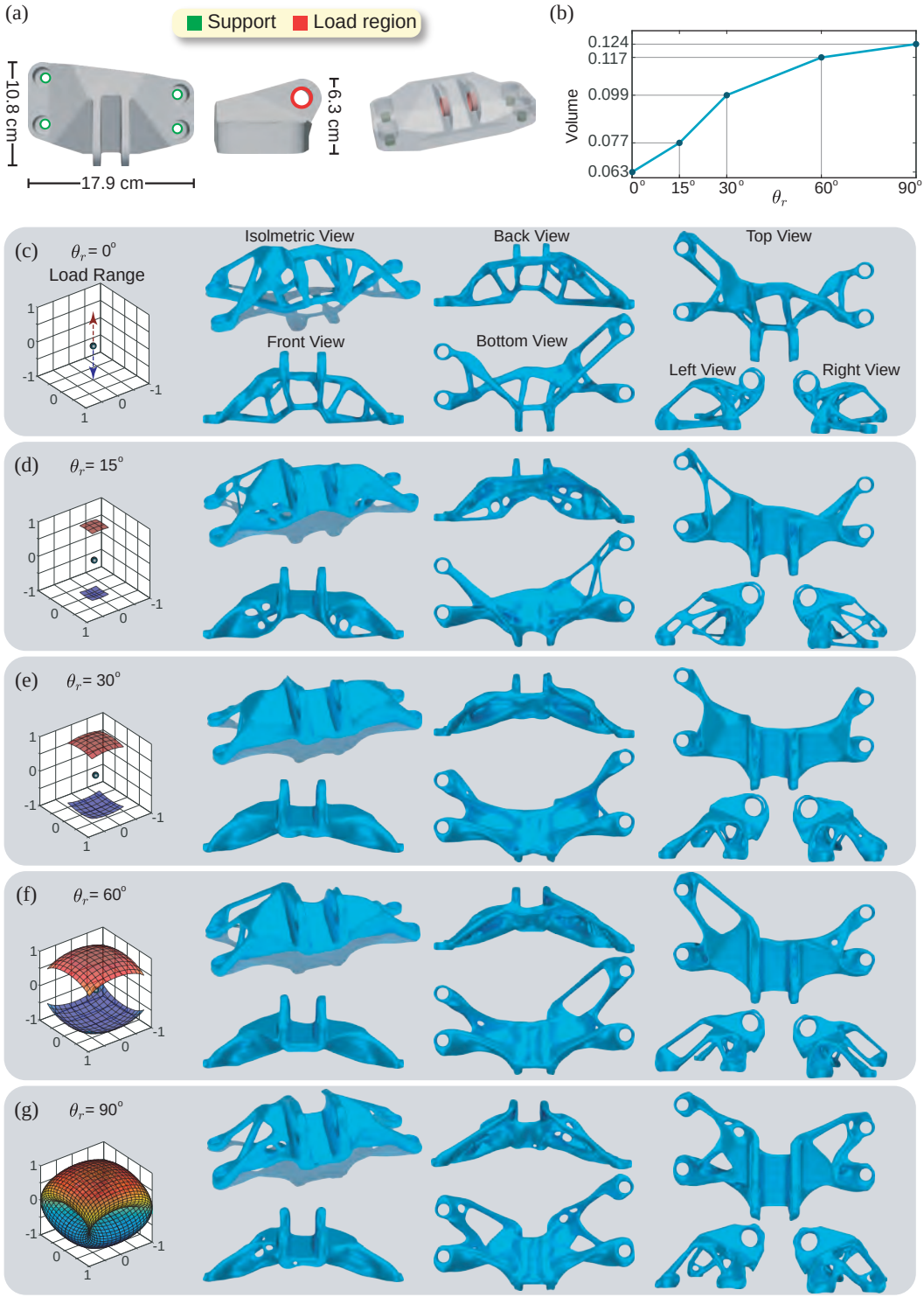


Figure 5.19: GE jet engine challenge problem; (a) design domain with the red regions indicating loading, and green regions indicating support. (b) Volume fraction of the GE jet engine challenge problem solutions displayed in (c)-(g) as we vary the load angle range, θ_r. (c)-(g) Isometric, top, and bottom views, of the optimized structures considering 0^o, 15^o, 30^o, 60^o and 90^o load angle range.

6
Conclusions

Stress-constrained topology optimization leads to designs that are more suitable to engineering applications, because it incorporates material limitations that are essential for real-life applications in the design process. In this work, we expanded on the the existing stress-constrained TO techniques by developing large-scale, GPU-based parallel computing techniques, and an efficient and consistent technique to consider a continuous range of load angles in the TO formulation. To handle the large number of stress constraints, we adopt an AL-based technique that was able to efficiently solve problems with over 46 million constraints (Double-Decked Bridge problem, Fig. 4.9, and L-Beam problem, Fig. 4.8). For the formulation of the stress constraint, and to address the associated singular optima phenomena, we present a modified piece-wise vanishing stress constraint that is more stable than previous formulations.

The AL-based stress-constrained TO high computational cost was addressed by the development of a C++/CUDA GPU parallel computing framework, that allowed the solution of large-scale problems. The large-scale stress-constrained TO lead to the development of:

- a **sequential filtering** technique with a low memory footprint;
- an **adaptive gradient-based optimization algorithm** that can easily be implemented in parallel;
- and a **parallel GPU assembly-free preconditioned conjugated gradient solver** for the FEA with and optimized local stiffness matrix product that can efficiently handle large-scale problems with unstructured meshes.

The optimization of the local stiffness matrix product was accomplished by the creation of a new optimization problem, the Set Collapsing Problem, which was mathematically proved to be NP-Hard. The problem was then solved by a specifically tailored Branch-and-Bound scheme. The solution reduced the number of floating point operations by *more than half*. These techniques allowed us to solve several numerical examples (L-Beam, Double-decked bridge, the Victoria-Regia, and the GE jet engine bracket) with over 47 million constraints, demonstrating the effectiveness of the framework. The numerical

examples provided insight on large-scale optimized structures, and served as base of comparison for the efficiency of the GPU implementation.

Continuing in our path to achieve more practical optimized design, we considered a continuous range of load angles in stress-constrained TO that more closely resemble realistic load scenarios. The technique to incorporate the continuously varying load is based on a worst-case scenario approach in which we find analytical solutions that bound the maximum stress caused by a range of admissible load angles. We developed analytical solutions for:

- a single load varying in a limited range of angles;
- a load varying in direction plus a fixed load;
- two or more loads varying independently in direction;
- a load varying in 3D in a limited range of angles.

The technique is extremely flexible, supporting a wide variety of load conditions, including loads that vary in intensity. This wide variety of load conditions provides engineers with the tools to design structures that are a better fit to practical requirements. As demonstrated by the numerical examples, designs optimized considering only one fixed load are extremely sensitive to variations in the load angle, and small variations in the load direction, which are common in real life situations, can cause severely high stresses, meaning that the structure might fail in practice. On the other hand, considering a range of load angles, instead of a single fixed load, leads to significantly different optimized structure that are closer to realistic engineering components, and are robust under variations of the load direction. Additionally, the technique developed here can be used for any TO problem with objective, and/or constraint functions that have a primarily bi-linear form (such as compliance), and linear state equations (see Appendix B for a compliance minimization formulation).

6.1 Suggestions for Future Work

In this Section we discuss possible future research, and extensions of the techniques developed in this work:

- **Multi-GPU**: The large-scale parallel GPU implementation developed in this work is limited to a single device (i.e. a single GPU), which limits the potential of the implementation by the capabilities of the GPUs available on the market. By incorporating Multi-GPU support into the implementation we could possibly efficiently solve larger problems.

However, a Multi-GPU framework presents many challenges due to the time consuming exchange of information between GPUs. Ideally, this communication between GPUs should be minimized, and the appropriate Hardware is necessary (NV-Link) to make this communication as efficient as possible.

- **Multi-Grid**: The PCG method performance is heavily dependent on the condition number of the input matrix. Unfortunately, the condition number of the stiffness matrix in TO problems quickly degenerates as the optimization progresses. This leads to significant increase in computational time for the solution of the equilibrium equations. To circumvent this issue, the use of preconditioners can greatly improve the condition number of the underlying problem. The Multi-Grid preconditioner is, potentially, the most promising option in the parallel GPU framework, because it has a high degree of parallelization, and a low memory footprint. The implementation of a Multi-Grid PCG could greatly improve the efficiency of the current framework.
- **Closer Upper bound for 3D varying Load**: The upper bound of the stress used for the 3D varying load case could potentially be improved by the use of approximations of the trigonometric functions. A better upper bound would improve the accuracy of the TO solution, and enable us to obtain lighter structures that more closely satisfy the stress constraints.
- **Loads varying location**: In real life structures, loads can vary, not only in direction, but also in location, e.g. a truck crossing a bridge imposes a load that transverse the whole length of said bridge. The development of a worst-case stress analytical solution for a load varying in location would greatly improve the capabilities of TO as a design tool for engineers.

Bibliography

[1] **cublas library**. `https://developer.nvidia.com/cublas`. Accessed: 2022-04-11.

[2] **cusparse library**. `https://developer.nvidia.com/cusparse`. Accessed: 2022-04-11.

[3] AAGE, N.; LAZAROV, B. S.. **Parallel framework for topology optimization using the method of moving asymptotes**. Structural and Multidisciplinary Optimization, 47(4):493–505, 2013.

[4] AAGE, N.; LAZAROV, B. S.. **Parallel framework for topology optimization using the method of moving asymptotes**. Structural and multidisciplinary optimization, 47(4):493–505, 2013.

[5] AAGE, N.; POULSEN, T. H.; GERSBORG-HANSEN, A. ; SIGMUND, O.. **Topology optimization of large scale stokes flow problems**. Structural and Multidisciplinary Optimization, 35(2):175–180, 2008.

[6] AAGE, N.; ANDREASSEN, E.; LAZAROV, B. S. ; SIGMUND, O.. **Gigavoxel computational morphogenesis for structural design**. Nature, 550 (7674):84–86, 2017.

[7] ACHTZIGER, W.; KANZOW, C.. **Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications**. Mathematical Programming, 114(1):69–99, Jul 2008.

[8] ACHTZIGER, W.; HOHEISEL, T. ; KANZOW, C.. **A smoothing-regularization approach to mathematical programs with vanishing constraints**. Computational Optimization and Applications, 55(3):733–767, Jul 2013.

[9] ANDREASSEN, E.; CLAUSEN, A.; SCHEVENELS, M.; LAZAROV, B. S. ; SIGMUND, O.. **Efficient topology optimization in matlab using 88 lines of code**. Structural and Multidisciplinary Optimization, 43(1):1–16, 2011.

[10] ASADPOURE, A.; TOOTKABONI, M. ; GUEST, J. K.. **Robust topology optimization of structures with uncertainties in stiffness–application to truss structures**. Computers & Structures, 89(11-12):1131–1141, 2011.

[11] AUGARDE, C.; RAMAGE, A. ; STAUDACHER, J.. **An element-based displacement preconditioner for linear elasticity problems**. Computers & structures, 84(31-32):2306–2315, 2006.

[12] BAANDRUP, M.; SIGMUND, O.; POLK, H. ; AAGE, N.. **Closing the gap towards super-long suspension bridges using computational morphogenesis**. Nature Communications, 11(2735), 2020.

[13] BELL, N.; HOBEROCK, J.. **Thrust: A productivity-oriented library for cuda**. In: GPU COMPUTING GEMS JADE EDITION, p. 359–371. Elsevier, 2012.

[14] BEN-TAL, A.; NEMIROVSKI, A.. **Robust optimization – methodology and applications**. Mathematical Programming, 92(3):453–480, May 2002.

[15] BENDSØE, M. P.. **Optimization of Structural Topology, Shape, and Material**. Springer, 1995.

[16] BENDSØE, M. P.; KIKUCHI, N.. **Generating optimal topologies in structural design using a homogenization method**. Computer Methods in Applied Mechanics and Engineering, 71(2):197–224, 1988.

[17] BENDSØE, M. P.; SIGMUND, O.. **Topology optimization: Theory, methods and applications**. Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-662-05086-6.

[18] BENDSOE, M. P.; SIGMUND, O.. **Topology optimization: theory, methods, and applications**. Springer Science & Business Media, 2003.

[19] BERTSEKAS, D. P.. **Nonlinear Programming**. Athena Scientific, 2nd edition, 1999.

[20] BERTSEKAS, D. P.. **Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)**. Athena Scientific, 1 edition, 1996.

[21] BLETZINGER, K.-U.. **Extended method of moving asymptotes based on second-order information**. Structural optimization, 5(3):175–183, 1993.

[22] BORRVALL, T.; PETERSSON, J.. **Large-scale topology optimization in 3D using parallel computing**. Computer Methods in Applied Mechanics and Engineering, 190(46):6201–6229, 2001.

[23] BOURDIN, B.. **Filters in topology optimization**. International Journal for Numerical Methods in Engineering, 50(9):2143–2158, 2001.

[24] BOURDIN, B.. **Filters in topology optimization**. International journal for numerical methods in engineering, 50(9):2143–2158, 2001.

[25] BRUGGI, M.. **On an alternative approach to stress constraints relaxation in topology optimization**. Structural and Multidisciplinary Optimization, 36(2):125–141, 2008.

[26] BRUNS, T. E.; TORTORELLI, D. A.. **Topology optimization of non-linear elastic structures and compliant mechanisms**. Computer methods in applied mechanics and engineering, 190(26-27):3443–3459, 2001.

[27] CHALLIS, V. J.; ROBERTS, A. P. ; GROTOWSKI, J. F.. **High resolution topology optimization using graphics processing units (gpus)**. Structural and Multidisciplinary Optimization, 49(2):315–325, 2014.

[28] CHENG, G. D.; GUO, X.. **ε-relaxed approach in structural topology optimization**. Structural Optimization, 13(4):258–266, 1997.

[29] CHENG, G. D.; JIANG, Z.. **Study on topology optimization with stress constraints**. Engineering Optimization, 20(2):129–148, 1992.

[30] CHOI, K. K.; KIM, N.-H.. **Structural sensitivity analysis and optimization 1: linear systems**. Springer Science & Business Media, 2004.

[31] CHRISTENSEN, P.; KLARBRING, A.. **An Introduction to Structural Optimization**. Solid Mechanics and Its Applications. Springer Netherlands, 2008.

[32] CLAUSEN, J.. **Branch and bound algorithms-principles and examples**. Department of Computer Science, University of Copenhagen, p. 1–30, 1999.

[33] CSÉBFALVI, A.. **Structural optimization under uncertainty in loading directions: Benchmark results**. Advances in Engineering Software, 120: 68 – 78, 2018. Civil-Comp - Part 2.

[34] CUELLAR, N.; PEREIRA, A.; MENEZES, I. F. ; CUNHA, A.. **Non-intrusive polynomial chaos expansion for topology optimization using**

polygonal meshes. Journal of the Brazilian Society of Mechanical Sciences and Engineering, 40(12):1–18, 2018.

[35] DA SILVA, G. A.; BECK, A. T. ; SIGMUND, O.. **Stress-constrained topology optimization considering uniform manufacturing uncertainties**. Computer Methods in Applied Mechanics and Engineering, 344:512–537, 2019.

[36] DA SILVA, G.; BECK, A. T. ; CARDOSO, E. L.. **Topology optimization of continuum structures with stress constraints and uncertainties in loading**. International journal for numerical methods in engineering, 113(1): 153–178, 2018.

[37] DE STURLER, E.; LE, C.; WANG, S. ; PAULINO, G.. **Large scale topology optimization using preconditioned krylov subspace recycling and continuous approximation of material distribution**. AIP Conference Proceedings, 973(February 2008):279–284, 2008.

[38] DÍAZ, A. R.; BENDSØE, M. P.. **Shape optimization of structures for multiple loading conditions using a homogenization method**. Structural Optimization, 4(1):17–22, 1992.

[39] DUARTE, L. S.; CELES, W.; PEREIRA, A.; M MENEZES, I. F. ; PAULINO, G. H.. **Polytop++: an efficient alternative for serial and parallel topology optimization on cpus & gpus**. Structural and Multidisciplinary Optimization, 52(5):845–859, 2015.

[40] DUNNING, P. D.; KIM, H. A. ; MULLINEUX, G.. **Introducing loading uncertainty in topology optimization**. AIAA journal, 49(4):760–768, 2011.

[41] DUYSINX, P.; BENDSØE, M. P.. **Topology optimization of continuum structures with local stress constraints**. International Journal for Numerical Methods in Engineering, 43(8):1453–1478, 1998.

[42] DUYSINX, P.; SIGMUND, O.. **New developments in handling stress constraints in optimal material distribution**. In: PROCEEDINGS OF THE 7TH AIAA/USAF/NASA/ISSMO SYMPOSIUM ON MULTIDISCIPLINARY ANALYSIS AND OPTIMIZATION, volumen 1, p. 1501–1509, 1998.

[43] EMMENDOERFER JR., H.; FANCELLO, E. A.. **A level set approach for topology optimization with local stress constraints**. International Journal for Numerical Methods in Engineering, 99(2):129–156, 2014.

[44] EMMENDOERFER JR., H.; FANCELLO, E. A.. **Topology optimization with local stress constraint based on level set evolution via reaction-diffusion**. Computer Methods in Applied Mechanics and Engineering, 305: 62–88, 2016.

[45] EULER, L.. **Elements of Algebra**. J. Johnson and Company, 2 edition, 1810.

[46] EVGRAFOV, A.; RUPP, C. J.; MAUTE, K. ; DUNN, M. L.. **Large-scale parallel topology optimization using a dual-primal substructuring solver**. Structural and Multidisciplinary Optimization, 36(4):329–345, 2008.

[47] FAUCETTE, W. M.. **A geometric interpretation of the solution of the general quartic polynomial**. The American Mathematical Monthly, 103 (1):51–57, 1996.

[48] GAREY, M. R.; JOHNSON, D. S.. **Computers and intractability**, volumen 174. freeman San Francisco, 1979.

[49] GIRALDO-LONDOÑO, O.; PAULINO, G. H.. **A unified approach for topology optimization with local stress constraints considering various failure criteria: von mises, drucker–prager, tresca, mohr–coulomb, bresler–pister and willam–warnke**. Proceedings of the Royal Society A, 476(2238):20190861, 2020.

[50] GIRALDO-LONDOÑO, O.; AGUILÓ, M. A. ; PAULINO, G. H.. **Local stress constraints in topology optimization of structures subjected to arbitrary dynamic loads: a stress aggregation-free approach**. Structural and Multidisciplinary Optimization, 64(6):3287–3309, 2021.

[51] GOLUB, G. H.; VAN LOAN, C. F.. **Matrix computations**. JHU press, 2013.

[52] GRAMA, A.; KUMAR, V.; GUPTA, A. ; KARYPIS, G.. **Introduction to parallel computing**. Pearson Education, 2003.

[53] GUEST, J. K.; IGUSA, T.. **Structural optimization under uncertain loads and nodal locations**. Computer Methods in Applied Mechanics and Engineering, 198(1):116 – 124, 2008. Computational Methods in Optimization Considering Uncertainties.

[54] GUO, X.; ZHANG, W. ; ZHANG, L.. **Robust structural topology optimization considering boundary uncertainties**. Computer Methods in Applied Mechanics and Engineering, 253:356 – 368, 2013.

[55] HASSANI, B.; HINTON, E.. **A review of homogenization and topology optimization iii—topology optimization using optimality criteria**. Computers & structures, 69(6):739–756, 1998.

[56] HERRERO-PÉREZ, D.; CASTEJÓN, P. J. M.. **Multi-gpu acceleration of large-scale density-based topology optimization**. Advances in Engineering Software, 157:103006, 2021.

[57] HOHEISEL, T.; KANZOW, C.. **Stationary conditions for mathematical programs with vanishing constraints using weak constraint qualifications**. Journal of Mathematical Analysis and Applications, 337(1):292 – 310, 2008.

[58] HOLMBERG, E.; TORSTENFELT, B. ; KLARBRING, A.. **Stress constrained topology optimization**. Structural and Multidisciplinary Optimization, 48(1):33–47, 2013.

[59] HOLMBERG, E.; THORE, C.-J. ; KLARBRING, A.. **Game theory approach to robust topology optimization with uncertain loading**. Structural and Multidisciplinary Optimization, 55(4):1383–1397, 2017.

[60] HU, X.; LI, Z.; BAO, R.; CHEN, W. ; WANG, H.. **An adaptive method of moving asymptotes for topology optimization based on the trust region**. Computer Methods in Applied Mechanics and Engineering, 393: 114202, 2022.

[61] JALALPOUR, M.; GUEST, J. K. ; IGUSA, T.. **Reliability-based topology optimization of trusses with stochastic stiffness**. Structural Safety, 43: 41 – 49, 2013.

[62] JAMES, K.; LEE, E. ; MARTINS, J.. **Stress-based topology optimization using an isoparametric level set method**. Finite Elements in Analysis and Design, 58:20 – 30, 2012.

[63] KANNO, Y.; TAKEWAKI, I.. **Robustness analysis of trusses with separable load and structural uncertainties**. International Journal of Solids and Structures, 43(9):2646 – 2669, 2006.

[64] KARP, R. M.. **Reducibility among combinatorial problems**. In: COMPLEXITY OF COMPUTER COMPUTATIONS, p. 85–103. Springer, 1972.

[65] KAWAMOTO, A.; MATSUMORI, T.; YAMASAKI, S.; NOMURA, T.; KONDOH, T. ; NISHIWAKI, S.. **Heaviside projection based topology optimization by a pde-filtered scalar function**. Structural and Multidisciplinary Optimization, 44(1):19–24, 2011.

[66] KIM, T. S.; KIM, J. E. ; KIM, Y. Y.. **Parallelized structural topology optimization for eigenvalue problems**. International Journal of Solids and Structures, 41(9-10):2623–2641, 2004.

[67] KIRSCH, U.. **Optimal topologies in truss structures**. Computer Methods in Applied Mechanics and Engineering, 72(1):15–28, 1989.

[68] KIRSCH, U.. **On singular topologies in optimum structural design**. Structural Optimization, 2(3):133–142, 1990.

[69] KIRSCH, U.; TAYE, S.. **On optimal topology of grillage structures**. Engineering with Computers, 1:229–243, 1986.

[70] LABANDA, R.. **Mathematical programming methods for large-scale topology optimization problems PhD Thesis**. PhD thesis, 2015.

[71] LAWLER, E. L.; WOOD, D. E.. **Branch-and-bound methods: A survey**. Operations research, 14(4):699–719, 1966.

[72] LAZAROV, B. S.; SIGMUND, O.. **Filters in topology optimization based on helmholtz-type differential equations**. International Journal for Numerical Methods in Engineering, 86(6):765–781, 2011.

[73] LAZAROV, B. S.; SCHEVENELS, M. ; SIGMUND, O.. **Topology optimization with geometric uncertainties by perturbation techniques**. International Journal for Numerical Methods in Engineering, 90(11):1321–1336, 2012.

[74] LE, C.; NORATO, J.; BRUNS, T.; HA, C. ; TORTORELLI, D.. **Stress-based topology optimization for continua**. Structural and Multidisciplinary Optimization, 41(4):605–620, 2010.

[75] LEADER, M. K.; CHIN, T. W. ; KENNEDY, G. J.. **High-resolution topology optimization with stress and natural frequency constraints**. AIAA Journal, 57(8):3562–3578, 2019.

[76] LEE, E.; JAMES, K. A. ; MARTINS, J. R. R. A.. **Stress-constrained topology optimization with design-dependent loading**. Structural and Multidisciplinary Optimization, 46(5):647–661, 2012.

[77] LIAN, H.; CHRISTIANSEN, A. N.; TORTORELLY, D. A. ; SIGMUND, O.. **Combined shape and topology optimization for minimization of maximal von mises stress**. Structural and Multidisciplinary Optimization, 55(5):1541–1557, 2017.

[78] LIU, J.; WEN, G.. **Continuum topology optimization considering uncertainties in load locations based on the cloud model**. Engineering Optimization, 50(6):1041–1060, 2018.

[79] LÓGÓ, J.; BALOGH, B. ; PINTÉR, E.. **Topology optimization considering multiple loading**. Computers & Structures, 207:233 – 244, 2018. CIVIL-COMP 2017.

[80] LOPES, C. G.; DOS SANTOS, R. B. ; NOVOTNY, A. A.. **Topological derivative-based topology optimization of structures subject to multiple load-cases**. Latin American Journal of Solids and Structures, 12 (5):834–860, 2015.

[81] LUO, Y.; ZHOU, M.; WANG, M. Y. ; DENG, Z.. **Reliability based topology optimization for continuum structures with local failure constraints**. Computers & Structures, 143:73 – 84, 2014.

[82] MAHDAVI, A.; BALAJI, R.; FRECKER, M. ; MOCKENSTURM, E. M.. **Topology optimization of 2D continua for minimum compliance using parallel computing**. Structural and Multidisciplinary Optimization, 32(2):121–132, 2006.

[83] MARTÍNEZ-FRUTOS, J.; HERRERO-PÉREZ, D.. **Large-scale robust topology optimization using multi-gpu systems**. Computer Methods in Applied Mechanics and Engineering, 311:393–414, 2016.

[84] MARTÍNEZ-FRUTOS, J.; HERRERO-PÉREZ, D.. **Gpu acceleration for evolutionary topology optimization of continuum structures using isosurfaces**. Computers & Structures, 182:119–136, 2017.

[85] MARTÍNEZ-FRUTOS, J.; MARTÍNEZ-CASTEJÓN, P. J. ; HERRERO-PÉREZ, D.. **Efficient topology optimization using gpu computing with multilevel granularity**. Advances in Engineering Software, 106:47–62, 2017.

[86] MICHELL, A. G. M.. **Lviii. the limits of economy of material in frame-structures**. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 8(47):589–597, 1904.

[87] MUKHERJEE, S.; LU, D.; RAGHAVAN, B.; BREITKOPF, P.; DUTTA, S.; XIAO, M. ; ZHANG, W.. **Accelerating large-scale topology optimization: State-of-the-art and challenges**. Archives of Computational Methods in Engineering, 28(7):4549–4571, 2021.

[88] NOCEDAL, J.; WRIGHT, S. J.. **Numerical Optimization**. Springer, 2 edition, 2006.

[89] OLSSON, D. M.; NELSON, L. S.. **The nelder-mead simplex procedure for function minimization**. Technometrics, 17(1):45–51, 1975.

[90] PARIS, J.; NAVARRINA, F.; COLOMINAS, I. ; CASTELEIRO, M.. **Block aggregation of stress constraints in topology optimization of structures**. Advances in Engineering Software, 41(3):433–441, 2010.

[91] PEREIRA, J. T.; FANCELLO, E. A. ; BARCELLOS, C. S.. **Topology optimization of continuum structures with material failure constraints**. Structural and Multidisciplinary Optimization, 26(1-2):50–66, 2004.

[92] PICELLI, R.; TOWNSEND, S.; BRAMPTON, C.; NORATO, J. ; KIM, H. A.. **Stress-based shape and topology optimization with the level set method**. Computer Methods in Applied Mechanics and Engineering, 329: 1–23, 2018.

[93] RAM, L.; SHARMA, D.. **Evolutionary and gpu computing for topology optimization of structures**. Swarm and evolutionary computation, 35: 1–13, 2017.

[94] RAMÍREZ-GIL, F. J.; SILVA, E. C. N. ; MONTEALEGRE-RUBIO, W.. **Topology optimization design of 3d electrothermomechanical actuators by using gpu as a co-processor**. Computer Methods in Applied Mechanics and Engineering, 302:44–69, 2016.

[95] REDDY, J. N.. **Introduction to the finite element method**. McGraw-Hill Education, 2019.

[96] ROKICKI, J.; OTHERS. **Adjoint lattice boltzmann for topology optimization on multi-gpu architecture**. Computers & Mathematics with Applications, 71(3):833–848, 2016.

[97] ROZVANY, G. I. N.. **On design-dependent constraints and singular topologies**. Structural and Multidisciplinary Optimization, 21(2):164–172, 2001.

[98] ROZVANY, G. I.. **Structural design via optimality criteria: the Prager approach to structural optimization**, volumen 8. Springer Science & Business Media, 2012.

[99] SARKISIAN, M.. **Designing tall buildings: Structure as architecture**. 2012.

[100] SCHMIDT, S.; SCHULZ, V.. **A 2589 line topology optimization code written for the graphics card**. Computing and Visualization in Science, 14(6):249–256, 2011.

[101] SCHMIDT, S.; SCHULZ, V.. **A 2589 line topology optimization code written for the graphics card**. Computing and Visualization in Science, 14(6):249–256, 2011.

[102] SENHORA, F. V.; GIRALDO-LONDONO, O.; MENEZES, I. F. ; PAULINO, G. H.. **Topology optimization with local stress constraints: a stress aggregation-free approach**. Structural and Multidisciplinary Optimization, 62(4):1639–1668, 2020.

[103] SHARMA, A.; MAUTE, K.. **Stress-based topology optimization using spatial gradient stabilized XFEM**. Structural and Multidisciplinary Optimization, 57(1):17–38, 2018.

[104] SHEWCHUK, J. R.; OTHERS. **An introduction to the conjugate gradient method without the agonizing pain**, 1994.

[105] SHUN, W.; DE, S. E. ; H., P. G.. **Large-scale topology optimization using preconditioned Krylov subspace methods with recycling**. International Journal for Numerical Methods in Engineering, 69(12):2441–2468, 2008.

[106] SIGMUND, O.. **Manufacturing tolerant topology optimization**. Acta Mechanica Sinica, 25(2):227–239, 2009.

[107] SIGMUND, O.. **Materials with prescribed constitutive parameters: an inverse homogenization problem**. International Journal of Solids and Structures, 31(17):2313–2329, 1994.

[108] SPIVAK, M.. **Calculus on manifolds. A modern approach to classical theorems of advanced calculus**. W. A. Benjamin, Inc., New York-Amsterdam, 1965.

[109] SVANBERG, K.. **The method of moving asymptotes (mma) with some extensions**. In: OPTIMIZATION OF LARGE STRUCTURAL SYSTEMS, p. 555–566. Springer, 1993.

[110] SVANBERG, K.. **The method of moving asymptotes—a new method for structural optimization**. International journal for numerical methods in engineering, 24(2):359–373, 1987.

[111] SVANBERG, K.. **Mma and gcmma, versions september 2007**. Optimization and systems theory, 104, 2007.

[112] SVED, G.; GINOS, Z.. **Structural optimization under multiple loading**. International Journal of Mechanical Sciences, 10(10):803–805, 1968.

[113] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A. ; MENEZES, I. F. M.. **`PolyTop`: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes**. Structural and Multidisciplinary Optimization, 45(3):329–357, 2012.

[114] THORE, C. J.; HOLMBERG, E. ; KLARBRING, A.. **A general framework for robust topology optimization under load-uncertainty including stress constraints**. Computer Methods in Applied Mechanics and Engineering, 319:1 – 18, 2017.

[115] TOOTKABONI, M.; ASADPOURE, A. ; GUEST, J. K.. **Topology optimization of continuum structures under uncertainty–a polynomial chaos approach**. Computer Methods in Applied Mechanics and Engineering, 201:263–275, 2012.

[116] TREFETHEN, L. N.; BAU III, D.. **Numerical linear algebra**, volumen 50. Siam, 1997.

[117] VEMAGANTI, K.; LAWRENCE, W. E.. **Parallel methods for optimality criteria-based topology optimization**. Computer Methods in Applied Mechanics and Engineering, 194(34-35):3637–3667, 2005.

[118] WADBRO, E.; BERGGREN, M.. **Megapixel topology optimization on a graphics processing unit**. SIAM review, 51(4):707–721, 2009.

[119] WALLIN, M.; IVARSSON, N.; AMIR, O. ; TORTORELLI, D.. **Consistent boundary conditions for pde filter regularization in topology optimization**. Structural and Multidisciplinary Optimization, 62(3):1299–1311, 2020.

[120] WANG, F.; LAZAROV, B. S. ; SIGMUND, O.. **On projection methods, convergence and robust formulations in topology optimization**. Structural and Multidisciplinary Optimization, 43(6):767–784, 2011.

[121] WU, J.; DICK, C. ; WESTERMANN, R.. **A system for high-resolution topology optimization**. IEEE transactions on visualization and computer graphics, 22(3):1195–1208, 2015.

[122] XIA, Q.; SHI, T.; LIU, S. ; WANG, M. Y.. **A level set solution to the stress-based structural shape and topology optimization**. Computers & Structures, 90:55–64, 2012.

[123] XIE, Y.; STEVEN, G. P.. **Optimal design of multiple load case structures using an evolutionary procedure**. Engineering computations, 1994.

[124] YANG, R. J.; CHEN, C. J.. **Stress-based topology optimization**. Structural Optimization, 12(2):98–105, 1996.

[125] YOUNG, V.; QUERIN, O. M.; STEVEN, G. ; XIE, Y.. **3D and multiple load case bi-directional evolutionary structural optimization (beso)**. Structural optimization, 18(2-3):183–192, 1999.

[126] ZEGARD, T.; PAULINO, G. H.. **Bridging topology optimization and additive manufacturing**. Structural and Multidisciplinary Optimization, 53(1):175–192, 2016.

[127] ZEGARD, T.; PAULINO, G. H.. **Toward gpu accelerated topology optimization on unstructured meshes**. Structural and multidisciplinary optimization, 48(3):473–485, 2013.

[128] ZEGARD, T.; PAULINO, G. H.. **Bridging topology optimization and additive manufacturing**. Structural and Multidisciplinary Optimization, 53(1):175–192, 2016.

[129] ZHANG, W.; KANG, Z.. **Robust shape and topology optimization considering geometric uncertainties with stochastic level set perturbation**. International Journal for Numerical Methods in Engineering, 110 (1):31–56, 2017.

[130] ZHANG, X. S.; DE STURLER, E. ; PAULINO, G. H.. **Stochastic sampling for deterministic structural topology optimization with many load cases: Density-based and ground structure approaches**. Computer Methods in Applied Mechanics and Engineering, 325:463 – 487, 2017.

[131] ZHANG, X. S.; PAULINO, G. H. ; RAMOS JR, A. S.. **Multimaterial topology optimization with multiple volume constraints: Combining the zpr update with a ground-structure algorithm to select a single material per overlapping set**. International Journal for Numerical Methods in Engineering, 114(10):1053–1073, 2018.

A
FLOP Optimization Solution of the BRICK8 Element Local Stiffness Matrix

After running the B&B algorithm described in Section 4.3.1.4 to optimize the number of FLOPs in the BRICK8 element local stiffness matrix (See Eq. (4-36)) product, we obtain the following set of optimal pairs:

$$
\begin{array}{lll}
t_{25} = U_{10} + U_{13} & t_{26} = U_5 + U_{14} & t_{27} = U_6 + U_{12} \\
t_{28} = U_7 + U_{16} & t_{29} = U_2 + U_{17} & t_{30} = U_3 + U_9 \\
t_{31} = U_4 + U_{19} & t_{32} = U_{11} + U_{20} & t_{33} = U_1 + U_{22} \\
t_{34} = U_8 + U_{23} & t_{35} = U_{18} + U_{24} & t_{36} = U_{15} + U_{21} \\
t_{37} = U_9 - U_{24} & t_{38} = U_{17} - U_{23} & t_{39} = U_9 - U_{18} \\
t_{40} = -U_{16} + U_{22} & t_{41} = -U_7 + U_{22} & t_{42} = -U_8 + U_{17} \\
t_{43} = -U_{12} + U_{21} & t_{44} = -U_{14} + U_{20} & t_{45} = U_{12} - U_{15} \\
t_{46} = U_{13} - U_{19} & t_{47} = U_{10} - U_{19} & t_{48} = -U_{11} + U_{14} \\
t_{49} = -U_3 + U_{18} & t_{50} = -U_3 + U_{24} & t_{51} = U_1 - U_{16} \\
t_{52} = U_2 - U_{23} & t_{53} = U_6 - U_{15} & t_{54} = -U_6 + U_{21} \\
t_{55} = -U_4 + U_{13} & t_{56} = U_5 - U_{20} & t_{57} = U_5 - U_{11} \\
t_{58} = -U_4 + U_{10} & t_{59} = -U_2 + U_8 & t_{60} = U_1 - U_7 \\
t_{61} = t_{37} + t_{38} & t_{62} = t_{39} + t_{40} & t_{63} = t_{41} + t_{42} \\
t_{64} = t_{43} + t_{44} & t_{65} = t_{45} + t_{46} & t_{66} = t_{47} + t_{48} \\
t_{67} = -t_{38} + t_{49} & t_{68} = -t_{40} + t_{50} & t_{69} = t_{51} + t_{52} \\
t_{70} = -t_{44} + t_{53} & t_{71} = -t_{46} + t_{54} & t_{72} = t_{55} + t_{56} \\
t_{73} = t_{53} + t_{57} & t_{74} = t_{45} + t_{58} & t_{75} = t_{48} + t_{55} \\
t_{76} = t_{49} + t_{59} & t_{77} = t_{39} + t_{60} & t_{78} = t_{42} + t_{51} \\
t_{79} = t_{43} - t_{57} & t_{80} = t_{54} - t_{58} & t_{81} = t_{47} + t_{56} \\
t_{82} = t_{37} - t_{59} & t_{83} = t_{50} - t_{60} & t_{84} = t_{41} + t_{52}
\end{array}
\tag{A-1}
$$

with these pairs, we can compute the multiplied terms:

$$
\begin{array}{cccc}
m_1 = k_1\,U_1 & m_2 = k_6\,U_1 & m_3 = k_9\,U_1 & m_4 = k_{10}\,U_1 \\
m_5 = k_1\,U_2 & m_6 = k_6\,U_2 & m_7 = k_9\,U_2 & m_8 = k_{10}\,U_2 \\
m_9 = k_1\,U_3 & m_{10} = k_6\,U_3 & m_{11} = k_9\,U_3 & m_{12} = k_{10}\,U_3 \\
m_{13} = k_1\,U_4 & m_{14} = k_6\,U_4 & m_{15} = k_9\,U_4 & m_{16} = k_{10}\,U_4 \\
m_{17} = k_1\,U_5 & m_{18} = k_6\,U_5 & m_{19} = k_9\,U_5 & m_{20} = k_{10}\,U_5 \\
m_{21} = k_1\,U_6 & m_{22} = k_6\,U_6 & m_{23} = k_9\,U_6 & m_{24} = k_{10}\,U_6 \\
m_{25} = k_1\,U_7 & m_{26} = k_6\,U_7 & m_{27} = k_9\,U_7 & m_{28} = k_{10}\,U_7 \\
m_{29} = k_1\,U_8 & m_{30} = k_6\,U_8 & m_{31} = k_9\,U_8 & m_{32} = k_{10}\,U_8 \\
m_{33} = k_1\,U_9 & m_{34} = k_6\,U_9 & m_{35} = k_9\,U_9 & m_{36} = k_{10}\,U_9 \\
m_{37} = k_1\,U_{10} & m_{38} = k_6\,U_{10} & m_{39} = k_9\,U_{10} & m_{40} = k_{10}\,U_{10} \\
m_{41} = k_1\,U_{11} & m_{42} = k_6\,U_{11} & m_{43} = k_9\,U_{11} & m_{44} = k_{10}\,U_{11} \\
m_{45} = k_1\,U_{12} & m_{46} = k_6\,U_{12} & m_{47} = k_9\,U_{12} & m_{48} = k_{10}\,U_{12} \\
m_{49} = k_1\,U_{13} & m_{50} = k_6\,U_{13} & m_{51} = k_9\,U_{13} & m_{52} = k_{10}\,U_{13} \\
m_{53} = k_1\,U_{14} & m_{54} = k_6\,U_{14} & m_{55} = k_9\,U_{14} & m_{56} = k_{10}\,U_{14} \\
m_{57} = k_1\,U_{15} & m_{58} = k_6\,U_{15} & m_{59} = k_9\,U_{15} & m_{60} = k_{10}\,U_{15} \\
m_{61} = k_1\,U_{16} & m_{62} = k_6\,U_{16} & m_{63} = k_9\,U_{16} & m_{64} = k_{10}\,U_{16} \\
m_{65} = k_1\,U_{17} & m_{66} = k_6\,U_{17} & m_{67} = k_9\,U_{17} & m_{68} = k_{10}\,U_{17} \\
m_{69} = k_1\,U_{18} & m_{70} = k_6\,U_{18} & m_{71} = k_9\,U_{18} & m_{72} = k_{10}\,U_{18} \\
m_{73} = k_1\,U_{19} & m_{74} = k_6\,U_{19} & m_{75} = k_9\,U_{19} & m_{76} = k_{10}\,U_{19} \\
m_{77} = k_1\,U_{20} & m_{78} = k_6\,U_{20} & m_{79} = k_9\,U_{20} & m_{80} = k_{10}\,U_{20} \\
m_{81} = k_1\,U_{21} & m_{82} = k_6\,U_{21} & m_{83} = k_9\,U_{21} & m_{84} = k_{10}\,U_{21} \\
m_{85} = k_1\,U_{22} & m_{86} = k_6\,U_{22} & m_{87} = k_9\,U_{22} & m_{88} = k_{10}\,U_{22} \\
m_{89} = k_1\,U_{23} & m_{90} = k_6\,U_{23} & m_{91} = k_9\,U_{23} & m_{92} = k_{10}\,U_{23} \\
m_{93} = k_1\,U_{24} & m_{94} = k_6\,U_{24} & m_{95} = k_9\,U_{24} & m_{96} = k_{10}\,U_{24} \\
m_{97} = k_5\,t_{25} & m_{98} = k_8\,t_{25} & m_{99} = k_5\,t_{26} & m_{100} = k_8\,t_{26} \\
m_{101} = k_5\,t_{27} & m_{102} = k_8\,t_{27} & m_{103} = k_5\,t_{28} & m_{104} = k_8\,t_{28} \\
m_{105} = k_5\,t_{29} & m_{106} = k_8\,t_{29} & m_{107} = k_5\,t_{30} & m_{108} = k_8\,t_{30} \\
m_{109} = k_5\,t_{31} & m_{110} = k_8\,t_{31} & m_{111} = k_5\,t_{32} & m_{112} = k_8\,t_{32} \\
m_{113} = k_5\,t_{33} & m_{114} = k_8\,t_{33} & m_{115} = k_5\,t_{34} & m_{116} = k_8\,t_{34} \\
m_{117} = k_5\,t_{35} & m_{118} = k_8\,t_{35} & m_{119} = k_5\,t_{36} & m_{120} = k_8\,t_{36}
\end{array}
\tag{A-2}
$$

$$
\begin{array}{llll}
m_{121} = k_2\, t_{61} & m_{122} = k_3\, t_{61} & m_{123} = k_4\, t_{61} & m_{124} = k_7\, t_{61} \\
m_{125} = k_2\, t_{62} & m_{126} = k_3\, t_{62} & m_{127} = k_4\, t_{62} & m_{128} = k_7\, t_{62} \\
m_{129} = k_2\, t_{63} & m_{130} = k_3\, t_{63} & m_{131} = k_4\, t_{63} & m_{132} = k_7\, t_{63} \\
m_{133} = k_2\, t_{64} & m_{134} = k_3\, t_{64} & m_{135} = k_4\, t_{64} & m_{136} = k_7\, t_{64} \\
m_{137} = k_2\, t_{65} & m_{138} = k_3\, t_{65} & m_{139} = k_4\, t_{65} & m_{140} = k_7\, t_{65} \\
m_{141} = k_2\, t_{66} & m_{142} = k_3\, t_{66} & m_{143} = k_4\, t_{66} & m_{144} = k_7\, t_{66} \\
m_{145} = k_2\, t_{67} & m_{146} = k_3\, t_{67} & m_{147} = k_4\, t_{67} & m_{148} = k_7\, t_{67} \\
m_{149} = k_2\, t_{68} & m_{150} = k_3\, t_{68} & m_{151} = k_4\, t_{68} & m_{152} = k_7\, t_{68} \\
m_{153} = k_2\, t_{69} & m_{154} = k_3\, t_{69} & m_{155} = k_4\, t_{69} & m_{156} = k_7\, t_{69} \\
m_{157} = k_2\, t_{70} & m_{158} = k_3\, t_{70} & m_{159} = k_4\, t_{70} & m_{160} = k_7\, t_{70} \\
m_{161} = k_2\, t_{71} & m_{162} = k_3\, t_{71} & m_{163} = k_4\, t_{71} & m_{164} = k_7\, t_{71} \\
m_{165} = k_2\, t_{72} & m_{166} = k_3\, t_{72} & m_{167} = k_4\, t_{72} & m_{168} = k_7\, t_{72} \\
m_{169} = k_2\, t_{73} & m_{170} = k_3\, t_{73} & m_{171} = k_4\, t_{73} & m_{172} = k_7\, t_{73} \\
m_{173} = k_2\, t_{74} & m_{174} = k_3\, t_{74} & m_{175} = k_4\, t_{74} & m_{176} = k_7\, t_{74} \\
m_{177} = k_2\, t_{75} & m_{178} = k_3\, t_{75} & m_{179} = k_4\, t_{75} & m_{180} = k_7\, t_{75} \\
m_{181} = k_2\, t_{76} & m_{182} = k_3\, t_{76} & m_{183} = k_4\, t_{76} & m_{184} = k_7\, t_{76} \\
m_{185} = k_2\, t_{77} & m_{186} = k_3\, t_{77} & m_{187} = k_4\, t_{77} & m_{188} = k_7\, t_{77} \\
m_{189} = k_2\, t_{78} & m_{190} = k_3\, t_{78} & m_{191} = k_4\, t_{78} & m_{192} = k_7\, t_{78} \\
m_{193} = k_2\, t_{79} & m_{194} = k_3\, t_{79} & m_{195} = k_4\, t_{79} & m_{196} = k_7\, t_{79} \\
m_{197} = k_2\, t_{80} & m_{198} = k_3\, t_{80} & m_{199} = k_4\, t_{80} & m_{200} = k_7\, t_{80} \\
m_{201} = k_2\, t_{81} & m_{202} = k_3\, t_{81} & m_{203} = k_4\, t_{81} & m_{204} = k_7\, t_{81} \\
m_{205} = k_2\, t_{82} & m_{206} = k_3\, t_{82} & m_{207} = k_4\, t_{82} & m_{208} = k_7\, t_{82} \\
m_{209} = k_2\, t_{83} & m_{210} = k_3\, t_{83} & m_{211} = k_4\, t_{83} & m_{212} = k_7\, t_{83} \\
m_{213} = k_2\, t_{84} & m_{214} = k_3\, t_{84} & m_{215} = k_4\, t_{84} & m_{216} = k_7\, t_{84}
\end{array}
\tag{A-3}
$$

After that, we can optimize the sum of the post-multiplication terms to obtain:

$$
\begin{array}{lll}
n_1 = m_{97} - m_{104} & n_2 = m_{99} - m_{116} & n_3 = m_{101} - m_{118} \\
n_4 = m_{103} - m_{98} & n_5 = m_{105} - m_{112} & n_6 = m_{107} - m_{120} \\
n_7 = m_{109} - m_{114} & n_8 = m_{111} - m_{106} & n_9 = m_{113} - m_{110} \\
n_{10} = m_{115} - m_{100} & n_{11} = m_{117} - m_{102} & n_{12} = m_{119} - m_{108}
\end{array}
\tag{A-4}
$$

Finally, we arrive at the following optimized equations:

$$
\begin{aligned}
f_1 &= -m_{85} + m_{121} + m_{170} - m_{135} - m_{74} - m_{184} - m_{15} + m_4 + n_1 \\
f_2 &= -m_{65} + m_{125} + m_{174} - m_{163} - m_{78} - m_{212} - m_{43} + m_8 + n_2 \\
f_3 &= -m_{33} + m_{129} + m_{178} + m_{203} - m_{82} + m_{156} - m_{59} + m_{12} + n_3 \\
f_4 &= -m_{73} + m_{133} + m_{182} - m_{123} - m_{86} - m_{172} - m_3 + m_{16} + n_4 \\
f_5 &= -m_{53} + m_{137} + m_{186} - m_{151} - m_{90} - m_{200} - m_{31} + m_{20} + n_5 \\
f_6 &= -m_{45} + m_{141} + m_{190} + m_{215} - m_{94} + m_{168} - m_{71} + m_{24} + n_6 \\
f_7 &= -m_{61} + m_{145} + m_{194} - m_{159} - m_{50} - m_{208} - m_{39} + m_{28} + n_7 \\
f_8 &= -m_{89} + m_{149} + m_{198} - m_{139} - m_{54} - m_{188} - m_{19} + m_{32} + n_8 \\
f_9 &= -m_9 + m_{153} + m_{202} + m_{179} - m_{58} + m_{132} - m_{83} + m_{36} + n_3 \\
f_{10} &= -m_{49} + m_{157} + m_{206} - m_{147} - m_{62} - m_{196} - m_{27} + m_{40} + n_9 \\
f_{11} &= -m_{77} + m_{161} + m_{210} - m_{127} - m_{66} - m_{176} - m_7 + m_{44} + n_{10} \\
f_{12} &= -m_{21} + m_{165} + m_{214} + m_{191} - m_{70} + m_{144} - m_{95} + m_{48} + n_6 \\
f_{13} &= -m_{37} - m_{193} - m_{146} + m_{207} - m_{26} + m_{160} - m_{63} + m_{52} + n_9 \\
f_{14} &= -m_{17} - m_{197} - m_{150} + m_{187} - m_{30} + m_{140} - m_{91} + m_{56} + n_5 \\
f_{15} &= -m_{81} - m_{201} - m_{154} - m_{131} - m_{34} - m_{180} - m_{11} + m_{60} + n_{11} \\
f_{16} &= -m_{25} - m_{205} - m_{158} + m_{195} - m_{38} + m_{148} - m_{51} + m_{64} + n_7 \\
f_{17} &= -m_5 - m_{209} - m_{162} + m_{175} - m_{42} + m_{128} - m_{79} + m_{68} + n_2 \\
f_{18} &= -m_{93} - m_{213} - m_{166} - m_{143} - m_{46} - m_{192} - m_{23} + m_{72} + n_{12} \\
f_{19} &= -m_{13} - m_{169} - m_{122} + m_{183} - m_2 + m_{136} - m_{87} + m_{76} + n_4 \\
f_{20} &= -m_{41} - m_{173} - m_{126} + m_{211} - m_6 + m_{164} - m_{67} + m_{80} + n_{10} \\
f_{21} &= -m_{57} - m_{177} - m_{130} - m_{155} - m_{10} - m_{204} - m_{35} + m_{84} + n_{12} \\
f_{22} &= -m_1 - m_{181} - m_{134} + m_{171} - m_{14} + m_{124} - m_{75} + m_{88} + n_1 \\
f_{23} &= -m_{29} - m_{185} - m_{138} + m_{199} - m_{18} + m_{152} - m_{55} + m_{92} + n_8 \\
f_{24} &= -m_{69} - m_{189} - m_{142} - m_{167} - m_{22} - m_{216} - m_{47} + m_{96} + n_{12}
\end{aligned}
\tag{A-5}
$$

The whole computation considering the calculation of all the intermediate terms costs 492 FLOPS (216 multiplications and 276 additions). Meanwhile, the direct computation of the matrix-vector product would cost 1128 FLOPS indicating a reduction of 56% on the total number of operations.

$$
\mathbb{K}_\nu = 0.083
\begin{pmatrix}
-4 & 0 & 0 & 0 & 2 & 2 & 1 & 0 & 1 & 0 & -2 & 0 & 0 & 0 & -2 & 1 & 1 & 0 & 0 & 0 & 1 & -1 & -1 \\
0 & -4 & 0 & -2 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 2 & 0 & 0 & -2 & -1 & 1 & -1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & -4 & -2 & 0 & 0 & -1 & -1 & 1 & 0 & -2 & 0 & 2 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & -2 & -2 & -4 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & -4 & 0 & -2 & 0 & 2 & 0 & 1 & -1 & 1 & 1 & 0 & 0 & 0 & -2 & -1 & 1 & 0 & 1 & 0 \\
2 & 0 & 0 & 0 & 0 & -4 & 0 & -2 & 0 & 1 & -1 & 1 & 0 & 1 & -2 & 2 & 0 & -1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & -1 & 0 & -2 & 0 & -4 & 0 & 0 & 0 & 2 & -2 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 2 & 1 & 0 \\
0 & 1 & -1 & 2 & 0 & -2 & 0 & -4 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 2 & -1 & 1 \\
1 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & -4 & 2 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -2 & -2 & 0 & 0 & -1 & 1 \\
0 & 2 & 0 & 1 & 0 & 1 & 0 & -2 & 2 & -4 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -2 \\
-2 & 0 & -2 & 0 & 1 & -1 & 2 & 0 & 0 & 0 & -4 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 2 \\
0 & 2 & 0 & -1 & 1 & 1 & -2 & 0 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 1 & 2 & 0 \\
0 & 0 & 2 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & -1 & 1 & -4 & 0 & 0 & 0 & 2 & -2 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 2 & -1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & -4 & 0 & 0 & -2 & 0 & 0 & 1 & -1 & 2 & 0 \\
-2 & -2 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & -4 & 2 & 0 & 0 & 1 & 1 & 0 & -1 & 2 \\
1 & -1 & 0 & 0 & 0 & -2 & 1 & 1 & -1 & 1 & 0 & 0 & -2 & 2 & 0 & -4 & 0 & 0 & 2 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 2 & -1 & 1 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & -4 & 0 & -2 & 0 & 0 & 1 & -1 \\
0 & -1 & 1 & 2 & -2 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & -4 & 0 & 2 & 0 & -1 & 1 \\
1 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & -2 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & -2 & 0 & -4 & 0 & 0 & 2 & 2 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & -2 & -1 & 1 & -1 & 0 & 1 & 1 & 2 & 0 & 2 & 0 & -4 & -2 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 0 & -2 & 0 & 0 & 0 & -4 & -2 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 2 & 2 & 0 & 1 & 0 & 1 & 0 & -2 & -2 & -4 & 0 \\
-1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & -2 & -2 & 0 & 2 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & -4 \\
-1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & -2 & 2 & 0 & 0 & -2 & 0 & 1 & -1 & 1 & 2 & 0 & 0 & 0 & -4
\end{pmatrix}
\tag{A-6}
$$

$$
\mathbb{K}_c = 0.021
\begin{pmatrix}
11.0 & 2.0 & 2.0 & -2.6 & -2.0 & -2.0 & -3.3 & -2.0 & -1.0 & 1.3 & 2.0 & 1.0 & 1.3 & 1.0 & 2.0 & -3.3 & -1.0 & -2.0 & -2.6 & -1.0 & -1.0 & -1.3 & 1.0 & 1.0 \\
2.0 & 11.0 & 2.0 & 2.0 & 1.3 & 1.0 & -2.0 & -3.3 & -1.0 & -2.0 & -2.6 & -2.0 & 1.0 & 1.3 & 2.0 & 1.0 & -1.3 & 1.0 & -1.0 & -2.6 & -1.0 & -1.0 & -3.3 & -2.0 \\
2.0 & 2.0 & 11.0 & 2.0 & 1.0 & 1.3 & 1.0 & 1.0 & -1.3 & 1.0 & 2.0 & 1.3 & -2.0 & -2.0 & -2.6 & -2.0 & -1.0 & -3.3 & -1.0 & -1.0 & -2.6 & -1.0 & -2.0 & -3.3 \\
-2.6 & 2.0 & 2.0 & 11.0 & -2.0 & -2.0 & 1.3 & -2.0 & -1.0 & -3.3 & 2.0 & 1.0 & -3.3 & 1.0 & 2.0 & 1.3 & -1.0 & -2.0 & -1.3 & -1.0 & -1.0 & -2.6 & 1.0 & 1.0 \\
-2.0 & 1.3 & 1.0 & -2.0 & 11.0 & 2.0 & 2.0 & -2.6 & -2.0 & 2.0 & -3.3 & -1.0 & 1.3 & 1.0 & 1.0 & -1.0 & 1.3 & 2.0 & 1.0 & -3.3 & -2.0 & 1.0 & -2.6 & -1.0 \\
-2.0 & 1.0 & 1.3 & -2.0 & 2.0 & 11.0 & -1.0 & 2.0 & 1.3 & -1.0 & 1.0 & -1.3 & 2.0 & -1.0 & -3.3 & 2.0 & -2.0 & -2.6 & 1.0 & -2.0 & -3.3 & 1.0 & -1.0 & -2.6 \\
-3.3 & -2.0 & 1.0 & 1.3 & 2.0 & -1.0 & 11.0 & 2.0 & -2.0 & -2.6 & -2.0 & 2.0 & -2.6 & -1.0 & 1.0 & -1.3 & 1.0 & -1.0 & 1.3 & 1.0 & -2.0 & -3.3 & -1.0 & 2.0 \\
-2.0 & -3.3 & 1.0 & -2.0 & -2.6 & 2.0 & 2.0 & 11.0 & -2.0 & 2.0 & 1.3 & -1.0 & -2.6 & 1.0 & -1.0 & -3.3 & 2.0 & 1.0 & 1.3 & -2.0 & 1.0 & -1.3 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.3 & -1.0 & -2.0 & 1.3 & -2.0 & -2.0 & 11.0 & -2.0 & -1.0 & 1.3 & 1.0 & 1.0 & -2.6 & 1.0 & 2.0 & -3.3 & 2.0 & -2.6 & 2.0 & 1.0 & -3.3 & -3.3 \\
1.3 & -2.0 & 1.0 & -3.3 & 2.0 & -1.0 & -2.6 & 2.0 & -2.0 & 11.0 & -2.0 & 2.0 & -1.3 & -1.0 & 1.0 & -2.6 & 1.0 & -1.0 & -3.3 & -2.0 & 1.3 & -1.0 & 2.0 & 2.0 \\
2.0 & -2.6 & 2.0 & 2.0 & -3.3 & 1.0 & -2.0 & 1.3 & -1.0 & 11.0 & 2.0 & 1.0 & -3.3 & 2.0 & 1.0 & -2.6 & 1.0 & -1.0 & -1.3 & -1.0 & -1.0 & 1.3 & -2.0 & 2.0 \\
1.0 & -2.0 & 1.3 & 1.0 & -1.0 & -1.3 & 2.0 & -1.0 & 1.3 & 2.0 & 11.0 & -1.0 & 2.0 & -3.3 & -1.0 & 1.0 & -2.6 & 2.0 & -2.0 & -1.0 & -3.3 & -2.0 & 1.3 & -2.0 \\
1.3 & 1.0 & -2.0 & -3.3 & -1.0 & -1.3 & 2.0 & -2.6 & 1.0 & -1.0 & 1.3 & -2.0 & 1.0 & -2.6 & -2.6 & -2.0 & 2.0 & -3.3 & -2.0 & -2.6 & 2.0 & -2.0 & 2.0 & -2.6 \\
1.0 & 1.3 & -2.0 & 1.0 & -1.3 & -1.0 & -1.0 & -2.6 & 1.0 & -1.0 & -2.6 & 2.0 & 1.3 & -1.0 & -2.0 & 2.0 & 1.3 & -1.0 & -2.0 & -3.3 & 1.0 & -2.0 & -2.6 & -1.0 \\
2.0 & 2.0 & -2.6 & 2.0 & 1.0 & -2.6 & 1.0 & -3.3 & -1.0 & -1.3 & 2.0 & 1.3 & -2.0 & -2.0 & 11.0 & -2.0 & -2.6 & 1.0 & -1.0 & -1.3 & -1.0 & -2.6 & 2.0 & 1.3 \\
-3.3 & 1.0 & -2.0 & 1.3 & 2.0 & -1.0 & -2.0 & -2.6 & -2.0 & 1.3 & -1.0 & -2.6 & 2.0 & -3.3 & -2.0 & 11.0 & -2.0 & -2.0 & -2.6 & -2.0 & -1.0 & -3.3 & -2.0 & -2.0 \\
-1.0 & -1.3 & -1.0 & -2.0 & -1.0 & 1.3 & -2.0 & -1.0 & -2.0 & 1.3 & -1.0 & -2.0 & -1.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.6 & -2.0 & -1.0 & -2.0 & 11.0 & 2.0 \\
-2.0 & 1.0 & -3.3 & -2.0 & -2.6 & -1.0 & 1.3 & -1.0 & -2.0 & 2.0 & -3.3 & -1.0 & 1.3 & 2.0 & -2.0 & -2.0 & 11.0 & 1.0 & -2.0 & 1.3 & 1.0 & -1.0 & -1.3 & -1.3 \\
-2.6 & -1.0 & -1.0 & -1.3 & 1.0 & -1.3 & 1.0 & -3.3 & -1.0 & -3.3 & -2.0 & -1.0 & -2.0 & -1.0 & -1.0 & -2.6 & 1.0 & 2.0 & 2.0 & -2.6 & 2.0 & -2.6 & -2.0 & -2.0 \\
-1.0 & -2.6 & -1.0 & -1.0 & -3.3 & -2.0 & 1.3 & -2.0 & -2.6 & -2.0 & -1.0 & -3.3 & 1.0 & -2.0 & -3.3 & -1.0 & 2.0 & 1.3 & 1.0 & -2.0 & 2.0 & -2.0 & 1.3 & 1.0 \\
-1.0 & -1.0 & -2.6 & -1.0 & -2.0 & -3.3 & -2.0 & 1.3 & -2.0 & -1.0 & -3.3 & 1.0 & -1.3 & -1.0 & -1.3 & 2.0 & 2.0 & 1.3 & 2.0 & 2.0 & 11.0 & 2.0 & 1.0 & 1.3 \\
-1.3 & -1.0 & -1.0 & -2.6 & 1.0 & 1.0 & -3.3 & 1.0 & 2.0 & 1.3 & -1.0 & -2.0 & 1.3 & -2.0 & -1.0 & -3.3 & 2.0 & 1.0 & -2.6 & 2.0 & 2.0 & 11.0 & -2.0 & -2.0 \\
1.0 & -3.3 & -2.0 & 1.0 & -2.6 & -1.0 & -1.0 & -1.3 & 1.0 & -1.0 & 1.3 & 2.0 & 2.0 & -2.6 & -2.0 & 2.0 & -3.3 & -1.0 & -2.0 & 1.3 & 1.0 & -2.0 & 11.0 & 2.0 \\
1.0 & -2.0 & -3.3 & 1.0 & -1.0 & -2.6 & 2.0 & -1.0 & -3.3 & 2.0 & -2.0 & -2.6 & -1.0 & 2.0 & 1.3 & -1.0 & 1.0 & -1.3 & -2.0 & 1.0 & 1.3 & -2.0 & 2.0 & 11.0
\end{pmatrix}
\tag{A-7}
$$

B
Compliance Minimization with Continuously Varying Loads

This appendix presents a brief description on how to extend the formulation for compliance minimization. The compliance minimization optimization statement is presented in Eq. (B-1):

$$\begin{aligned} \min_{\mathbf{z}} \quad & C(\mathbf{z},\boldsymbol{\theta}) = \mathbf{U}^T(\mathbf{z},\boldsymbol{\theta})\mathbf{K}(\mathbf{z})\mathbf{U}(\mathbf{z},\boldsymbol{\theta}) \\ \text{s.t.} \quad & g_V(\mathbf{z}) \leq 0 \\ & 0 \leq z_e \leq 1, \quad e = 1,\ldots,N_e \\ \text{with:} \quad & \mathbf{K}(\mathbf{z})\mathbf{U}(\mathbf{z},\boldsymbol{\theta}) = \mathbf{F}(\boldsymbol{\theta}) \end{aligned} \tag{B-1}$$

in which, $C(\mathbf{z},\boldsymbol{\theta})$ is the compliance, and $g_V(\mathbf{z})$ is the volume constraint. Similar to the stress constraint formulation, the load direction in the compliance formulation also depends on a variable θ. However, instead of stress constraints, which limit the worst-case stress for any load direction possible, now we minimize the worst-case compliance. Here, we will derive the formulation for the equivalent of the previously describe case 1 of load variation (Section 5.1.1), in which we have a single load that can vary 360^o in direction. The other load cases can be derived following the same approach. The derivations for the worst-case compliance starts in the same way as the stress, by decomposing the load into linearly independent components:

$$\mathbf{F}(\theta) = \mathbf{F}_x\cos(\theta) + \mathbf{F}_y\sin(\theta) \tag{B-2}$$

we then replace the this expression for the loads in the equilibrium equation:

$$\mathbf{U}(\mathbf{z},\boldsymbol{\theta}) = \mathbf{K}^{-1}\mathbf{F}(\theta) = \left(\mathbf{K}^{-1}\mathbf{F}_x\right)\cos(\theta) + \left(\mathbf{K}^{-1}\mathbf{F}_y\right)\sin(\theta) \tag{B-3}$$

By defining $\mathbf{U}_x = \left(\mathbf{K}^{-1}\mathbf{F}_x\right)$ and $\mathbf{U}_y = \left(\mathbf{K}^{-1}\mathbf{F}_y\right)$, we can compute the compliance as:

$$C(\mathbf{z},\boldsymbol{\theta}) = \left[(\mathbf{U}_x)\cos(\theta) + \left(\mathbf{U}_y\right)\sin(\theta)\right]^T \mathbf{K}(\mathbf{z}) \left[(\mathbf{U}_x)\cos(\theta) + \left(\mathbf{U}_y\right)\sin(\theta)\right] \tag{B-4}$$

To simplify the expression in Eq. B-4 we define the quadratic compliance terms $t_{xx} = \mathbf{U}_x^T\mathbf{K}(\mathbf{z})\mathbf{U}_x$, $t_{yy} = \mathbf{U}_y^T\mathbf{K}(\mathbf{z})\mathbf{U}_y$, and $t_{xy} = \mathbf{U}_x^T\mathbf{K}(\mathbf{z})\mathbf{U}_y$, which we then substitute in Eq. B-4:

$$C(\mathbf{z},\boldsymbol{\theta}) = t_{xx}\ \cos^2(\theta) + t_{yy}\ \sin^2(\theta) + 2t_{xy}\ \cos(\theta)\sin(\theta) \tag{B-5}$$

We simplify this equation even further using trigonometric identities:

$$C(\mathbf{z}, \boldsymbol{\theta}) = t_{xy} \sin(2\,\theta) + 0.5 \left[\Big(t_{xx} - t_{yy} \Big) \cos(2\,\theta) + t_{xx} + t_{yy} \right], \tag{B-6}$$

and we finally obtain the optimization problem for the worst-case compliance:

$$\begin{aligned} \max_{\theta \in \Gamma} \;\; & C(\mathbf{z}, \boldsymbol{\theta}) = t_{xy} \sin(2\,\theta) + 0.5 \left[\Big(t_{xx} - t_{yy} \Big) \cos(2\,\theta) + t_{xx} + t_{yy} \right] \\ \text{with:} \;\; & \mathbf{K}(\mathbf{z})\mathbf{U}_x = \mathbf{F}_x \\ & \mathbf{K}(\mathbf{z})\mathbf{U}_y = \mathbf{F}_y \end{aligned} \tag{B-7}$$

which is exactly the same as the one in Eq. 5-8, for which the solution is:

$$\theta^{\star} = \theta^{cr}_{max} = \frac{1}{2} \tan^{-1} \Big(\, 2\, t_{xy}, \; t_{xx} - t_{yy} \, \Big) \tag{B-8}$$