TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM School of Engineering and Design

Enhanced computational design methods for large industrial node-based shape optimization problems

Ihar Antonau

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. habil. Fabian Duddeck

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Kai-Uwe Bletzinger
2. Prof. Dr.-Ing. Fernass Daoud
3. Prof. Dr.-Ing. Axel Schumacher

Die Dissertation wurde am 17.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 25.05.2023 angenommen.

Schriftenreihe des Lehrstuhls für Statik
TU München

Band 59

Ihar Antonau

Enhanced computational design methods for large industrial node-based shape optimization problems

Quasi-Newton relaxed gradient projection algorithm and Vertex Morphing with adaptive filtering radius

München 2023

Veröffentlicht durch

Kai-Uwe Bletzinger
Lehrstuhl für Statik
Technische Universität München
Arcisstr. 21
80333 München

Telefon: +49(0)89 289 22422
Telefax: +49(0)89 289 22421
E-Mail: kub@tum.de
Internet: www.bgu.tum.de/st/startseite/

ISBN: 978-3-943683-71-4

Kurzfassung

In dieser Arbeit werden zwei Aspekte der knotenbasierten Formoptimierung behandelt. Das erste Thema betrifft die Entwicklung von mathematischen Algorithmen für große Formoptimierungsprobleme mit technischen Einschränkungen. Der zweite Aspekt betrifft die Modifizierung der Vertex-Morphing-Methode, um ihre Nutzbarkeit, Robustheit und Flexibilität in praktischen Anwendungen zu verbessern.

Die relaxierte Gradientenprojektionsmethode wird als guter mathematischer Algorithmus für die knotenbasierte Formoptimierung vorgeschlagen. Der Algorithmus basiert auf Rosens bekannter Gradientenprojektionsmethode und bietet eine Lösung für das Zick-Zack-Problem. Der vorgestellte Algorithmus ist problemunabhängig.

Darüber hinaus wird in dieser Arbeit die Möglichkeit untersucht, die Barzilai-Borwein-Liniensuchtechnik anzuwenden, um die Stabilität und Effizienz des Optimierungsprozesses zu verbessern. Es werden mehrere Modifikationen der ursprünglichen Methode vorgeschlagen, um die Methode für große Formoptimierungsprobleme anzupassen. Zusätzlich wird die relaxierte Gradientenprojektionsmethode mit der modifizierten Barzilai-Borwein-Methode kombiniert, um große eingeschränkte Formoptimierungsprobleme zu lösen.

In dieser Arbeit wird der Einfluss der Größe des Filterradius in einem auf der Vertex-Morphing-Methode basierenden Formoptimierungsprozess auf das Endergebnis untersucht. In dieser Arbeit wird vorgeschlagen, einen adaptiven Filterradius zu verwenden, der bei jeder Optimierungsiteration berechnet und aktualisiert wird. Darüber hinaus können unterschiedliche Filterradien für verschiedene Teile der Designoberfläche verwendet werden, verglichen mit einem konservativen Radius in der ursprünglichen Methode.

Ein weiterer Schwerpunkt dieser Arbeit ist die Verwendung von Formoptimierungsmethoden für additive Fertigungsanwendungen. In dieser Arbeit werden zwei spezifische Fertigungseinschränkungen formuliert und untersucht: Stapelbarkeit und Selbsttragfähigkeit.

Verschiedene Beispiele zur Formoptimierung demonstrieren die Leistungsfähigkeit der entwickelten Optimierungsverfahren.

Abstract

The thesis discusses two aspects of node-based shape optimization. The first topic concerns the development of mathematical algorithms for large-scale shape optimization with engineering constraints. The second one modifies the Vertex morphing method to improve its usability, robustness, and flexibility in practical applications.

The relaxed gradient projection method is proposed as a good mathematical algorithm for node-based shape optimization. The algorithm is based on the well-known Rosen's gradient projection method and suggests a solution to the zig-zagging problem. The presented algorithm is problem-independent.

Furthermore, this thesis studies the possibility of applying the Barzilai-Borwein line search technique to enhance the stability and efficiency of the optimization process. Several modifications to the original method are proposed to adapt the method for large shape optimization problems. Additionally, the relaxed gradient projection method is combined with the modified Barzilai-Borwein method to solve large constrained shape optimization problems.

In this work, the influence of the filtering radius size in shape optimization workflow based on the Vertex Morphing method is studied. This thesis suggests using an adaptive filtering radius size that is computed and updated every optimization iteration. Additionally, different filtering radius sizes can be applied for different parts of the design surface compared to one conservative radius in the original method.

Another main focus of this work is the usage of shape optimization methods for additive manufacturing applications. Two specific manufacturing constraints are formulated and studied in this work: stackability and self-support.

Various shape optimization examples demonstrate the performance of the developed optimization techniques.

Acknowledgements

This thesis was written from 2018 to 2023 during my time as a researcher at the Chair of Structural Analysis at the Technical University of Munich.

First of all, I would like to express my gratitude towards Prof. Dr.-Ing. Kai-Uwe Bletzinger for providing the possibility to work in the field of shape optimization. His interest and passion to structural optimization leads to a great working environment.

I also want to thank Prof. Dr.-Ing. Fernass Daoud and Prof. Dr.-Ing. Axel Schumacher for completing my board of examiners, as well as Prof. Dr.-Ing. Fabian Duddeck for the organization. Their interest in my work is gratefully acknowledged.

Thanks are also due to Dr.-Ing. Majid Hojjat for organizing our incredible cooperation between our chair and BMW Group and his mentoring of my work. I also wish to thank Dr.-Ing. Steffen Jahnke for his support.

Another thanks are directed at all my colleagues at the Chair of Structural Analysis, Shape Module and BMW Group for the cooperation and pleasant times. Thank you Armin Geiser, Andrew Brodie, David Schmölz, Reza Najian Asl, Bastian Devresse for the close cooperation over the years. Thanks are also due to Mr. Vignesh Manickavasagam Manian for his technical support in my numerical experiments and long discussions on and off-topic. I also wish to thank Prof. Dr.-Ing. Roland Wüchner for discussions and his support.

Finally, I want to thank my family and friends for their unconditional support during my studies and life in Germany.

Ihar Antonau
Technische Universität München
January, 2023

Contents

Chapter 1

Introduction

1.1 Motivation

Design optimization has become a standard technology in the design optimization cycle. It is applied in various fields of engineering, for example: aerospace (Baumgärtner et al. [7], James et al. [35], and Schramm et al. [52]), automotive (Bartz et al. [4] and Othmer [47]), additive manufacturing (Ghantasala et al. [28], Langelaar [36], and Lianos et al. [38]), shipbuilding (Müller et al. [43] and Stück et al. [54]). The main advantage of design optimization is that it can replace an iterative design process to accelerate the design cycle and obtain better results. Engineers can use design optimization in an automated workflow to solve templated problems, but it doesn't provide a "push-button" solution for complex optimization problems. A designer's decisions are still needed to determine the specifications and initial design and formulate the optimization problem. It requires expertise in both the subject area and numerical optimization. One should decide the objective, constraints, design parameters, etc. Different optimization methods may find other optimal solutions or fail to converge. As a result, the optimization problem formulation, correct parameterization, and optimization methods directly influence the design's success, so the designer must be able to use the design optimization tools well.

Following the challenges mentioned above, the design tools, such as optimization algorithms or parameterization strategies, must be simple to setup to solve unknown problems. At the same time, the methods should be robust and efficient to solve the optimization problem with reasonable computational time and be able to work with strongly non-linear functions.

In previous work, the gradient projection method has been successfully applied to solve constraint optimization problems with Vertex Morphing, Baumgärtner [6], Ertl [17], Najian Asl [44], and Najian Asl et al. [45] with a scaled constant step size. However, the method performance can be strongly affected by the well-known zig-zagging behavior, Fletcher [20], while the constraint enters and leaves the active set of the constraints in a course.

In this work, we propose the relaxed gradient projection method to avoid zig-zagging by keeping constraints active in the critical zone close to the constraint limit.

In the book by Martins et al. [41], the authors suggest using backtracking line search methods based on the strong Wolfe's condition. However, the drawback of this solution is the implementation complexity because the communication and data exchange between the physical solvers and the optimizer happens a few times every iteration. Additionally, the computational efficiency of the method is not guaranteed. In this work, we try to incorporate the Barzilai-Borwein method, Barzilai et al. [5], with limited step length to solve constrained shape optimization problems.

In this work, Vertex Morphing is used as a parameterization method to parameterize the design space, which was introduced by Bletzinger [8] and Hojjat et al. [33]. The filtering radius size plays a key role in Vertex Morphing for the outcome results. The filtering radius size has two roles:

1. It controls the surface and meshes quality by filtering the sensitivities and the shape updates.

2. It guides the optimizer to a certain local minimum with required shape modes and smoothness.

In this work, I study the possibilities of separating these two roles of the filtering radius. On the one side, to compute the filtering radius size that is necessary for the filtering operation and it adjusts the radius sizes for each node at every iteration. On the other side, the designer should be able to locally set up the filtering radius size for each node to find desired local optimums and better explore the design space.

The latest development of additive manufacturing (AM) and its usage in small series production opens a prospective integration between numerical design optimization tools and the manufacturing process. In contrast to classical manufacturing methods, AM has drastically fewer restrictions on the printed shapes and topologies, allowing the manufacture of complex high-performing parts, designed by numerical optimization techniques.

My motivation for this work is to improve the existing methods to make them easier for engineers in daily practice, avoiding additional parameters which may strongly reduce the robustness and efficiency of the optimization process.

1.2 Goals

Based on the aforementioned motivation, the research goals can be formulated as follows:

1. Extend the Vertex Morphing method for daily practical usage in industrial applications. On the one side, the goal is to simplify the parametrization setup by computing the required minimal radius size to ensure that a generated design has a smooth design surface and a

good quality of the numerical mesh. On the other side, the localized filter radius sizes can be chosen to explore the design space better.

2. Find or modify the optimization algorithms that can handle various industrial, geometrical and physical response functions with minimal user input. Ideally, the method should have suitable default parameters with acceptable performance and accuracy in most cases.

3. Formulate the AM-specific constraint for shape optimization to improve the manufacturability of the found solutions, and allow better integration into the overall design process.

1.3 Overview

This paper-based dissertation is based on two published peer-reviewed research papers, which are presented in Appendix A & B, and they are referred in the text as Publication I and Publication II accordingly.

1.3.1 Publication I

Antonau, I., Hojjat, M. & Bletzinger, KU. Relaxed gradient projection algorithm for constrained node-based shape optimization. Struct Multidisc Optim 63, 1633–1651 (2021). https://doi.org/10.1007/s00158-020-02821-y

Contributions: Ihar Antonau developed and implemented the relaxed gradient projection method to avoid zig-zagging behavior and solve shape optimization methods in the efficient way. Majid Hojjat and Kai-Uwe Bletzinger supervised this study and reviewed the manuscript. All authors approved the final version.

1.3.2 Publication II

Antonau, I., Warnakulasuriya, S., Bletzinger, KU., Bluhm, F. M., Hojjat, M. & Wüchner, R. Latest developments in node-based shape optimization using Vertex Morphing parameterization. Struct Multidisc Optim 65, 198 (2022). https://doi.org/10.1007/s00158-022-03279-w

Contributions: Ihar Antonau developed the Vertex Morphing technique with adaptive filtering radius sizes and applied the method to the industrial example. Suneth Warnakulasuriya implemented the proposed method in the open-source framework Kratos Multiphysics and he solved the academic example. Fabio Michael Bluhm has done the Master thesis ("Adaptive filtering for the Vertex Morphing technique in the context of a node-based shape optimization") under the supervision of Ihar Antonau. In the thesis, the proposed ideas has been firstly applied to academic examples. Majid Hojjat, Roland Wüchner and Kai-Uwe Bletzinger supervised this study and reviewed the manuscript. All authors approved the final version.

1.3.3 Text part

The text part of the dissertation is structured as follows:

- Chapter 2 introduces basic definitions of shape optimization. The Chapter discusses the mathematical foundation of the node-based parameterizations, where the filtering operator is based on the Euclidian distances, e.g., Vertex Morphing.
- Chapter 3 introduces Vertex Morphing with adaptive filtering sizes. The Chapter is based on Publication II. The Chapter discusses about the influence of the filter radius sizes on the optimization outcome. The academic structural optimization problem supports the Chapter.
- Chapter 4 formulates the shape optimization problem with Vertex Morphing. The Chapter discusses the usability of the optimization algorithms and their advantages and disadvantages in the Vertex Morphing context.
- Chapter 5 is an unpublished work about the Barzilai-Borwein method. The method is introduced in Publication II. The Chapter 5 shows the studies of the method for structural and CFD-based shape optimization problems with Vertex Morphing. The discussions are supported with academic and industrial importance examples.
- Chapter 6 is an unpublished work about shape optimization for additive manufacturing (AM) application. The two crucial AM-specific constraints are reviewed and formulated for node-based shape optimization. First, the review of the stackabilization process using shape optimization is shown, based on the previous work of my colleagues, Aditya Lakshmi Ghantasala and Reza Najian Asl. In this work, stackability is formulated as an objective function with a fixed stackable direction. The second AM-specific constraint is the self-support constraint, which is well-known in topology optimization. This work proposes a novel formulation for node-based shape optimization based on two feasibility criteria: critical angle and distance to supporting nodes. Both formulations are supported with numerical examples of industrial importance.

Finally, the conclusions, discussions, possible future research and the open questions are summarized for each goal in Conclusions.

Chapter 2

Shape optimization fundamentals

This section introduces the basic theory of shape optimization and follows the definitions by Prof. Kai-Uwe Bletzinger, Bletzinger [9].

2.1 Shape evolution

The shape optimization follows the idea of a design evolution process that depends on the *pseudo time* t. Pseudo time represents not more than the sequence of shape changes, from initial to optimal shape, Fig. 2.1. Similarly, the pseudo time is used in Computation Fluid Dynamics for steady-state cases, where the flow converges from initial conditions to its steady solution through pseudo time. Consequently, the design surface Γ and the volume $\Omega(\Gamma)$ of a body are functions of t, where the initial state is related to time t_0. The material points are defined on the surface Γ using surface material coordinate, $[\xi]$. The material coordinates are invariant of pseudo time t; hence they always represent the same point or part of the design surface. In Euclidean space, the position of a material point can be described by its position vector $\boldsymbol{x}(\xi, t)$. *Design trajectory* is a trajectory, that a material point travels in space through pseudo time t, and it connects the initial position $\boldsymbol{x}_0(\xi, 0)$ and the current one $\boldsymbol{x}(\xi, t)$. Two independent components define the design trajectory: a *design control field* (or also called *design variables*) and an *optimization algorithm*. The *design control field* determines "what" can be changed and the *optimization algorithm* decides "how" to change.

2.2 Design control field

In optimization problems, *design variables* are the unknowns that have to be found to minimize the objective function. In the context of shape optimization, the unknown is the shape of the design surface, which is controlled via *design control field*. The field can be understood as parameters that define the shape

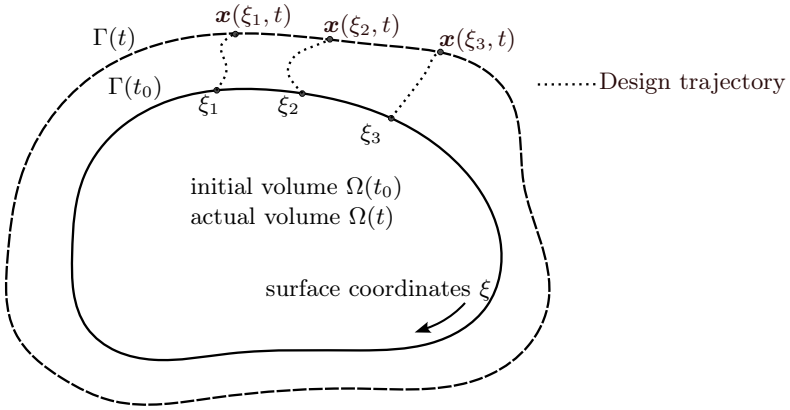
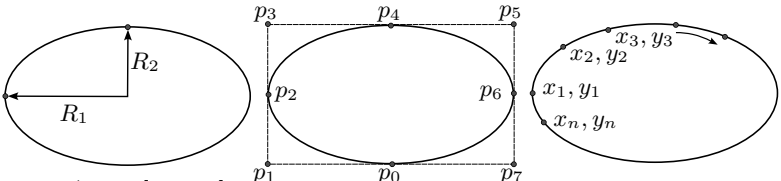


Figure 2.1: Design evolution of the body

of a design trajectory, for instance, its curvature. Moreover, the positions of the material points appear to be functions of the design control field:

$$\boldsymbol{x}(\xi, t) = \boldsymbol{x}(\boldsymbol{p}(t)) \tag{2.1}$$

The classification of the shape optimization problem depends on the design control field choice. In Fig. 2.2, three classes of the parametrization are shown: a) *technical design optimization*, b) CAD, and c) CAD-free or *parameter-free* shape optimization. CAD and CAD-free can be classified as *form finding*.



a) $\boldsymbol{p} = [R_1, R_2]$ b) $\boldsymbol{p} = [p_0, p_2, ..., p_7]$ c) $\boldsymbol{p} = [x_1, y_1, ..., x_n, y_n]$

Figure 2.2: Controlling ellipse shape with different design controls

Technical design optimization is characterized by a large number of constraints and small design freedom to modify the shape. It is often used in the latest stages of the design cycle to "tune" the system's performance. Typical choices of design variables are diameters, lengths, angles, etc.

Form finding, in contrast, is a method to explore and find new before unknown shapes. Consequently, these problems are characterized by a very large, ideally an "infinite" number of degrees of freedom and few, often no constraints. In CAD-based optimization problems, the CAD parameters are used directly as the design control field, for instance, NURBS, Fig. 2.2b. In CAD-free, the discretization of the design surface, e.g., nodes of the surface

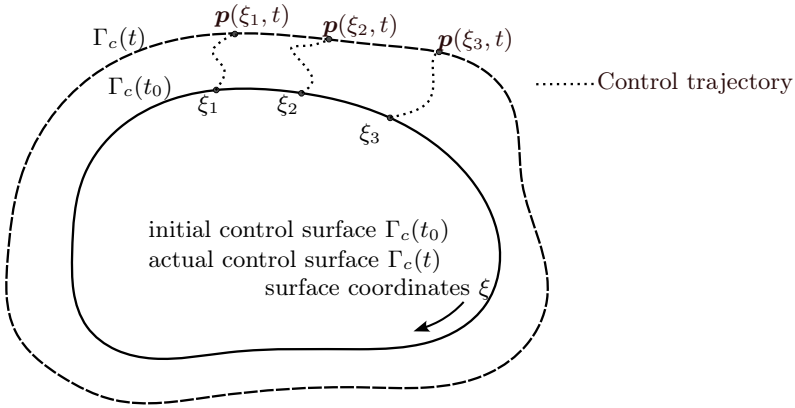


Figure 2.3: Control field evolution of the design body

mesh, is used as the design control field. A particular case where the spatial coordinates of the nodes are used as a design control field is called *node-based* shape optimization problem.

In CAD-free based optimization problems, the position of the material point appears to be functions of the design control field. In 3D space, it is as follows:

$$\boldsymbol{x}(\xi, t) = \boldsymbol{x}(\boldsymbol{p}(\xi, t) = \begin{cases} x_x(p_x(\xi, t)) \\ x_y(p_y(\xi, t)) \\ x_z(p_z(\xi, t)) \end{cases} \tag{2.2}$$

An example of a good choice for $\boldsymbol{p}$ as p_x, p_y, p_z for each spatial direction of the position vector x. Alternatively, one can choose $\boldsymbol{p}$ in the normal direction aligned to surface normal vector $\boldsymbol{n(\xi, t)}$. In general, the design control field is an abstract field that controls the shape evolution and can be aligned to any distinct "move" directions due to specific design problems. Importantly, the design control field is always connected to material points on the actual design surface, $\boldsymbol{p}(\xi, t)$. Consequently, each control parameter update Δp can be divided into two components: in the direction of the surface normal, p_n, and tangential to the normal, p_t.

Fig. 2.3 represents the design control field as a control surface Γ_c, which represents an abstract surface with a different set of material points. *Control trajectory* is an analog of *design trajectory*, that shows the changes of the design control parameter in control space. In practice, the control surface is not required, and it isn't calculated.

Remark. The *control trajectory* is an abstract path, which doesn't show the path of the material point. It is also important to understand, that the design control parameter $\boldsymbol{p}$ is not the spatial coordinates of the material points, where they are defined, but an abstract field of parameters. An understandable example of such a control field can be curvatures at material

point ξ. Additionally, the design control field can be extended via technical parameters, for instance, scaling or rotation angles.

2.3 Design and control velocity

Design velocity is the change in time of the spatial position $\boldsymbol{x}$ of a material point. The term velocity is adopted from theoretical mechanics, Savchuk et al. [51], and it is described as the first derivative of the position vector with respect to pseudo time:

$$\boldsymbol{v}(\xi, t) = \frac{d\boldsymbol{x}}{dt} = \frac{d\boldsymbol{x}}{d\boldsymbol{p}} \frac{d\boldsymbol{p}}{dt} \tag{2.3}$$

The design velocity can be understood as the rate of boundary change in time. The velocity vector is tangential to the design trajectory, Fig. 2.4. In general, the design velocity is not aligned with the actual surface normal. At the same time, only the normal component of the design velocity vector changes the actual design boundary, while the tangential component moves the material point along the body's surface. In practice, the tangential component plays a key part in conserving the quality of the discretized design surface as the basis of the numerical analysis and optimization.

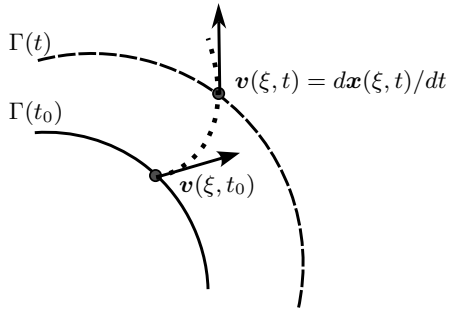


Figure 2.4: Design velocity $\boldsymbol{v}$

In a similar way, the *control velocity* is the change rate of the control design field:

$$\boldsymbol{v}_c(\xi, t) = \frac{d\boldsymbol{p}}{dt} \tag{2.4}$$

The *control velocity* is computed directly by the optimization algorithm.

2.4 Design objective

Design objective function F is a function that defines the performance of the system and allows us to quantify which design is better. The volumetric objective, for instance, the mass of the body, as the volume integral, is:

$$F(t) = F(\boldsymbol{p}(t)) = \int_{\Omega} f_{\Omega}(\boldsymbol{p}(t)) d\Omega \tag{2.5}$$

The surface design objective function, e.g., drag force of the body as the surface integral of the aerodynamic forces at each material point:

$$F(t) = F(\boldsymbol{p}(t)) = \int_{\Gamma} f_{\Gamma}(\boldsymbol{p}(t)) d\Gamma \tag{2.6}$$

where f_{Ω} and f_{Γ} are the design volume and surface density of the objective function. It is important to note that the design objective function changes in pseudo time due to the evolution of the design surface Γ. In general, the objective function is invariant to time, e.g., the mass of the body doesn't change without design surface evolution.

2.5 Shape derivative

The time derivative of an objective function F in the context of shape optimization is called *shape derivative*, and it is a sum of contributions induced by surface and volume change. The change of volume is related to the normal surface component of the design velocity $\boldsymbol{v}$ at the surface, and the Gauss-Ostrogradsky theorem can be applied:

$$\frac{dF(t)}{dt} = \frac{d}{dt}\left[\int_{\Omega(t)} f_{\Omega} d\Omega\right] = \int_{\Omega(t)} \left(\frac{\partial f_{\Omega}}{\partial t} + \frac{df_{\Omega}}{d\boldsymbol{x}}\boldsymbol{v} + f_{\Omega}\ \text{div}\ \boldsymbol{v}\right) d\Omega = \int_{\Omega(t)} \frac{\partial f_{\Omega}}{\partial t} d\Omega + \int_{\Omega(t)} \frac{df_{\Omega}}{d\boldsymbol{x}}\boldsymbol{v} d\Omega + \int_{\Omega(t)} f_{\Omega}\boldsymbol{v}\cdot\boldsymbol{n} d\Omega \tag{2.7}$$

2.5.1 Surface driven volumetric problems

Considering a volumetric objective function, the design density f_{Ω} is invariant to time and to changes of the design surface. For example, in the mass minimization problem, where the design density is the material density, it is invariant to time, $\frac{\partial f_{\Omega}}{\partial t} = 0$, and it doesn't change with respect to changes of the design surface, $\frac{df_{\Omega}}{d\boldsymbol{x}}\boldsymbol{v} = 0$. The shape derivative holds:

$$\frac{dF(t)}{dt} = \int_{\Omega(t)} f_{\Omega}\ \text{div}\ \boldsymbol{v} d\Omega = \int_{\Gamma(t)} f_{\Omega}(\boldsymbol{v}\cdot\boldsymbol{n}) d\Gamma = \int_{\Gamma(t)} f_{\Omega}\frac{dx_n}{dt} d\Gamma = \int_{\Gamma(t)} \frac{df_{\Gamma}}{dx_n}\frac{dx_n}{dt} d\Gamma \tag{2.8}$$

Eq. 2.8 shows that changes in the volumetric objective function, e.g., mass, are only induced by changes of the design surface in normal direction x_n.

2.5.2 Surface optimization problems

The surface objective function is modified through the changes in the design surface Γ and the tangential growth across the surface edge Ψ (stretching). Considering eq. 2.6, eq. 2.7 can be modified as:

$$\frac{dF(t)}{dt} = \int_{\Gamma(t)} \left(\frac{\partial f_\Gamma}{\partial t} + \frac{df_\Gamma}{d\boldsymbol{x}} \boldsymbol{v} + f_\Gamma \ \mathrm{div}_\Gamma \boldsymbol{v} \right) d\Gamma = \int_{\Gamma(t)} \left[\frac{\partial f_\Gamma}{\partial t} + \left(\frac{\partial f_\Gamma}{\partial \boldsymbol{x_n}} + f_\Gamma \ \mathrm{div}_\Gamma \boldsymbol{n} \right) \frac{dx_n}{dt} \right] d\Gamma + \int_\Psi f_\Gamma (\boldsymbol{v} \cdot \boldsymbol{n_\Psi}) d\Psi \quad (2.9)$$

where $\boldsymbol{n}_\Psi$ is the normalized normal vector on the surface edge Ψ, $-\mathrm{div}_\Gamma$ reflects the change of the curved surface area when it is modified in the normal direction $\boldsymbol{n}$, see Fig. 2.5. Similar to volume problems, the surface design density f_Γ is time-invariant, $\partial f_\Gamma / \partial t = 0$.

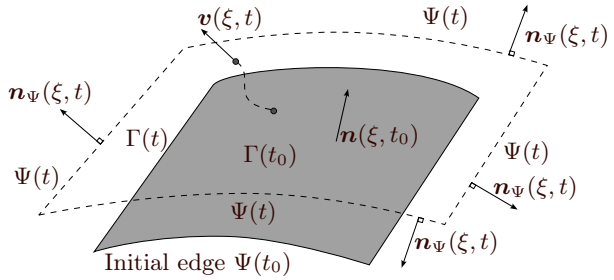


Figure 2.5: Surface optimization problem

It is possible to obtain the equivalent to eq. 2.5, if all design evolutions are in surface normal direction, $\int_\Psi f_\Gamma (\boldsymbol{v} \cdot \boldsymbol{n}_\Psi) d\Psi = 0$:

$$\frac{dF(t)}{dt} = \int_{\Gamma(t)} \left(\frac{\partial f_\Gamma}{\partial \boldsymbol{x_n}} + f_\Gamma \ \mathrm{div}_\Gamma \boldsymbol{n} \right) \frac{dx_n}{dt} d\Gamma = \int_{\Gamma(t)} \frac{df_\Gamma}{d\boldsymbol{x_n}} \frac{dx_n}{dt} d\Gamma \quad (2.10)$$

2.6 Design gradients

The optimizer modifies the design boundary Γ by varying the control field $\boldsymbol{p}$. The eq. 2.8 can be extended as follows:

$$\frac{dF(t)}{dt} = \int_{\Gamma(t)} \frac{df_\Gamma}{d\boldsymbol{x}_n} \frac{d\boldsymbol{x}_n}{dt} d\Gamma = \int_{\Gamma(t)} \frac{df_\Gamma}{d\boldsymbol{p}} \frac{d\boldsymbol{p}}{d\boldsymbol{x}_n} \frac{d\boldsymbol{x}_n}{dt} d\Gamma \quad (2.11)$$

where $df_\Gamma / d\boldsymbol{p}$ and $df_\Gamma / d\boldsymbol{x}_n$ are *design* and *shape gradients* of the objective function F. Again, in the eq. 2.11, only the projected on the surface normals shape gradients are considered. This information has to be calculated at each

material point at every pseudo-time step. Consequently, if this information is available, the gradient-based optimizers are applied to solve an optimization problem. The difference between shape and design gradients is called *filtering error* and it can be controlled by *filtering intensity*, Firl et al. [19].

2.7 Relation between design controls and surface coordinates

In a general shape optimization problem, the relation between the design control field and design surface can be formulated in continuous form as follows:

$$\boldsymbol{x}(\xi_0) = \int_\Gamma A(\xi_0, \boldsymbol{p})\boldsymbol{p}(\xi)d\Gamma \tag{2.12}$$

where A is a filtering operator that defines the relation between spatial coordinates $\boldsymbol{x}$ and the design control parameter $\boldsymbol{p}$. The name comes from the fact that this operator is used to regularize the optimization problem and helps to avoid final design solutions with rough and kinky design surfaces. In some publications, it is also called a mapping operator because it is used to map the shape derivatives from the design space to the design control space as the design derivates, Najian Asl [44]. The operator A is also known as *shape Hessian*.

2.8 Design controls as surface coordinates

In the shape optimization, where the design control field is represented as spatial coordinates of the material point ξ, the relation between the spatial coordinates $\boldsymbol{x}_0$ of the material point ξ_0 can be formulated in continuous form as follows:

$$\begin{aligned} \boldsymbol{x}(\xi_0) &= \int_\Gamma A(d_0)\boldsymbol{p}(\xi)d\Gamma \\ d_0 &= \|\xi_0 - \xi\| \end{aligned} \tag{2.13}$$

where d is Euclidian distance between two material points $[\xi_0], [\xi]$. The filtering operation A has a positive value inside a filtering radius and zero elsewhere. The filter function $A(d_{ij})$ reflects the filtering effect between material points i and j:

$$\begin{aligned} A(d_{ij}) &= \begin{cases} A \geq 0 : d_{ij} \leq r; \\ A = 0 : \text{ elsewhere}; \end{cases} \\ d_{ij} &= \|\xi_i - \xi_j\| \\ \int_\Gamma A(d_i)d\Gamma &= 1 \\ d_i &= \|\xi_i - \xi\| \end{aligned} \tag{2.14}$$

The derivative of the spatial coordinate $\boldsymbol{x}$ at the point ξ_i with respect to design control $\boldsymbol{p}$ at the point ξ_j:

$$\frac{d\boldsymbol{x}(\xi_i)}{d\boldsymbol{p}(\xi_j)} = A(d_{ij}) \tag{2.15}$$

If the filter operator A is symmetric, then:

$$\begin{aligned} A(d_{ij}) &= A(d_{ji}) \\ \frac{d\boldsymbol{x}(\xi_i)}{d\boldsymbol{p}(\xi_j)} &= \frac{d\boldsymbol{x}(\xi_j)}{d\boldsymbol{p}(\xi_i)} \end{aligned} \tag{2.16}$$

Remark. Eq. 2.14 can be extended to have variable radius: $d_{i,j} \leq r_i$, where r_i is a non-constant filter radius size at material point ξ_i.

2.9 Relation between updates in design control and surface coordinates

Using eq.2.11 and eq. 2.13, design gradients at position ξ_0 can be calculated:

$$\frac{df_\Gamma}{d\boldsymbol{p}(\xi_0)} = \int_\Gamma \frac{df_\Gamma}{d\boldsymbol{x}(\xi)} A(d_0) d\Gamma \tag{2.17}$$

while the design velocity from eq. 2.3 can be found as follows:

$$\boldsymbol{v}(\xi_0) = \frac{d\boldsymbol{x}(\xi_0)}{dt} = \frac{d\boldsymbol{x}(\xi_0)}{d\boldsymbol{p}} \frac{\boldsymbol{dp}}{dt} = \int_\Gamma A(d_0) \boldsymbol{v}_c(\xi) d\Gamma \tag{2.18}$$

where the v_c is the control velocity found by the optimization algorithm based on the design gradients $\frac{df_\Gamma}{d\boldsymbol{p}}$. As a result, the filter operation A is applied two times:

1. Map shape gradients from design space onto control space: $\frac{df_\Gamma}{d\boldsymbol{p}} \leftarrow \frac{df_\Gamma}{d\boldsymbol{x}}$. This operation called *backward mapping*;
2. Map control velocity from control space onto design space: $v_c(\xi) \rightarrow v(\xi)$. This operation called *forward mapping*;

2.10 Temporal discretization

In the field of applied physics and mathematics, the governing equations require discretization in both space and time. In the case of a shape optimization problem, the time is represented by pseudo time and, consequently, can be simply discretized as *optimization iterations* ($\Delta t = 1$). This approach conjugates well with iterative optimization algorithms that are usually applied to solve optimization problems.

Shape update $\Delta\boldsymbol{x}$ is a discrete analogue of the continuous *design velocity*:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \Delta\boldsymbol{x} \tag{2.19}$$

and *design control update* $\Delta \boldsymbol{p}$ is the *control velocity*:

$$\Delta \boldsymbol{p} = \alpha \boldsymbol{s} \tag{2.20}$$

where $\boldsymbol{s}$ is a *search direction* and α is a step length.

2.11 Spatial discretization

Space discretization has to be applied to solve structural or CFD problems using finite element (FE) methods. Consequently, it leads to the need to discretize the structural geometry $\boldsymbol{x}^T = [x_1^x, x_1^y, x_1^z, ..., x_n^x, x_n^y, x_n^z]$ and design control geometry $\boldsymbol{p}^T = [p_1^x, p_1^y, p_1^z, ..., p_n^x, p_n^y, p_n^z]$ by standard techniques. In this context, as the design control geometry, we mean the same or different discretization of the design surface, where the control nodes are conjugated with the design control field. Eq. 2.13 appears to be as follows:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{p} \tag{2.21}$$

where $\boldsymbol{x}$ and $\boldsymbol{p}$ are the vectors of spatial coordinates and control parameters of the surface nodes, where the coordinates and control parameters in the spatial x–, y– and z– directions of a node are arranged sequentially. $\boldsymbol{A}$ is a filtering matrix with a size of $n \times m$, where n is a size of vector $\boldsymbol{x}$ and m is size of $\boldsymbol{p}$. Hence, the size of the optimization problem in the classical definition equals to a number of design variables, m. If $n = m$, the design control field provides the largest design space.

Remark. In case the discretization of the control space is finer than the surface mesh, the actual geometry won't be able to represent the shape modes generated by the control space and the optimizer.

In the same way, the relation between shape gradients and design gradients, eq. 2.18 can be reformulated in a discrete form:

$$\frac{df}{d\boldsymbol{p}} = \frac{df}{d\boldsymbol{x}} \frac{d\boldsymbol{x}}{d\boldsymbol{p}} = \boldsymbol{A}^T \frac{df}{d\boldsymbol{x}} \tag{2.22}$$

and the relation between shape update and design control update, 2.17:

$$\Delta \boldsymbol{x} = \boldsymbol{A} \Delta \boldsymbol{p} \tag{2.23}$$

The filtering operator $\boldsymbol{A}$ is applied two times: the first time for gradients and a second time for shape update, identical to the continuous form. Filtering operation doesn't modify the original optimization problem and conserves the change of the objective function in both spaces:

$$\nabla_x \boldsymbol{f}^T \Delta \boldsymbol{x} = (\boldsymbol{A}^{-T} \nabla_p \boldsymbol{f})^T \boldsymbol{A} \Delta \boldsymbol{p} = \nabla_p \boldsymbol{f}^T \boldsymbol{A}^{-1} \boldsymbol{A} \Delta \boldsymbol{p} = \nabla_p \boldsymbol{f}^T \Delta \boldsymbol{p} \tag{2.24}$$

Chapter 3

Adaptive Vertex Morphing

3.1 Vertex Morphing

Vertex Morphing is a successful method that has been introduced by Hojjat et al. [33] and Bletzinger [8]. Furthermore, it has been successfully used for various practical shape optimization applications, Baumgärtner et al. [7], Geiser et al. [25], Geiser A. [26], Ghantasala et al. [28], and Najian Asl et al. [45]. Further details and discussion about the filtering operation $\boldsymbol{A}$ and shape optimization problems with Vertex Morphing are continued in Publication II. Chapter 4 discusses the shape optimization workflow based on Vertex Morphing with various optimization methods.

3.2 Two Roles of the filtering radius

The filtering radius size has two roles: regularization and design variable. In the following section, we discuss both of them and their influence on the outcome.

3.2.1 Design variable

In Vertex Morphing, the filtering radius is considered as a design variable because the filtering radius changes the optimization results dramatically. Fig. 3.1 (left) shows the influence of the filtering radius size on the design gradients. On the left side, one can see the shape gradients of the drag force on the ONERA M6 wing, computed by the SU2 solver. On the right side top, there are various filtering radii sizes, which are compared with wing size. All sizes are valid and generate usable designs. On the bottom right side of the figure, there are obtained design gradients. By comparing the design gradients obtained by different radius sizes, we can observe that with a larger filtering size, the design gradients are smoother but more different compared to the raw shape gradients. In contrast, with a smaller filtering radius size, the design gradients are closer to the shape gradients. As a result, the filtering radius size

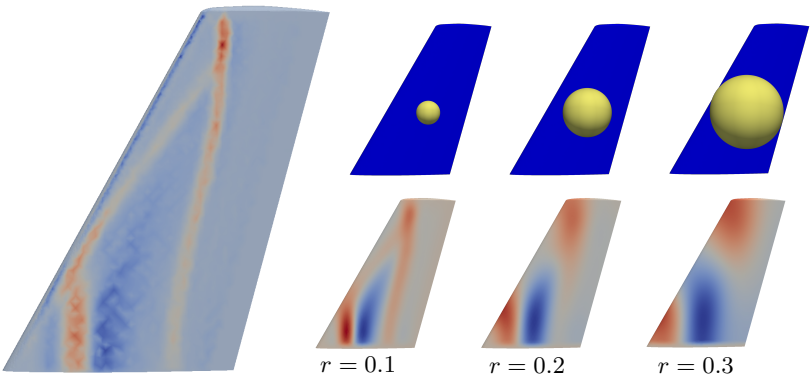


Figure 3.1: Shape gradients (left) vs design gradients (right) w.r.t. different filtering radius sizes.

"guides" an optimizer to a certain local minimum, where the design surface is formed with shape updates with certain shape modes. If the optimization problem is convex, the optimizer will always converge to the "true" minimum, Hojjat [32] and Hojjat et al. [33].

3.2.2 Regularization

The second role of the filtering radius is to regularize the optimization problem. As each node of the control mesh is considered to be a design parameter, there are a large number of design solutions. However, a lot of them are noisy, with non-smooth design surfaces and low-quality meshes. Such solutions are considered as unusable or invalid. Therefore, Vertex Morphing applies filtering operations on shape gradients and design updates to compute shape updates that are considered as *usable*. The *usable shape update* is a shape update that can be applied to the current numerical mesh without generating invalid cells, self-penetrations, and kinks, and it reduces the objective function while the constraints are satisfied for small shape changes.

Fig. 3.2 compares the generated meshes using different filtering radii. The element size of the mesh is approximately 3 mm. For the radius size $r = 2$ mm, no filtering occurs; hence, the generated mesh is extremely noisy and unpractical. If $r = 6$ mm, the computed shape update is smoother, but it is still unusable because it generates a rough surface. If $r = 18$ mm, the shape update is usable, and the generated mesh is smooth. Following Firl et al. [19] and Hojjat [32], the filtering radius size has to cover at least four elements to generate a usable shape update. The Publication II, Section 3, studies the radius size effect on the surface smoothness. In the following Section, we demonstrate and discuss such examples.

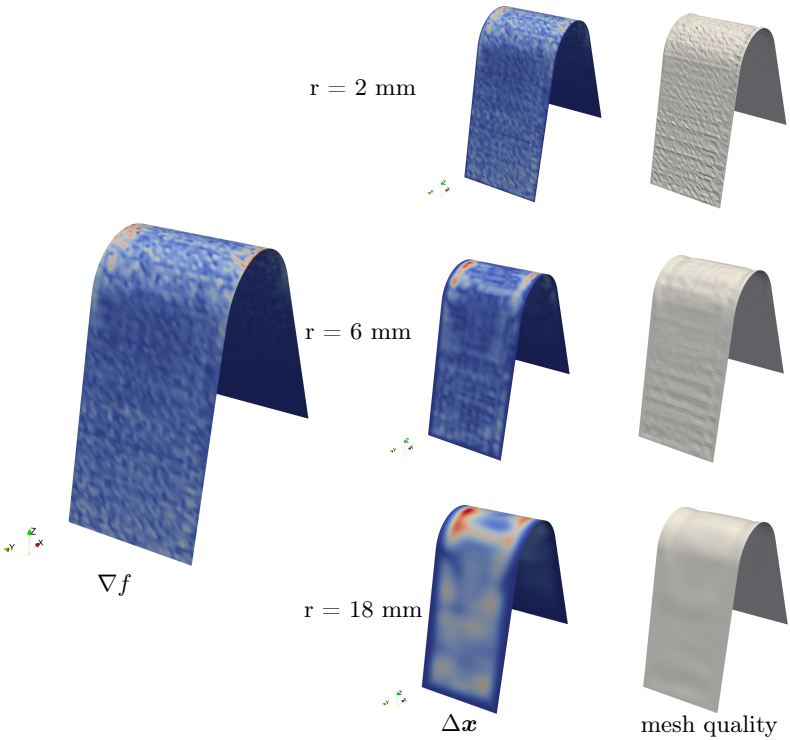


Figure 3.2: Shape gradients (left) vs shape update and mesh quality (right) w.r.t. different filtering radius sizes.

3.3 Vertex Morphing with adaptive filtering radius size

Publication II proposes Vertex Morphing with adaptive filtering radius (AVM method). The idea is based on the Vertex Morphing practice to choose the filtering radius based on the mesh size. The method suggests solutions for three main challenges to choosing the size of the filtering radius in daily practice:

1. A filtering radius size dramatically influences the optimization outcome. Therefore, a user should choose it carefully. AVM provides the flexibility to choose individual filtering radii for each node, hence, exploring the design space with more freedom to find better solutions.

2. A priori, the "good" size of the filtering radius is unknown. If the radius is too small, the filtering operation fails to generate smooth shape updates and, as a result, produces high-frequency, noisy geometries. AVM adjusts the filtering radius locally in the regions where the initial filtering radius

gets too small to ensure that the generated shape updates are usable. Consequently, the radius can be reduced if the local elements have been compressed to conserve the design freedom through the optimization process.

3. A designer may have difficulties choosing a "proper" filtering radius for an unknown, new complex model. AVM can be used without any user input by applying the computed radius field. Based on the obtained results, the designer can adjust the radius sizes for the next design iteration.

3.4 Mesh independency

An important property of any parameterization technique is mesh independency, which can be understood as follows:

"The found optimal shape is independent of FE discretization and depends only on the choice of the design variables."

Firl et al. [19] in his works has shown that the FE-based parameterization with the same filtering radii finds similar solutions with different FE meshes. The original Vertex Morphing has the same property.

In contrast, AVM may find different solutions with different FE meshes. However, AVM can be accepted as mesh-independent parameterization as well because of its two properties:

1. If the radius field is fixed for a certain coarse mesh, and the radius field is applied to a new refined mesh, and then the found solution is going to be qualitatively the same as the solution on the coarse mesh.

2. If the radius field is computed based on a new FE mesh, AVM will find a new solution due to a newly chosen set of the design variables, $\boldsymbol{p} = \boldsymbol{A}^{-1}(\boldsymbol{r})\boldsymbol{x}$.

3.5 Numerical examples with VM and AVM

3.5.1 Structural academic example

A structural optimization problem is solved to demonstrate the flexibility and robustness of the AVM method. The goal is to minimize the compliance of the shell structure under the distributed load. Additionally, the design boundaries are applied, which limits the motion of each surface node, (Chapter 4.2.3):

$$\Delta x_k \leq \Delta x_{max} \tag{3.1}$$

$$\Delta \boldsymbol{x} = \sum_i \Delta \boldsymbol{x}^{(i)} \tag{3.2}$$

The size of the optimization problem is 15198 (3 times the number of nodes). The number of geometrical constraints is 5066 (number of nodes).

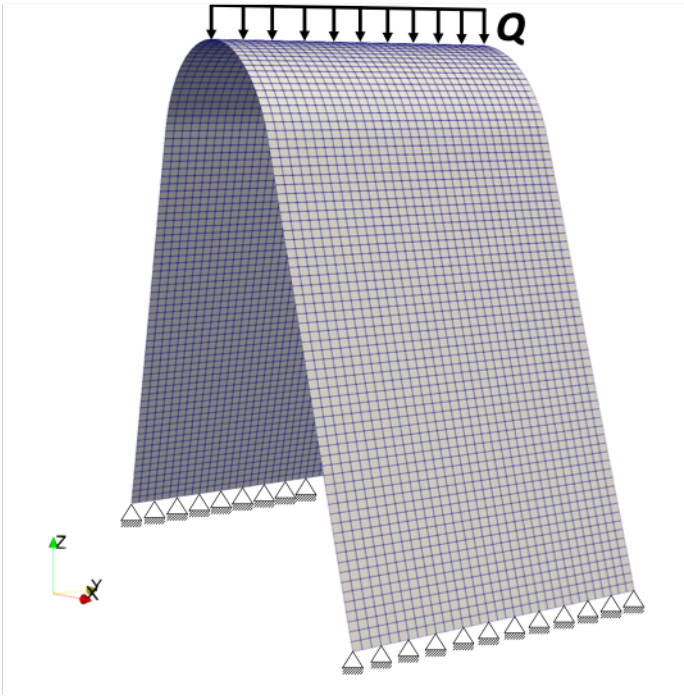


Figure 3.3: Structural FE-model.

The structural analysis model is shown in Fig. 3.3. The distributed load Q is applied in the middle of the structure. On both sides, fixed supports are applied. The convergence criteria are:

$$\sum_{j=i-5}^{j=i} \frac{f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^{(0)})}{f(\boldsymbol{x}^{0})} \leq 1e^{-5} \tag{3.3}$$

$$i \leq 100 \tag{3.4}$$

The numerical model has an initialy structured mesh with a very small variation of the element sizes next to the surface edges. As a result, the initial radius field is almost constant and equals 21 mm. Consequently, in our numerical experiments with Vertex Morphing, the filtering radius $r = 21$ mm.

Case 1, maximum absolute update 4 mm.

The final results, obtained with AVM and VM $r = 21$ mm, represent different local minimums with smooth design surfaces, Fig. 3.4. As the absolute update is limited to 4 mm, there are no large deformations in the surface mesh and elements. Consequently, the initial radius size is suitable during the whole optimization process.

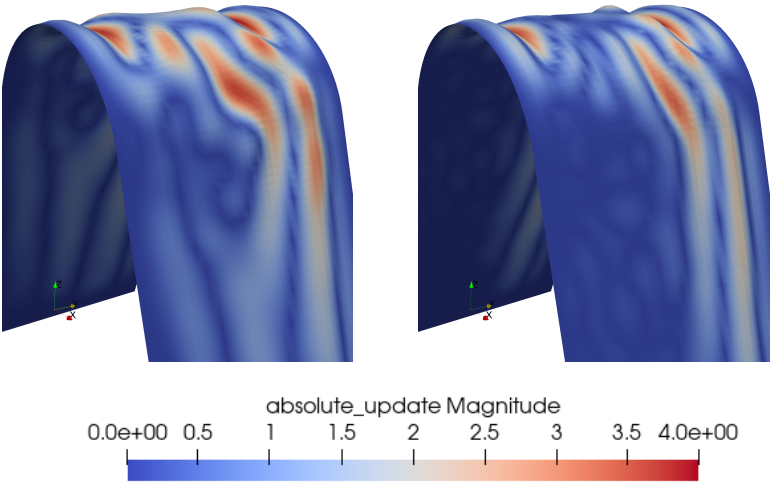


Figure 3.4: Optimized shell structure: AVM (left) and VM, $r = 21$ mm (right). Maximum absolute update 4 mm.

Case 2: maximum absolute update 12 mm.

In contrast to Case 1, the final results with VM and maximum absolute update 12 mm have kinks and a non-smooth design surface, Fig. 3.5 It happens due to enlarged surface deformations and insufficient size of the initial filtering radius. Fig. 3.6 shows the change in the radius field computed by AVM. It can be seen that the radius field has changed dramatically for the regions, where the larges shape changes acquire.

Case 3: AVM with custom radius size and maximum absolute update 12 mm.

In this case, the custom radius size $r = 60$ mm is given on the top of the plate to show the possibility of exploring new designs. In Fig. 3.7 the obtained radius field is shown, and the final solution without high-frequency shape modes is on the top of the structure.

In all shown cases, the optimizer finds different local minima with various performances. However, there is no rule that performance is better in the case of a smaller or bigger filtering radius; see Publication II, the airfoil optimization. Typically, it changes from case to case. Hence, the designer should explore the possibilities by changing the filtering radius globally and locally.

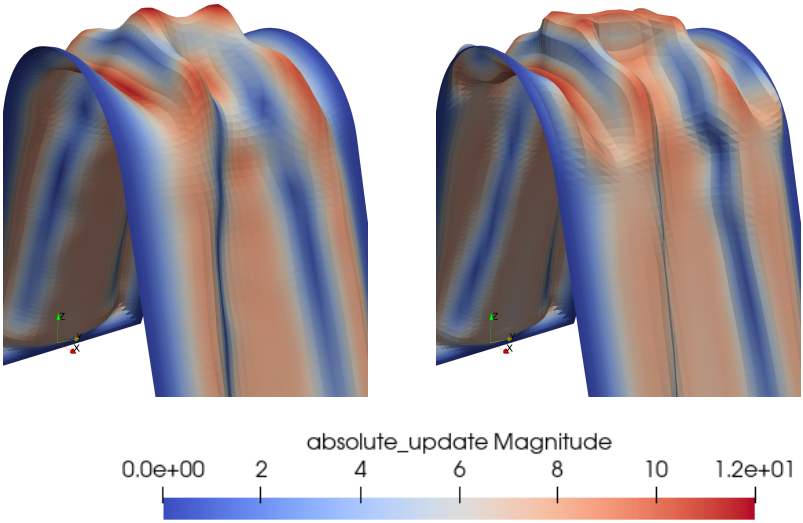


Figure 3.5: Optimized shell structure: AVM (left) and VM, $r = 21$ mm (right). Maximum absolute update 12 mm.

3.6 Design process with AVM

One of the goals of developing the Adaptive Vertex Morphing is to improve the general design workflow with Vertex Morphing. As it is mentioned above, it can be challenging to choose the "right" filtering radius size for the new model. With Adaptive Vertex Morphing, one can start the design process without giving filtering radius size parameters. Based on the obtained results, the designer can choose the regions where the radius should be increased. However, reducing the radius is impossible because AVM computes the smallest acceptable radius based on the local mesh size. If the radius size must be smaller than the computed one, the designer must refine the FE mesh in that region.

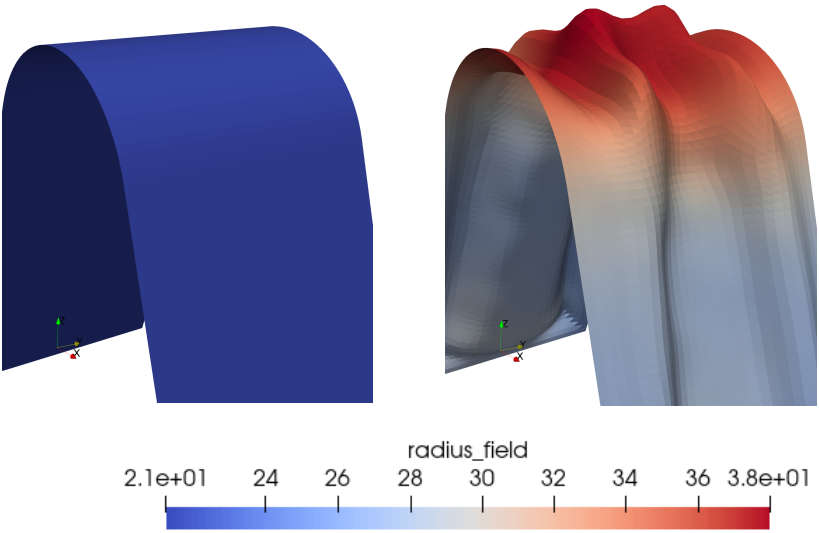


Figure 3.6: Radius field computed by AVM: initial field (left), latest field (right).

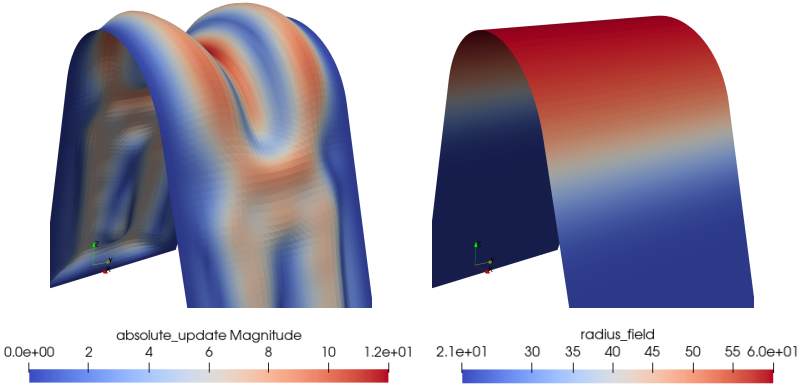


Figure 3.7: Optimized shell structure: AVM and custom radius $r = 60$ mm on top. Maximum absolute update 12 mm.

Chapter 4

Optimization algorithms with Vertex Morphing

4.1 Engineering optimization problem formulation

The design optimization process requires the designer to interpret the goals and design specifications into a mathematical model that can be solved using numerical optimization methods. The challenge of these processes is that it isn't trivial to formulate an adequate optimization problem with a solution. Unrealistic requirements cause the optimization to fail or cause it to converge to a mathematical optimum that is unrealistic from an engineering point of view.

Therefore, following Martins et al. [41], the formulation of the optimization problems starts with the problem description. At this step, all the goals, requirements, and a statement of the system are described. Secondly, all available information, numerical and experimental data, physical aspects, previous models, etc., are collected regarding the performance and behavior of the designed model. The second step is important in the early stages because it helps to understand the task better and formulate a good optimization problem.

The third step defines the design variables that should be found during the optimization process. The choice of the design variables defines the type of optimization to be solved. In topology optimization, one should find the optimal material distribution that can carry the load in a most efficient way. In the size optimization, the engineering parameters are chosen, such as cross-sections, cross-areas, angles of attacks, etc. In shape optimization, a set of the typical design variables are CAD parameters or positions of the surface nodes. Depending on the available data, for instance, the existence of the initial models, or the stage of the design cycle, various types of structural optimization are applied.

The optimization response functions are defined in the last two steps: objective and constraints. The problem formulation process is summarized in

Fig. 4.1. The three last steps are discussed in the following Sections.

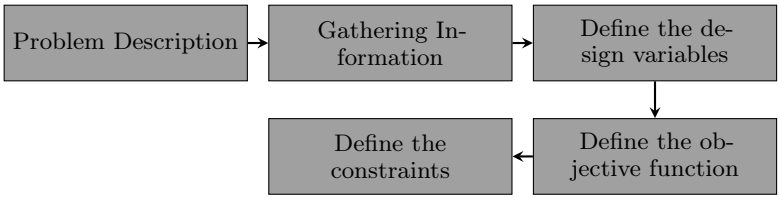


Figure 4.1: Optimization problem formulation

4.1.1 Design variables

The design variables are represented by the column vector $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ and define a given design. Different vectors $\boldsymbol{x}$ define different designs. The size of the vector determines the optimization problem's dimensionality.

The design variables must be independent of each other or any other parameter, and the optimization algorithm must independently control any variable. It means that during the physical analysis of a given design, the variables are the input parameters for the simulations, and they remain constant during the simulation.

Example. In the size optimization of the wing, a wingspan (b) and a chord (c) can be used as design variables. In the practice, an area ($A = bc$) and an aspect ratio ($AR = b^2/S$) are also preferred during the design process. Variables A and AR can't be added to the variables b and c because the variables aren't independent. Instead, any two design variables of four can parameterize the optimization problem (4 variables, 2 dependencies).

There are two types of design variables: continuous and discrete. The continuous variable is usually a real number that can continuously be any value in a given range. In contrast, the discrete variable can take only specified values.

Example. The previous example of the wing's size optimization problems is considered. The continuous set of the design variables, for instance, chord (c) and wingspan (b) which can take any real value. The discrete variable would be a set of specific wing models, for instance, [ONERA M5, ONERA M6] with a fixed pairs [b,c].

4.1.2 Objective and constraint functions

An objective function is a quantity that determines if one design is better than another one. In a standard format, the objective function has to be minimized:

$$\textbf{minimize} : f(\boldsymbol{x}) \tag{4.1}$$

In some applications, the goal function has to be maximized. The equivalent minimization problem can be formulated in two ways:

$$\textbf{maximize} : f(\boldsymbol{x}) = \textbf{minimize} : -f(\boldsymbol{x}) \tag{4.2}$$

or

$$\textbf{maximize} : f(\boldsymbol{x}) = \textbf{minimize} : 1/f(\boldsymbol{x}) \tag{4.3}$$

Eq. 4.2 is preferred because $f(\boldsymbol{x}) = 0$ is possible.

The constraints are classified as equality $h(\boldsymbol{x} = 0)$ and inequality $g(\boldsymbol{x} \leq 0)$. Every equality constraint can be formulated as a pair of inequality ones:

$$h(\boldsymbol{x}) = 0 = \begin{cases} h(\boldsymbol{x}) \leq 0.0 \\ h(\boldsymbol{x}) \geq 0.0 \end{cases} \tag{4.4}$$

In literature, the design bound $x_i^l \leq x_i \leq x_i^u$ is classified as an additional type, where x^l is a lower boundary and x^u – upper boundary. The design variable x_k is then called bounded. If the found optimum solution is $\boldsymbol{x}^*$, then the constraints can be classified as:

1. If $g(\boldsymbol{x}^*) = 0.0$, the constraint is active;
2. If $g(\boldsymbol{x}^*) < 0.0$, the constraint is non-active;
3. $h(\boldsymbol{x}^*) = 0.0$, is always active;

The number of active constraints must be less or equal to the problem's dimensionality. The design point $\boldsymbol{x}$, that satisfies all the given constraints, is called *feasible*, otherwise – *infeasible*.

4.1.3 Standard optimization problem

The optimization problem statement is: "Minimize the objective function by varying the design variables within their bounds such that all constraints are satisfied", Martins et al. [41]. The standard constrained optimization problem can be formulated as follows:

$$\begin{aligned} \textbf{minimize} &: f(\boldsymbol{x}) \\ \textbf{design variables} &: \boldsymbol{x} \\ x_i^l \leq x_i &\leq x_i^u \\ \text{s.t.:} g_j(\boldsymbol{x}) &\leq 0, \text{ where } j = 1..n_g \\ h_k(\boldsymbol{x}) &= 0, \text{ where } k = 1..n_h \end{aligned} \tag{4.5}$$

where f is an objective function that has to be minimized by varying the design parameters $\boldsymbol{x}$, with design bounds x^l and x^u and such that the equality h_k and inequality constraints g_j are satisfied. n_h and n_g are the numbers of equality and inequality constraints respectively.

4.2 Shape optimization problem formulation with Vertex Morphing

4.2.1 Problem formulation

The formulation of a shape optimization problem with Vertex Morphing follows the procedures in Fig. 4.1, where the design variables are already chosen. Nevertheless, the engineer should adjust the parameterization technique to have the desired outcome. In Vertex Morphing, the filtering radius size plays a key role in adjusting the generated shape updates, Chapter 3.

Shape optimization problems with Vertex Morphing are based on the standard definition 4.6, and it can be reformulated as follows:

$$\begin{aligned} \textbf{minimize} &: f(\boldsymbol{x}(\boldsymbol{p}), \boldsymbol{u}(\boldsymbol{p})) \\ \textbf{design variables} &: \boldsymbol{p} \\ x_i^l \leq x_i &\leq x_i^u \\ \text{s.t.:} g_j(\boldsymbol{x}) &\leq 0, \text{ where } j = 1..n_g \\ h_k(\boldsymbol{x}) &= 0, \text{ where } k = 1..n_h \end{aligned} \tag{4.6}$$

where $\boldsymbol{x}$ are spatial coordinates of the design surface nodes, $\boldsymbol{p}$ is the control design field, $\boldsymbol{u}$ are the state variables, x^l and x^u are the design bounds, h_k and g_j are the equality and inequality constraints.

4.2.2 Independency of control field parameters

In Vertex Morphing, the design control field is considered as design variables. The position of the node x is dependent on several control parameters p inside the filtering radius size. In contrast, the control parameters are independent of each other. The optimization algorithm can change each control parameter based on the design gradients information and the line search method. Consequently, the position of the node x appears to be a sum of the control parameter multiplied by influence weights:

$$\Delta x_i = \frac{\sum_j A_{ij} \Delta p_j}{\sum_j A_{ij}} \tag{4.7}$$

4.2.3 Design boundary formulation

In a standard problem formulation, eq. 4.6, the design boundary appears to be a linear constraint that bounds the design variable directly. In Vertex Morphing practice, it is convenient to apply design boundaries to the design nodes and not to the control design parameters. As a result, the constraints are non-linear functions anymore:

$$\boldsymbol{A}_{ij}^{-1} x_i^l \leq p_j \leq \boldsymbol{A}_{ij}^{-1} x_i^u \tag{4.8}$$

where, $\boldsymbol{A}^{-1}$ is an inverted filtering matrix that may be computationally expansive to compute. To overcome this issue, we can reformulate the

boundaries as a legal constraint by computing its value and shape gradients. The constraint for node k restricting its motion in a normal direction is:

$$\begin{aligned} g(x_k) &= \Delta \boldsymbol{x}_k \cdot \boldsymbol{n}_k - \Delta x_k^{max} \\ \Delta \boldsymbol{x}_k &= \sum_i \Delta \boldsymbol{x}_k^{(i)} \end{aligned} \tag{4.9}$$

or restricting the absolute motion:

$$g(x_k) = \|\Delta \boldsymbol{x}_k\| - \Delta x_k^{max} \tag{4.10}$$

As a result, the number of newly formulated constraints is as large as the number of restricted nodes. In order to simplify the optimization problem, the aggregation techniques can be applied, Brelje et al. [10], Damigos et al. [16], and Geiser et al. [25]. If the square-sum aggregation function is applied, the aggregated design boundary constraint is:

$$\begin{aligned} g^+(\boldsymbol{x}^{(i)}) &= \begin{cases} g(x_k^{(i)}) : g(x_k^{(i)}) > 0.0 \\ 0 : \text{ elsewhere} \end{cases} \\ \nabla g(\boldsymbol{x}^{(i)})_{square} &= \sum_k 2 g^+(x_k^{(i)}) \nabla g^+(x_k^{(i)}) \end{aligned} \tag{4.11}$$

Alternatively, the max-value aggregation can be applied for the relaxed gradient projection method, Publication II, Chapter 4.4.

4.2.4 Initial design

To solve a shape optimization problem iteratively, an *initial* or *starting point* should be given. Most of the gradient-based optimization methods, for instance, steepest descent, find a local optimum close to the starting point. Hence, it is recommended to try different starting points to explore the design space better.

In Vertex Morphing, the initial start is the initial geometry and its FE model, which is used to solve the applied physical problem. Equivalently, the initial geometry predefines the local minimum, which the optimizer would find. For instance, in the full-car optimization example to improve the aerodynamic efficiency, can be performed with or without various aerodynamic elements, like rear spoiler. As a result, the outcome from optimization is a different car models, which can be used for different scenarios, in the city or on the racing track.

One should carefully choose the optimization methods and starting points because some methods, like the interior penalty function method, Vanderplaats [55], must have a feasible start.

4.3 Gradient-based optimization methods with Vertex Morphing

The optimization methods are typically classified based on the data that they use to solve the optimization problem:

1. Function values → zero order methods. The well-known methods are Genetic search, Particle swarm, Michalewicz et al. [42]. Usually, the methods are efficient with a small number of design variables $n \leq 20$. These methods aren't considered in this work.

2. Function gradients → first order methods. The methods are: Steepest Descent, Conjugate Gradients, Gradient Projection, Sequential Linear Programming (SLP), Sequential Quadratic Programming (SQP), Augmented Lagrange Method, etc. In engineering problems, sensitivity analysis is required to compute design gradients. Suppose a number of design variables is larger than a number of objective functions. In that case, the adjoint based sensitivity analysis is the most efficient and the state of the art technique for gradient computation, Najian Asl [44]. The first-order optimization methods are the most used ones for engineering shape optimization problems with Vertex Morphing.

3. Second order information, the Hessian matrix → Newton methods. These methods use second-order information to find a design update. The Hessian is typically not available or expensive to compute in engineering optimization. In this case, the method approximates the Hessian matrix, and they are called the Quasi-Newton method. The well-known algorithm is Broyden–Fletcher–Goldfarb–Shanno (BFGS).

In Vertex Morphing, the number of design variables is usually thousands or millions. Therefore, gradient-based optimization methods are preferred.

4.3.1 Steepest descent technique

The most simple but at the same robust optimization technique to solve an engineering optimization problem is the steepest descent. The search direction is the negative objective gradient:

$$\boldsymbol{s} = -\nabla f(\boldsymbol{x}) \tag{4.12}$$

and the design change is:

$$\Delta \boldsymbol{x}^{(i)} = \alpha^{(i)} \boldsymbol{s}^{(i)} \tag{4.13}$$

where the $\alpha^{(i)}$ step length can be found by different line search techniques or approximation techniques. The new solution point is:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \Delta \boldsymbol{x}^{(i)} \tag{4.14}$$

Algorithm 1 shows the workflow of the steepest descent method with Vertex Morphing. The algorithm solves the optimization problem in the control design

space, and the solution process isn't any different from the standard problem process. Additional necessary steps are forward and backward mapping. During these steps, the shape gradients are mapped to design gradients, and design update is mapped to shape update.

Algorithm 1: Steepest Descent method with Vertex Morphing

Start: $\boldsymbol{x}_0, \alpha, i \leftarrow 0$
while *Optimality criteria are not met* **do**
  Compute filtering matrix: $\boldsymbol{A}$;
  $\nabla f(\boldsymbol{p}^{(i)}) \leftarrow \boldsymbol{A}^T \nabla f(\boldsymbol{x}^{(i)})$;
  $\boldsymbol{s}^{(i)} \leftarrow -\nabla f(\boldsymbol{p}^{(i)})$;
  $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} / \left\| \boldsymbol{s}^{(i)} \right\|$;
  Line Search finds: $\alpha^{(i)}$;
  $\Delta \boldsymbol{p}^{(i)} \leftarrow \alpha^{(i)} * \boldsymbol{s}^{(i)}$;
  $\Delta \boldsymbol{x}^{(i)} \leftarrow \boldsymbol{A} \Delta \boldsymbol{p}^{(i)}$;
  $\boldsymbol{x}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} + \Delta \boldsymbol{x}^{(i)}$;
  $i \leftarrow i + 1$;

The steepest descent has been successfully applied to solve unconstrained optimization problems with Vertex Morphing, Baumgärtner [6], Ertl [17], Hojjat [32], and Najian Asl [44]. It is often used with constant step size, where the design surface is changed by a fixed amount of motion, say a couple of mm every iteration.

4.3.2 Newton's method

Newton's method is the classical second-order method with various modifications to improve its efficiency. Here, we follow Vanderplaats [56] to introduce the method. The method is derived from the second-order Taylor series expansion:

$$f(\boldsymbol{x}^{(i+1)}) \approx f(\boldsymbol{x}^{(i)}) + \nabla f(\boldsymbol{x}^{(i)}) \Delta \boldsymbol{x}^{(i)} + \frac{1}{2} \Delta \boldsymbol{x}^{(i),T} [\boldsymbol{H}(\boldsymbol{x}^{(i)})] \Delta \boldsymbol{x}^{(i)} \tag{4.15}$$

where

$$\Delta \boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i)} \tag{4.16}$$

Solving eq. 4.15 for the stationary condition leads to:

$$\Delta \boldsymbol{x}^{(i)} = -[\boldsymbol{H}(\boldsymbol{x}^{(i)})]^{-1} \nabla f(\boldsymbol{x}^{(i)}) \tag{4.17}$$

Eq. 4.17 can be formulated using first-order update schema as:

$$\Delta \boldsymbol{x}^{(i)} = \alpha^{(i)} \boldsymbol{s}^{(i)} = -[\boldsymbol{H}(\boldsymbol{x}^{(i)})]^{-1} \nabla f(\boldsymbol{x}^{(i)}) \tag{4.18}$$

Considering $\alpha = 1$, it gives:

$$\boldsymbol{s}^{(i)} = -[\boldsymbol{H}(\boldsymbol{x}^{(i)})]^{-1}\nabla f(\boldsymbol{x}^{(i)}) \tag{4.19}$$

Inverting the Hessian matrix can be an expensive process for large $\boldsymbol{H}$. Hence, the linear system of equation is solved to find the search direction $\boldsymbol{s}$ that can be used in a general one-dimensional search:

$$[\boldsymbol{H}(\boldsymbol{x}^{(i)})]\boldsymbol{s}^{(i)} = -\nabla f(\boldsymbol{x}^{(i)}) \tag{4.20}$$

In a quadratic optimization problem and $\alpha = 1$, the Newton method converges in one iteration. There are two common modifications to the Newton methods to improve computational efficiency. The first improvement is based on the assumption, that the Hessian doesn't change strongly after each iteration and it is not required to compute it every step. This assumption allows saving the computational time, which is necessary to compute the Hessian matrix. The second modification is to search in direction $\boldsymbol{s}$, noting that $\alpha = 1$ should be an excellent first estimate for $\alpha^{(i)}$.

The principal difficulty with Newton's method is that the $\boldsymbol{H}$ matrix may be singular, or at least non-positive, as required. The $\boldsymbol{H}$ matrix is singular if the objective is linear in one or more design variables. If the objective function is close to being linear in some variables, the computed search direction $\boldsymbol{s}$ may become so ill-conditioned that the result won't be valid. On the other hand, the ill-conditioning of the $\boldsymbol{H}$ matrix can be used to identify an unbounded solution.

If the $\boldsymbol{H}$ is not positive definite, the optimization problem is not convex. The predicted design updates can be so large as to cause oscillation in the solution. To avoid oscillations, one can apply the move boundaries, e.g., trust region, to prevent unnecessary ill-conditioning and apply the properties of $\boldsymbol{H}$ matrix only locally.

In engineering optimization, the computation of the $\boldsymbol{H}$ matrix is often a challenging and computationally expansive task. If the Hessian is available, it should be mapped onto the control space by applying the filtering matrix two times:

$$\boldsymbol{H}_p = \boldsymbol{H}(\boldsymbol{p}) = \boldsymbol{A}\boldsymbol{H}(\boldsymbol{x})\boldsymbol{A}^T \tag{4.21}$$

Assuming that $\boldsymbol{A}$ and $\boldsymbol{H}$ is invertible, the design update is:

$$\Delta\boldsymbol{p} = -[\boldsymbol{H}_p]^{-1}\nabla_p f = -[\boldsymbol{A}^T\boldsymbol{H}_x\boldsymbol{A}]^{-1}\nabla_x f\boldsymbol{A} \tag{4.22}$$

and the shape update is:

$$\begin{aligned}\Delta\boldsymbol{x} = \boldsymbol{A}\Delta\boldsymbol{p} = -\boldsymbol{A}[\boldsymbol{A}\boldsymbol{H}_x\boldsymbol{A}^T]^{-1}\nabla_x f\boldsymbol{A} = \\ -\boldsymbol{A}[\boldsymbol{A}^{-1}\boldsymbol{H}_x^{-1}\boldsymbol{A}^{-T}]\boldsymbol{A}^T\nabla_x f = \\ \boldsymbol{H}_x^{-1}\nabla_x f\end{aligned} \tag{4.23}$$

As a result, Newton's method cancels the filtering effect of Vertex Morphing, and it makes the method unpractical, Hojjat [32]. As a matter of fact, for the

applications with Vertex Morphing, reported in the previous work, the authors had made very good experience with simple steepest descent techniques using the filter to converge to intentionally selected local minima.

4.3.3 Quasi-Newton methods

The main idea of the Quasi-Newton methods is to approximate the $\boldsymbol{H}$ matrix by using the update strategy at every iteration based on the latest information:

$$\widetilde{\boldsymbol{H}}^{(i+1)} = \widetilde{\boldsymbol{H}}^{(i)} + \Delta\widetilde{\boldsymbol{H}}^{(i)} \tag{4.24}$$

where the update $\Delta\widetilde{\boldsymbol{H}}^{(i)}$ is a function of the last two gradients. The initial Hessian matrix is usually set to the identity matrix or scaled version of it, $\widetilde{\boldsymbol{H}}^{(0)} = \boldsymbol{I}$. The eq. 4.19 is also valid for Quasi-Newton methods:

$$[\widetilde{\boldsymbol{H}}^{(i)}]\boldsymbol{s}^{(i)} = -\nabla f(\boldsymbol{x}^{(i)}) \tag{4.25}$$

The Quasi-Newton method's most successful update schema is independently developed by Broyden, Fletcher, Goldfarb, and Shanno, Broyden [11], Fletcher [21], Goldfarb [29], and Shanno [53]. Omitting the derivations, the Hessian matrix update is:

$$\Delta\widetilde{\boldsymbol{H}}^{(i)} = \frac{\boldsymbol{y}^{(i)}\boldsymbol{y}^{(i),T}}{\boldsymbol{y}^{(i),T}\boldsymbol{d}^{(i-1)}} - \frac{\widetilde{\boldsymbol{H}}^{(i)}\boldsymbol{d}^{(i-1)}\boldsymbol{d}^{(i-1),T}\widetilde{\boldsymbol{H}}^{(i)}}{\boldsymbol{d}^{(i-1),T}\widetilde{\boldsymbol{H}}^{(i)}\boldsymbol{d}^{(i-1)}} \tag{4.26}$$

where $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i)}) - \nabla f(\boldsymbol{x}^{(i-1)})$ and $\boldsymbol{d}^{(i-1)} = \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i-1)}$. By using the Sherman–Morrison-Woodbury formula, the inverted approximation of the Hessian can be analytically found:

$$[\Delta\widetilde{\boldsymbol{H}}^{(i)}]^{-1} = (\boldsymbol{I} - \sigma^{(i)}\boldsymbol{d}^{(i-1)}\boldsymbol{y}^{(i),T})[\widetilde{\boldsymbol{H}}^{(i)}]^{-1}(\boldsymbol{I} - \sigma^{(i)}\boldsymbol{y}^{(i)}\boldsymbol{d}^{(i-1)}, T) + \sigma^{(i)}\boldsymbol{d}^{(i-1)}\boldsymbol{d}^{(i-1),T} \tag{4.27}$$

where

$$\sigma^{(i)} = \frac{1}{\boldsymbol{y}^{(i),T}} \tag{4.28}$$

In the case of Vertex Morphing, the approximated Hessian matrix should be computed directly in the control space using smoothed design gradients. In contrast to Newton's method, the filtering effect is not canceled, and the smooth shape updates are computed. With Quasi-Newton methods, the design control update is a discrete field because the objective gradients are multiplied with approximated Hessian matrix. Still, the shape update becomes smooth after forward mapping. Algorithm 2 describes the Quasi-Newton method's workflow with Vertex Morphing.

Algorithm 2: Quasi-Newton methods with Vertex Morphing

Start: $\boldsymbol{x}_0, \alpha, i \leftarrow 0$
while *Optimality criteria are not met* **do**
  Compute filtering matrix: $\boldsymbol{A}$;
  $\nabla f(\boldsymbol{p}^{(i)}) \leftarrow \boldsymbol{A}^T \nabla f(\boldsymbol{x}^{(i)})$;
  Approximate inverse Hessian matrix: $\boldsymbol{H^{-1}}_p(\boldsymbol{p}^{(i)})$;
  $\boldsymbol{s}^{(i)} \leftarrow -[\boldsymbol{H^{-1}}_p(\boldsymbol{p}^{(i)})] \nabla f(\boldsymbol{p}^{(i)})$;
  $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} / \left\| \boldsymbol{s}^{(i)} \right\|$;
  Line Search finds: $\alpha^{(i)}$;
  $\Delta \boldsymbol{p}^{(i)} \leftarrow \alpha^{(i)} * \boldsymbol{s}^{(i)}$;
  $\Delta \boldsymbol{x}^{(i)} \leftarrow \boldsymbol{A} \Delta \boldsymbol{p}^{(i)}$;
  $\boldsymbol{x}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} + \Delta \boldsymbol{x}^{(i)}$;
  $i \leftarrow i + 1$;

4.3.4 Gradient projection method

The feasible search direction methods deal directly with the nonlinearity of the problem. One of these methods is the gradient projection method, Rosen [49]. It can be seen as an extension of the steepest descent for constraint optimization problems. With this method, we will first find a search direction $\boldsymbol{s}$ and then move in this direction to update the $\boldsymbol{x}$. Having determined a search direction, the step length is found using line search techniques. The emphasis is to find a search direction that improves the objective function while maintaining a feasible design.

The found search direction $\boldsymbol{s}$ should satisfy two conditions: feasibility and usability. The usability condition shows if the objective function can be reduced in the search direction:

$$\nabla f(\boldsymbol{x})^T \boldsymbol{s} \leq 0.0 \tag{4.29}$$

A direction is called feasible if, for some small move in that direction, the active constraints will not be violated. That can be formulated as follows:

$$\nabla g(\boldsymbol{x})^T \boldsymbol{s} \leq 0.0 \tag{4.30}$$

Observing these two conditions, the search direction finds the greatest objective function reduction if the eq. 4.29 is minimized, with eq. 4.29 is equal to zero. The gradient projection method finds the search direction that satisfies both conditions by projecting the steepest-descent search direction onto the subspace tangent to the active constraints. Publication I introduces the derivation of the formulas and discusses the common problems, such as zig-zagging.

In node-based shape optimization with Vertex Morphing, the gradient projection has been often used in the previous work, Baumgärtner [6], Ertl

[17], Najian Asl [44], and Najian Asl et al. [45]. Algorithm 3 describes the optimization workflow of the method with Vertex Morphing. In contrast to algorithm 1, the constraint's gradients are also mapped to the control space, but the general process is similar.

Algorithm 3: Gradient projection method with Vertex Morphing

Start: $\boldsymbol{x}_0, \alpha, i \leftarrow 0$
while *Optimality criteria are not met* **do**
  Compute filtering matrix: $\boldsymbol{A}$;
  $\nabla f(\boldsymbol{p}^{(i)}) \leftarrow \boldsymbol{A}^T \nabla f(\boldsymbol{x}^{(i)})$;
  $\nabla g_j(\boldsymbol{p}^{(i)}) \leftarrow \boldsymbol{A}^T \nabla g_j(\boldsymbol{x}^{(i)})$;
  Build active constraint matrix: $\boldsymbol{N}$;
  $\boldsymbol{s}^{(i)} \leftarrow [\boldsymbol{I} - \boldsymbol{N}(\boldsymbol{N}^T\boldsymbol{N})^{-1}\boldsymbol{N}^T]\nabla f(\boldsymbol{p}^{(i)})$;
  $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} / \left\| \boldsymbol{s}^{(i)} \right\|$;
  Line Search finds: $\alpha^{(i)}$;
  $\Delta\boldsymbol{p}^{(i)} \leftarrow \alpha^{(i)} * \boldsymbol{s}^{(i)}$;
  $\Delta\boldsymbol{x}^{(i)} \leftarrow \boldsymbol{A}\Delta\boldsymbol{p}^{(i)}$;
  $\boldsymbol{x}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} + \Delta\boldsymbol{x}^{(i)}$;
  $i \leftarrow i + 1$;

4.4 Relaxed gradient projection method

Publication I introduces a relaxed gradient projection method. The method is a modification of the classical Rosen's gradient projection algorithm (Rosen [49, 50]). The proposed algorithm can be classified as:

- First order. The method uses only the first-order information: response functions values and gradients.
- Direct. In contrast to SUMT, the method solves the constraint problem directly by finding feasible search directions and following active constraint boundaries. The optimization process can have a non-feasible start.
- Active set method. The algorithm uses the technique to identify active and non-active constraints at each optimization iteration.
- Feasible direction method. The algorithm tries to find a feasible direction to avoid constraint violations. In contrast to the original method, the proposed one doesn't guarantee the usability of the search direction if there is a highly violated constraint or a strongly infeasible start. The usability condition, eq. 4.29, is not forced.

In contrast to the original method, the relaxed gradient projection method can define the transient stage between active and non-active constraint status, where the constraint is considered active but with a relaxation coefficient. Therefore, the relaxation and correction factors mildly control the projection and correction components of the search direction. The proposed method contains the buffer (critical) zone around the constraints limit value. As a result, the algorithm is more stable with respect to zigzagging behavior when it follows the design boundaries. The proposed method can activate the constraint before the limit value is reached. It doesn't require accurate parameter setup, therefore, it is easier and more robust in daily practice. The algorithm can efficiently solve optimization problems with engineering or manufacturing constraints. The requirements for the constraint functions are:

- The response function should have well-defined values that define the system's performance to estimate the gap to the limit value or needed correction.
- Due to the natural complexity of the manufacturing constraints, it may be hard to define the function and its gradients mathematically. As a result, the response may use normal vectors as functional gradients at the nodes, which doesn't satisfy the feasibility condition. The method can solve problems with such constraints if the response value is well-defined.

4.5 Globalization strategies

Engineering problems require iterative methods to find minima based only on the function values and gradients. To ensure convergence to an optimum, we need a *globalization strategy*. *Globalization* enhances the optimization method, that it converges from any starting point in the domain to a local minimum. It can be understood as the method to globalize the local information at the point to a domain's region. Here, we don't discuss the strategies to find a global optimum, which is a different problem. The are two different globalization strategies: *trust-region* and *line search*.

Line search techniques consist of three steps:

1. Compute a search direction about the current point.
2. Using line search techniques, find a step length along the search direction.
3. Move to the new point, and update all the values.

It also can be formulated as a 1D optimization problem, where the step length is unknown. If the search direction $\boldsymbol{s}$ is found, the step length can be found as:

$$\min_{\alpha} : \ f(\boldsymbol{x} + \alpha \boldsymbol{s}) \tag{4.31}$$

Hence, firstly, we find a search direction, and then we decide how far we should move along it.

The trust-region strategy consists of three steps:

1. Create a model about the current point based on a Taylor series approximation or surrogate model.
2. Solve the created model within the trust region to find a new point.
3. Move to the new point, update all the values and the size of the trust region.

In contrast to line search, in the trust-region strategy, we simultaneously solve for the search direction and the length. In literature, the line search approaches are commonly used to solve nonlinear problems, and we focus on line search in this section.

4.5.1 Constant scaled step length

In shape optimization problem with Vertex Morphing, the constant step size has been used in various research works, Baumgärtner [6], Chen [13], Ertl [17], Hojjat [32], and Najian Asl [44]. It attracts with its simplicity and robustness. The scaled step size means:

$$\tilde{\alpha} = \frac{\alpha}{\|\boldsymbol{s}\|} \tag{4.32}$$

where α is the desired size of the shape update, say 5 mm. The scaled step size ensures that the shape update at each iteration remains constant throughout the optimization process. The drawbacks of constant step size are:

1. *Unknown.* "Good" step size is unknown in the beginning and may be difficult to guess.
2. *Accuracy.* The constant step size might be suitable at the beginning of the optimization process, but not valid later. For instance, when several non-linear constraints activate, the step length might be too large, but using a smaller step size leads to more optimization iterations till the same point.
3. *Zig-zagging.* When the optimizer gets close to the local minimum, it may oscillate around it due to the fixed step size.

4.5.2 Backtracking techniques

In the work of Martins et al. [41], the authors recommend the backtracking techniques as a robust and efficient method for engineering optimization problems. The robustness is guaranteed by trying several step sizes until the "good enough" is found. In their book, the authors introduce techniques to find the step size most efficiently. In this work, we will present the main formulas and ideas of the methods, omitting the derivations.

To simplify the formulas, we will use the notation:

$$\begin{aligned} \phi(\alpha) &= f(\boldsymbol{x} + \alpha \boldsymbol{s}) \\ \phi'(\alpha) &= \nabla f(\boldsymbol{x} + \alpha \boldsymbol{s})^T \boldsymbol{s} \end{aligned} \tag{4.33}$$

where $\phi(\alpha)$ is a value of the objective function f at the point $\boldsymbol{x} + \alpha\boldsymbol{s}$, $\phi'(\alpha)$ is a slope of the 1D function ϕ at the point $\boldsymbol{x} + \alpha\boldsymbol{s}$ with respect to α.

The "good enough" step size can be described using the sufficient decrease condition, also known as *Armijo condition*:

$$\phi(\alpha) = \phi(0) + \mu_1 \alpha \phi'(0) \tag{4.34}$$

where μ_1 is a constant, $0 < \mu_1 < 1$. The quantity $\mu_1\alpha\phi'(0)$ represents the expected decrease of f for a given α. In practice, μ_1 is typically several orders of magnitude smaller than 1, for instance, $\mu_1 = 1e^{-4}$. To satisfy the condition, eq. 4.29, $\phi'(0) = \nabla f(\boldsymbol{x})\boldsymbol{s} < 0$, while the $\boldsymbol{s}$ is always a descent direction, the step size has to be positive, $\alpha > 0$.

The simple backtracking algorithm tries the initial guess α_0 and reduces it by constant $\rho < 1$, till the Armijo condition is fulfilled. The typical value of constant $\rho = 0.5$. There are two scenarios when the technique fails:

1. If the initial guess is too large, and the acceptable step is several orders of magnitude lower, the backtracking techniques require a large number of function evaluations.

2. If the initial guess is too low and it is accepted immediately. However, the function's slope is still negative, and the function can be further reduced with a larger step size.

There are enhanced techniques that deal with these scenarios much more efficiently. It seems to be an excellent solution to increase μ_1 to prevent small steps. Nevertheless, increasing μ_1 also prevents accepting large steps that results in a reasonable decrease because the term $\mu_1\alpha\phi'(0)$ gets very large and harder to satisfy. As a result, the method's convergence may slow down, because the large steps that provide reasonable decrease are desirable and lead to faster convergence. Simultaneously, it would be nice to analyze the initial step better to understand if it is too large or too small. A solution for both issues is to compare the function's slope at the candidate point with the slope at the start point. The condition is called *sufficient curvature condition* and can be formulated as follows:

$$|\phi'(\alpha)| \leq \mu_2 |\phi'(0)| \tag{4.35}$$

This condition requires that the magnitude of the slope at the candidate point is lower than the magnitude of the slope at the starting point by a constant $0 < \mu_2 < 1$. The condition can be understood as "flattening" of the slope at the candidate point. If $\mu \to 0$, the condition requires an exact line search solution. If $\mu_2 \to 1$, we accept larger magnitudes of the slope. Typical values for μ_2 are in the range $[0.1, 0.9]$, and the best value depends on the problem and applied optimization method.

Combining the *Armijo condition* and *sufficient curvature condition* gives the *strong Wolfe conditions*. To guarantee that there are steps that satisfy the strong Wolfe conditions, the constants must be $0 < \mu_1 < \mu_2 < 1$. The line search techniques that find a step satisfying the strong Wolfe conditions have two phases:

1. The *bracketing* phase finds boundaries of the interval where there is an acceptable step size.
2. The *pinpointing* phase finds a step size that satisfies the strong Wolfe conditions within the interval provided by the bracketing phase.

Overall, the bracketing algorithm increases the step size until it either finds an interval that must contain a point satisfying the strong Wolfe conditions or a point that already meets those conditions. In the pinpointing phase, the polynomial approximation techniques are discussed in the following Section.

4.5.3 Polynomial approximations

A polynomial approximation is one of the most efficient techniques for finding the minimum of the one-dimensional function, Vanderplaats [56]. Here, we follow Martins et al. [41] to introduce the formulas to compute the polynomial approximation for the line search problem.

The Polynomial approximation's procedure is to evaluate the function at several points, or derivatives of the function, and fit the approximated polynomial to those known points. The third-order approximating polynomial for line search is:

$$p(\alpha) = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 \tag{4.36}$$

where the coefficients a_0, a_1, a_2, a_3 are the unknowns, that have to be found. The evaluated values and derivatives at the points give us the boundary conditions to find the unknown coefficients. In the case of two-point quadratic approximation:

$$\begin{aligned} p(\alpha_1) &= a_0 + a_1\alpha_1 + a_2\alpha_1^2 \\ p(\alpha_2) &= a_0 + a_2\alpha_2 + a_2\alpha_2^2 \\ p'(\alpha_1) &= a_1 + 2a_2\alpha_1 \\ a_3 &= 0 \end{aligned} \tag{4.37}$$

By solving this equation, we can find the coefficients for quadratic polynomials. Once the coefficients are found, the minimum of the quadratic can be found analytically by finding α^*, where $p'(\alpha^*) = 0$. The solution is $\alpha^* = -2a_1/2a_2$ or it can be reformulated using the boundary conditions:

$$\alpha^* = \frac{2\alpha_1[p(\alpha_2) - p(\alpha_1)] + p'(\alpha_1)(\alpha_1^2 - \alpha_2^2)}{2[p(\alpha_2) - p(\alpha_1) + p'(\alpha_1)(\alpha_1 - \alpha_2)]} \tag{4.38}$$

If computing the gradients of the function at one more point α_2 is inexpensive or it is already evaluated, the cubic interpolation can be performed with extended boundary conditions:

$$\begin{aligned} p(\alpha_1) &= a_0 + a_1\alpha_1 + a_2\alpha_1^2 + a_3\alpha_1^3 \\ p(\alpha_2) &= a_0 + a_2\alpha_2 + a_2\alpha_2^2 + a_3\alpha_2^3 \\ p'(\alpha_1) &= a_1 + 2a_2\alpha_1 + 3a_3\alpha_1^2 \\ p'(\alpha_2) &= a_1 + 2a_2\alpha_2 + 3a_3\alpha_2^2 \end{aligned} \tag{4.39}$$

Using these four equations, the unknown coefficients can be found. In contrast to quadratic case, there are two solution α^* which satisfy the stationary condition $p'(\alpha^*) = a_1 + 2a_2\alpha^* + 3a_3\alpha^{*2}$. However, only one solution is valid to our problem, for which the curvature is positive, $p''(\alpha^*) = 2a_2 + 6a_3\alpha^* > 0$. As a result, the minimum solution is:

$$\alpha^* = \alpha_2 - (\alpha_2 - \alpha_1)\frac{p'(\alpha_2) + \beta_2 - \beta_1}{p'(\alpha_2) - p'(\alpha_1) + 2\beta_2} \tag{4.40}$$

where

$$\begin{aligned} \beta_1 &= p'(\alpha_1) + p'(\alpha_2) - 3\frac{p(\alpha_1) - p(\alpha_2)}{\alpha_1 - \alpha_2} \\ \beta_2 &= \text{sign}(\alpha_2 - \alpha_1)\sqrt{\beta_1^2 - p'(\alpha_1)p'(\alpha_2)} \end{aligned} \tag{4.41}$$

Note. The polynomial approximation is a very powerful tool. It can be used for pinpointing phase of the line search algorithm to find a point that satisfies the strong Wolfe conditions, as it is shown above. Additionally, the polynomial approximation can be applied to solve the line search problem 4.31 directly as a line search technique.

The backtracking and polynomial approximation techniques generally share the same drawback for large engineering problems. It is a necessity to run the primal and adjoint analysis a few times in one optimization iteration. It might be more efficient to perform several optimization iterations with small steps rather than perform the line search process.

4.5.4 Barzilai-Borwein method

Barzilai-Borwein method is the approximation technique to estimate the step length for the steepest descent algorithm, Barzilai et al. [5]. This method attracts many researchers with its simplicity and superior performance in various problems, Fletcher [23]. In the work of Raydan [48], the author has used the Barzilai-Borwein method to estimate the initial step for the backtracking algorithm. In Chapter 5, we study the method in the context of node-based shape optimization and propose the modification for large-scale problems.

4.6 Convergence criteria

A critical part of the overall process is determining when to stop the search. The convergence or termination criteria have a major effect on efficiency and reliability. This section introduces the mathematical definition of the optimum point and several practical convergence criteria.

4.6.1 The Karush-Kuhn-Tucker conditions

The Kuhn-Tucker conditions are the necessary conditions for optimality. Considering the constrained optimization problem, eq. 4.6. The Lagrangian

function, including both equality and inequality constraints, is then:

$$L(\boldsymbol{x}, \boldsymbol{\lambda},) = f(\boldsymbol{x}) + \sum_j \lambda_j g_j(\boldsymbol{x}) + \sum_k h_k(\boldsymbol{x}) \tag{4.42}$$

Only active constraints can be considered at the optimum in the Lagrangian function. The inequality constraint g_j is active if $g_j = 0$ and it is inactive if $g_j < 0$. In advance, it is unknown which constraints are active; therefore we need to include all inequality constraints. To represent inequality constraints in the Lagrangian, it can be defined as equality:

$$g_j + s_j^2 = 0.0 \tag{4.43}$$

where s_j is a new unknown associated with each inequality constraint called a *slack variable*, Martins et al. [41]. Eq. 4.43 can only be satisfied when $g_j \leq 0$. The importance of the squared slack variable is that it takes only positive values and that when $s_j = 0$, the corresponding constraint $g_j = 0$, and when $s_j \neq 0$, the corresponding constraint is nonactive $g_j \leq 0$. The modified Lagrangian function is:

$$L(\boldsymbol{x}, \boldsymbol{\lambda},) = f(\boldsymbol{x}) + \sum_j \lambda_j (g_j(\boldsymbol{x}) + s \odot s) + \sum_k h_k(\boldsymbol{x}) \tag{4.44}$$

where $\odot$ is a element-wise multiplication of s. To derive the first-order optimality conditions, we need to find the stationary point of the Lagrangian function:

$$\begin{aligned} \nabla_x L &= 0 \\ \nabla_\lambda L &= 0 \\ \nabla_s L &= 0 \end{aligned} \tag{4.45}$$

Taking partial derivatives of the Lagrangian with respect to design variables, we get:

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_j \lambda_j \frac{\partial g_j}{\partial x_i} + \sum_k \lambda_k \frac{\partial h_k}{\partial x_i} = 0 \tag{4.46}$$

with respect to Lagrange multipliers associated with equality constraints:

$$\frac{\partial L}{\partial \lambda_k} = h_k = 0 \tag{4.47}$$

with respect to Lagrange multipliers associated with inequality constraints:

$$\frac{\partial L}{\partial \lambda_j} = g_j + s_j^2 = 0 \tag{4.48}$$

Finally, differentiating the Lagrangian with respect to the slack variables, we get:

$$\frac{\partial L}{\partial s_j} = 2\lambda_j s_j = 0 \tag{4.49}$$

which is called the *complementary slackness condition*. This condition insures that for each inequality constraint, if $s_j = 0$ then $\lambda_j > 0$, or if $s_j > 0$ then $\lambda_j = 0$. The complementary condition helps to distinguish between active and nonactive inequality constraints. Unfortunately, the complementary slackness condition leads to a combinatorial problem, and its complexity grows with the number of inequality constraints. One can apply the active-set methods, such as gradient projection to avoid using the slack variables. It constructs the set of active constraints at every iteration and includes only active constraints to the Lagrangian function.

The so-called *Karush-Kuhn-Tucker* (KKT) conditions can be summarized as:

$$\begin{aligned} \nabla f + \lambda_j \nabla g_j + \lambda_k g_k &= 0 \\ h_k &= 0 \\ g_j + s_j^2 &= 0 \\ \lambda_j s_j &= 0 \\ \lambda_j &\geq 0 \end{aligned} \tag{4.50}$$

These first-order conditions are necessary but not sufficient. The second-order conditions require that the Hessian of the Lagrangian must be positive definite in all feasible directions $\boldsymbol{s}$:

$$\begin{aligned} \boldsymbol{s}^T \boldsymbol{H}_L \boldsymbol{s} &> 0 \\ \nabla h \boldsymbol{s} &= 0 \\ \nabla g \boldsymbol{s} &\leq 0 \end{aligned} \tag{4.51}$$

These conditions require positive definiteness in the intersection of the nullspace of the equality constraint Jacobian with the feasibility cone of the active inequality constraints, Martins et al. [41]. In the practical application, the $\boldsymbol{H}_l$ is usually not available, therefore, only first-order conditions are checked, or more pragmatic criteria, are discussed in the following sections.

4.6.2 Maximum number of iterations

The simplest and most commonly used criteria to terminate when the current optimization iteration count reaches the maximum value. This criterion ensures that the optimization process isn't going to process indefinitely, or it helps to control the computational time better. This rule applies to any iterative process, whether the optimization algorithm, solving a coupled physical problem, or functional evaluation.

4.6.3 Absolute or Relative change in the objective function

The second termination condition which should be used is a check on the optimization process. Here, two conditions can be used to identify if the improvement of the objective function is slow enough to stop. First is to compare the absolute value of $f(\boldsymbol{x})$ on successive iterations:

$$|f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i-1)})| \leq \epsilon_A \tag{4.52}$$

where ϵ_A is a specialized tolerance, which can be a constant, say $\epsilon_A = 1e^{-5}$, or relative to the initial value, $\epsilon_A = 0.001 f(\boldsymbol{x}^{(0)})$. The other criterion is a relative change, that checks the relative difference on successive iterations:

$$\frac{|f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i-1)})|}{max[|f(\boldsymbol{x}^{(i)})|, 1e^{-5}]} \leq \epsilon_R \tag{4.53}$$

This criteria ensures that the optimization process is stopped for large or small values of f. If either criterion is satisfied, we define this as a convergence. The requirements can be forced to be satisfied on several successive iterations to avoid a situation when the improvement has slowed down for a couple of iterations and speeds up in the following steps.

4.6.4 Averaged absolute improvement rate condition

In practice, the objective values may strongly oscillate due to a non-smooth function, optimization algorithm, or line search technique. Therefore, we can average the improvement rate from the last N iterations to smooth value oscillations and obtain a more stable improvement rate measurement. For instance, the *averaged change of absolute rate* for last N iterations is:

$$\begin{aligned} \sum_{k}^{i-N+1} \frac{\epsilon_k - \epsilon_{k-1}}{N} &\leq \epsilon_{Aai} \\ \epsilon_k &= \frac{f(\boldsymbol{x}^k) - f(\boldsymbol{x}^0)}{f(\boldsymbol{x}^0)} \end{aligned} \tag{4.54}$$

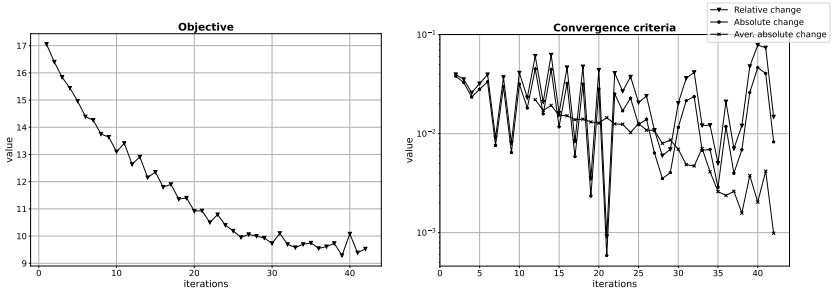
where N is a number of last successive iterations, ϵ_k is an absolute change rate. The number of averaged iterations N is constant, say $N = 10$. ϵ_{Aai} is the required average improvement rate for one iteration, say $\epsilon_{Aai} = 0.001$, which is mean, in every iteration we need to improve at least 0.1% for absolute improvement. In the case of the constraint optimization problem, if the termination criterium is met, one should check the feasibility of the solution point.

4.6.5 Convergence criteria comparison example

In this example, we optimize the simple shell with a point load to minimize the max stress:

$$\min_{\boldsymbol{x}} : \ f(\boldsymbol{x}) = \max_{k}(\sigma_k) \tag{4.55}$$

where σ_k is the *von Mises stress* in the element k. The function is discrete because if the stress is minimized at one place, the maximum stress appears at another place. It is a challenging function not only for the optimization algorithm to minimize but also for the termination condition.



(a) Objective values (b) Convergence measures

Figure 4.2: Comparing the objective values evaluations vs convergence measures

Figure 4.2 compares different convergence measures with respect to the history of objective function values. The absolute and relative change is close to each other, but both criteria have large oscillations, and it is impossible to identify when the optimizer can't further improve. In contrast, the averaged change of absolute rate ($N = 10$) shows monotone behavior with small oscillations.

Chapter 5

Barzilai-Borwein method

Publication II introduces the Quasi-Newton Barzilai-Borwein (QN-BB) method and the combined relaxed gradient projection algorithm with the Quasi-Newton Barzilai-Borwein method (QN-BB-RGP), but they are not well discussed. Therefore, this chapter shows unpublished studies of the BB, QN-BB, and QN-BB-RGP methods with analytical and analysis-based examples.

5.1 Overview on the Barzilai-Borwein method

Constant step size is often used in practical applications due to its simplicity. However, the good constant step size is an unknown *a priori* and may lead to poor performance or higher computational cost. There are various methods to calculate the exact or approximated step length. The method finds a minimum of the objective function or sufficient reduction along the descent search direction. The drawback of these methods is that efficiency improvement is not guaranteed. For instance, Cauchy methods may require calculating the Hessian matrix, which is not always available or very expensive to compute (Zhou et al. [57]). Besides, Armijo's backtracking schemes try several step sizes until the acceptance criteria are satisfied (Ahookhosh et al. [1]). Martins et al. [41] shows the usage of Armijo's backtracking schemes with strong Wolfe conditions as one of the successful practical methods. However, in large optimization problems, additional functional evaluation may excessively increase the computational cost of each optimization iteration.

On the other hand, the Barzilai-Borwein (BB) method (Barzilai et al. [5]) attracts many research groups because of its simplicity and surprising efficiency in unconstrained optimization problems. The method's main advantage is that it doesn't require any costly computational operations to approximate the step size. The original work of Barzilai et al. [5] shows that the method has R-superlinear convergence for 2D quadratic problems. The method performs a way better in comparison to the "classical" steepest descent method with exact line search (Cauchy [12]). In the work of Fletcher (Fletcher [23]), the method has been compared to well-known Conjugated-Gradient methods

(Fletcher et al. [22] and Golub et al. [30]). It has been shown that the Conjugated-Gradient method works better for a large quadratic problem, but both methods are a lot better than "classical" steepest descent. However, in a non-quadratic example where the non-quadratic contribution is small, the Barzilai-Borwein method outperforms the Conjugated-Gradient method and Limited memory BFGS (Liu et al. [39]). In recent years, modified versions of the method to solve unconstrained optimization problems have been proposed: Adaptive Barzilai-Borwein method (Zhou et al. [57]), Stabilized Barzilai-Borwein method (Oleg Burdakov [46]), accelerated Barzilai-Borwein method (Huang et al. [34]).

There are limited results regarding using the Barzilai-Borwein method for constrained problems in the literature. Dai et al. [15] and Raydan [48] have applied the method for optimization problems with box constraints. The common idea is to compute the design update based on the steepest direction of the objective function and project it on the active set of constraints:

$$\boldsymbol{x}^{(i+1)} = P(\boldsymbol{x}^{(i)} - \alpha^{(i)} \nabla f(\boldsymbol{x}^{(i)})) \tag{5.1}$$

In this work, we propose to use Barzilai-Borwein for constraint problems, where the search direction is incorporated into the step size explicitly.

5.2 Original Barzilai-Borwein method

The Barzilai-Borwein (BB) method suggests a step size approximation using current and previous sensitivity information. The Barzilai-Borwein method computes a new step size as follows:

$$\alpha^{(i)} = \frac{\boldsymbol{d}^{(i-1),T} \boldsymbol{d}^{(i-1)}}{\boldsymbol{d}^{(i-1),T} \boldsymbol{y}^{(i)}} \tag{5.2}$$

or

$$\alpha^{(i)} = \frac{\boldsymbol{y}^{(i),T} \boldsymbol{d}^{(i-1)}}{\boldsymbol{y}^{(i),T} \boldsymbol{y}^{(i)}} \tag{5.3}$$

where $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i)}) - \nabla f(\boldsymbol{x}^{(i-1)})$ is a change in the sensitivities of the objective function and $\boldsymbol{d}^{(i-1)} = \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i-1)}$ is the previous update of the design variables. Therefore, if $\boldsymbol{s}^{(i)}$ is a search direction at iteration i, the design update is:

$$\Delta \boldsymbol{x}^{(i)} = \alpha^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{5.4}$$

5.3 Quasi-Newton Barzilai-Borwein method

In contrast to the original method, the Quasi-Newton Barzilai-Borwein (QN-BB) method independently computes each design variable's step size. Therefore, each design parameter has its step size based on the local sensitivity

information. The design update can be found as follows:

$$\boldsymbol{H}^{(i)} = [\alpha_k^{(i)}] \tag{5.5}$$

$$\alpha_k^{(i)} = \min\left(abs\left[\frac{\boldsymbol{y}_k^{(i),T}\boldsymbol{d}_k^{(i-1)}}{\boldsymbol{y}_k^{(i),T}\boldsymbol{y}_k^{(i)}} \right], \alpha_{k,max}^{(i)} \right) \tag{5.6}$$

$$\boldsymbol{y}_k^{(i)} = \boldsymbol{s}_k^{(i)} - \boldsymbol{s}_k^{(i-1)} \tag{5.7}$$

$$\Delta\boldsymbol{x}^{(i)} = \boldsymbol{H}^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{5.8}$$

where $\boldsymbol{s}^{(i)}$ is a search direction computed by the optimization algorithm at iteration i and $\alpha_{k,max}^{(i)}$ is a maximum allowed step size at design variable k.

To extend the QN-BB method to apply to constrained and unconstrained problems, the $\boldsymbol{y}_k^{(i)} = \boldsymbol{s}_k^{(i-1)} - \boldsymbol{s}_k^{(i)}$ is based on the search direction $\boldsymbol{s}_k^{(i)}$ computed by descent gradient method. If $\boldsymbol{s}_k^{(i)} = -\nabla f(\boldsymbol{x}^{(i)})$, eq. 5.8 transforms into the original Barzilai-Borwein method, $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i)}) - \nabla f(\boldsymbol{x}^{(i-1)})$.

5.3.1 Comments to absolute operator

Raydan [48] and Zhou et al. [57] have reported that the original Barzilai-Borwein method might suggest a negative step length in non-quadratic problems. It also applies to structural (or CFD-based) optimization problems, where the functions are typically highly non-linear. Grippo et al. [31] and Raydan [48] suggests an additional condition to accept the step size:

$$f(\boldsymbol{x}^{(i)} + \boldsymbol{d}^{(i)}) \leq \max_{\max(i-M,1)\leq j\leq i} f(\boldsymbol{x}^{(j)}) - \gamma\nabla f(\boldsymbol{x}^{(i)})^T\boldsymbol{d} \tag{5.9}$$

where M is a nonnegative integer and γ is a small positive number. This condition is a weaker form of the Armijo-Goldstein-Wolfe condition and it allows to accept of any point if it improves sufficiently on the largest of the $M+1$ (or i if $i \leq M$) most recent function values. The drawbacks of this solution for simulation-based optimization problems are:

1. Implementation of the Armijo-type line search into the optimization framework is not straightforward because it requires additional communication with physical solvers and data management.

2. The computation time of one optimization step can be dramatically increased if the functional evaluation requires solving a complex numerical model, for instance, non-linear FEM or CFD analysis.

3. The success of the proposed step size is not guaranteed; Hence, the additional computational effort might lead to an inefficient process.

In our work, I apply the BB method to active-set constrained optimization methods, where the activated constraints may strongly change the search direction due to their contributions. In our work, we propose to use an absolute operator to avoid non-positive α. As a result, the design update is ensured to follow the proposed search direction and can't be reversed.

5.4 Analytical examples

Well-known unconstraint optimization problems are solved to demonstrate the performance of the methods.

5.4.1 Raydan Function

If the deviation of the objective function from a quadratic function is insufficient, one can successfully use the original BB method. However, convergence is not guaranteed. Raydan [48] has defined test problem referred to as *Strictly Convex 2* to demonstrate the phenomenon:

$$\begin{aligned} f(x) &= \sum_{i=1}^{d} \frac{i}{10}(\exp(x_i) - x_i) \\ x_0 &= [1, 1, 1..., 1] \end{aligned} \tag{5.10}$$

The Hessian matrix at $\boldsymbol{x}^*$ is $\frac{1}{10}\text{diag}(1, 2, ..., n)$ so that the condition number is n. The Raydan function is a strictly convex function and has a positive definite Hessian for all $\boldsymbol{x}$. It has been verified that the original BB method converges to the solution if $n = 20$ and it diverges if $n = 30$, Fletcher [23] and Raydan [48]. In our results, depending on the choice of the initial step size, the original BB method may converge or diverge. For instance, if $\alpha_1 = 1.0$ it diverges and with $\alpha_1 = 0.97$ it converges. The proposed QN-BB method always converges in $8 - 9$ iterations almost independently of d and initial α_1 because the Hessian matrix is strictly diagonal. The results are summarized in Table 5.1 with $\alpha_1 = 1e - 12$.

Table 5.1: Raydan Function resutls

Problem dimension	# Iterations, BB	# Iterations, QN-BB
$d = 20$	40	8
$d = 30$	54	8
$d = 10^6$	550	9

5.4.2 Generalized Rosenbrock Function

The second test case for the methods is:

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{d-1} c(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \\ c &= 100 \\ \boldsymbol{x}_0 &= [-1.2, 1, -1.2, 1, ..., -1.2, 1] \end{aligned} \tag{5.11}$$

where $f(x)$ is a well-known Rosenbrock function. It is a challenging optimization test due to its narrow, highly curved valley, which makes it challenging to minimize. The starting point is at $\boldsymbol{x}_0$. The convergence criterion is

$\|\nabla f\| \leq 1e^{-6}$. The initial step size has been fixed to $\alpha_1 = 1e-12$. The original BB method can't solve this problem using both formulas eq. 5.2, 5.3. With formula 5.3 at iteration 7, the suggested step size $\alpha < 0.0$. The absolute operator is considered to avoid a negative step length, and the solution is found in 56 iterations. The method is referred to as "BB-abs".

Table 5.2: Rosenbrock function resutls

Problem dimension	# Iterations, BB-abs	# Iterations, QN-BB
$d = 2$	56	18 817
$d = 4$	428	21 437
$d = 10$	15530	nan
$d = 20$	100 000+	51 697
$d = 40$	100 000+	79 128
$d = 100$	100 000+	100 000+

Table 5.2 summarizes the results of both methods with respect to optimization size d. For the problem dimension $d = 2$, the original BB method converges a lot faster than the QN-BB method, and its performance is close to Newton (24 iter.) and Quasi-Newton methods (36 iter.), Martins et al. [41]. Still, the proposed QN-BB method is comparable with the Steepest Descent method (10662 iter.). As the problem dimension increases, the original BB method solves the problem with a highly increased number of iterations compared to QN-BB. For $d = 20$ it requires 100000+ iterations, while the QN-BB method converges in 42985 iterations. Still, both methods are not efficient in solving analytical problems compared to well-established methods, for instance, the Global Barzilai-Borwein algorithm (1429 iter.), Raydan [48].

5.4.3 D-dimension QN-BB method

Motivated by the superior performance of the BB-abs method for the Rosenbrock function with $d < 20$ and taking into account that the original BB method has an R-linear convergence rate in quadratic problems for $d = 2$, Fletcher [23], we have modified the QN-BB method:

$$\boldsymbol{s}_k^{(i)} = [s_k^{(i)}, s_{k+1}^{(i)}, ..., s_{k+m}^{(i)}] \tag{5.12}$$

$$\boldsymbol{d}_k^{(i)} = [d_k^{(i)}, d_{k+1}^{(i)}, ..., d_{k+m}^{(i)}] \tag{5.13}$$

where, to compute the step size for variable k, we use values from k and $k+1$ variables. The method is referred as d-2-QN-BB method ($m = 2$) and d-4-QN-BB method ($m = 4$). The results are summarized in Table 5.3. The d-2-QN-BB and d-4-QN-BB method shows a better performance compared to QN-BB. The d-2-QN-BB method solves the problem faster only for $d = 20$, but the advantage is small. The 3-d-QN-BB method can't be applied for the given dimensions.

Table 5.3: Raydan Function resutls

Problem dimension	# Iterations, d-2-QN-BB	# Iterations, d-4-QN-BB
$d = 20$	11	17
$d = 30$	11	17
$d = 10^6$	11	17

Table 5.4: Rosenbrock function results

Problem dimension	# Iterations, d-2-QN-BB	# Iterations, d-4-QN-BB
$d = 2$	59	nan
$d = 4$	11 650	502
$d = 10$	11 862	nan
$d = 20$	13 641	14 043
$d = 40$	15 685	16 769
$d = 100$	22 700	24 947

5.4.4 D-dimension QN-BB method with Vertex Morphing

In the shape optimization with Vertex Morphing, the d-3-QN-BB method is used, where the step length is computed for each node as a 3D problem, based on its spatial directions [x-, y-, z,]. Hence, the $\boldsymbol{s}, \boldsymbol{d}$ in eq. 5.5 appear to be vectors $[s_x, s_y, s_z], [d_x, d_y, d_z]$ at the node k.

5.5 QN-BB-RGP method

The Quasi-Newton relaxed gradient projection (QN-BB-RGP) method combines the QN-BB and the RGP methods. Linear approximation of the constraint functions is used to improve the constraint handling. In Figure 5.1, the QN-BB-RGP method is shown, and the method's workflow is following:

1. Compute response values at the current design state: $f(x^{(i)})$, $g(x^{(i)})$;
2. Compute gradients of the objective function and active constraints: $\nabla \boldsymbol{f}(x^{(i)})$, $\nabla \boldsymbol{g}(x^{(i)})$;
3. Find shape update $\Delta \boldsymbol{x}^{(i)}$:
   a) Compute search direction $\boldsymbol{s}^{(i)}$;
   b) Compute shape update $\Delta \boldsymbol{x}^{(i)}$;
   c) Compute linear approximation to response function for the computed shape update: $\tilde{g}(\boldsymbol{x}^{(i+1)})$, $\tilde{h}(\boldsymbol{x}^{(i+1)})$;
   d) If $\tilde{g}(\boldsymbol{x}^{(i+1)}) <= 0$ and $\tilde{h}(\boldsymbol{x}^{(i+1)}) = 0$, then the feasible shape update is found. The inner loop is converged;
   e) If $\tilde{g}(\boldsymbol{x}^{(i+1)}) > 0$ and $\tilde{h}(\boldsymbol{x}^{(i+1)})! = 0$, then the feasible shape update is not found. Update the buffer coefficients $\omega_j^{(i)} += c$ and repeat the inner loop process;

4. Save current $\Delta \boldsymbol{x}^{(i)}$, $\boldsymbol{s}^{(i)}$;

5. Check if the optimization algorithm has converged. If not, go to Step 1.

The constant $c = [0.01, 0.2]$ to increase the $\omega_j^{(i)}$ is based on the numerical experiments, and it shows a good compromise between accuracy and cost. It has no effect on $\omega_j^{(i+1)}$.

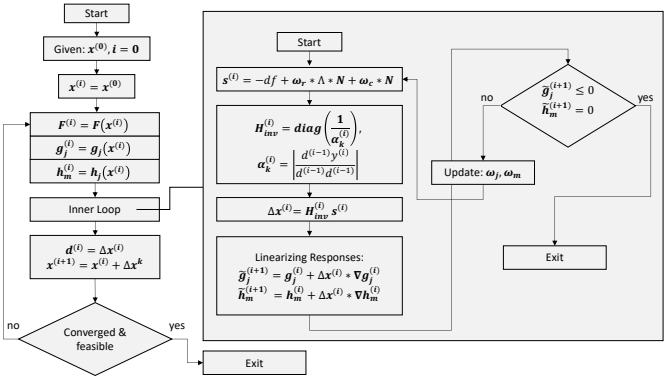


Figure 5.1: Flow chart of the Quasi-Newton relaxed gradient projection method

5.6 Academic Shape Optimization example

An academic structural optimization problem has been solved using different optimization methods to show the efficiency, performance, and drawbacks of the proposed methods compared to "classical" methods.

5.6.1 Case description

The model is represented by a solid hook model with height 237 mm and is modeled with a linear elastic material with Young's modulus $E = 209.6$ GPa and a Poisson's ratio $\nu = 0.29$. The objective of the optimization problem is to minimize the mass while the model has to stay feasible concerning constraints. The initial compliance for two static load cases, where one load is applied at the center (LC1: $F = 32$ kN) and the second one on the tip (LC2: $F = 16$ kN). The hook is supported at its top. Another geometrical constraint (GC1) is represented by the packaging response with the curved packaging geometry at the back side of the hook. The nodes and areas where the loads and supports are applied are excluded from the design set, and it leads to 21351

design variables (7117 number of design surface nodes). Vertex Morphing with constant radius $r = 25$ mm is applied as a parametrization method. The optimization process includes the structural analysis (*StructuralMechanics Application*), the adjoint sensitivity analysis and optimizer (*ShapeOptimization Application*), and the pseudo-elastic mesh motion for the internal nodes of solid elements (*MeshMoving Application*). The case study and all related implementations are done in the open-source software Kratos-Multiphysics Dadvand et al. [14] and Ferrándiz et al. [18]. The optimization is stopped after 30 iterations. The shape optimization benchmark is prepared by Mr. Armin Geiser for the library of examples, Kratos-Multiphysics.

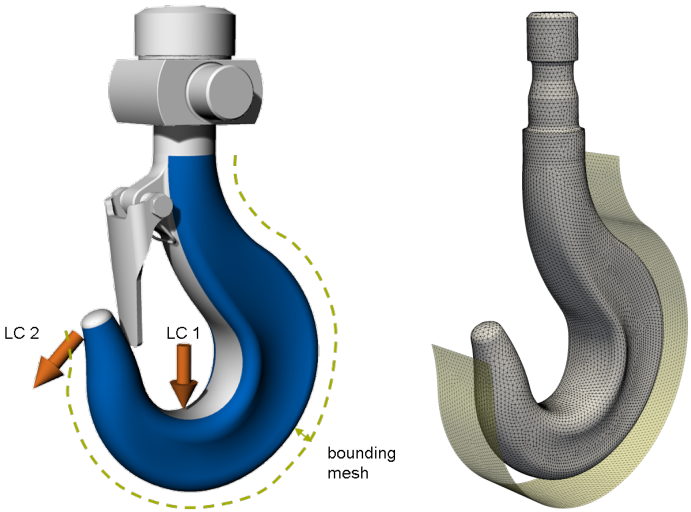


Figure 5.2: Solid Hook, structural optimization benchmark, Geiser et al. [25]

The objective function "mass" and strain energy constraint functions are smooth functions with stable design gradients during the optimization process. In contrast, the packaging constraint with the square-sum aggregation has a non-linear penalty behavior with highly changeable sensitivities because the nodes consequently inter and leave an infeasible domain. Due to specific geometry and discretization, this structural problem can be classified as a small node-based problem.

5.6.2 Tested optimization algorithms

The following optimization algorithms and their modifications have been implemented and tested:

1. Gradient Projection method, Rosen [49]. In our implementation, the correction term is based on the constraint violation at the iteration point $\boldsymbol{x}^{(i)}$. The scaled constant step size is used $\alpha = 3.0$ mm. It is referred to as "GP".

2. Relaxed gradient projection method, Antonau et al. [3]. The scaled constant step size is used $\alpha = 3$ mm. It is referred to as "RGP".

3. Relaxed gradient projection method with the inner loop to check the linear approximation of the constraint values. The scaled constant step size is used $\alpha = 3$ mm. It is referred to as "RGPnl".

4. Relaxed gradient projection method with the inner loop to check the linear approximation of the constraint values and Barzilai-Borwein, eq.5.2. It is referred to as "RGPnl-BB1".

5. Relaxed gradient projection method with the inner loop to check the linear approximation of the constraint values and Barzilai-Borwein, eq.5.3. It is referred to as "RGPnl-BB2".

6. Relaxed gradient projection method with the inner loop to check the linear approximation of the constraint values and Quasi-Newton Barzilai-Borwein, eq.5.5. It is referred to as "QN-BB-RGP".

5.6.3 Results

The optimization results of the Hook benchmark are summarized in Table 5.5. It compares the improvements of the objective function, constraint violations, and computational time obtained by the optimization methods. Fig. 5.3 shows the objective values, constraints values, and shape update size during the optimization process.

Table 5.5: Hook benchmark results

Method	h:mm:ss	Δf	Δ LC1	Δ LC2	GC1
GP	0:08:09	-19.44%	0.2%	0.1%	645
RGP	0:08:11	-17.45%	0.04%	-0.8%	347
RGPnl	0:08:35	-16.1%	0.09%	-0.8%	5.12
RGPnl-BB1	0:09:49	-17.66%	-0.01%	-0.62%	0.995
RGPnl-BB2	0:10:24	-13.85%	-0.14%	-0.23%	0.31
QN-BB-RGP	0:11:10	-13.31%	-0.01%	-0.19%	0.005

Computational time. In all numerical optimization methods, the number of structural analyses is the same, one per optimization iteration. Also, all other processes, such as saving data, are identical in all runs. The difference in the computational time is due to the chosen optimization algorithm. The fastest methods are GP and RGP because they don't have the inner loop to check. RGPnl requires only 24 sec more to finish 30 optimization iterations and find a feasible solution. QN-BB-RGP takes more time than the RGPnl-BB methods because it requires more inner iterations to compute shape

updates. However, the computation of the step size for each design variable is a relatively inexpensive operation.

Objective function improvement. The GP method finds the best improvement, but the solution is infeasible. As the GP method keeps decreasing the mass, the constraint violations also increase. Hence, the GP method with step size 3 mm diverges, and it can't converge to a feasible local minimum. RGPnl-BB1 finds the best feasible solution, which differs from QN-BB-RGP and RGPnl-BB2. My hypothesis is that the RGPnl-BB1 converges to a different local minimum because it chooses larger step sizes and converges to the local minimum from the infeasible side. In contrast to GP, all RGP-based runs show a non-smooth mass reduction in the first iterations until the buffer zones are constructed. Then, as the buffer zone is adjusted, the amount of the violations are reduced.

Constraint function violations. All methods found the final solutions that are feasible with respect to strain energy constraints or have minor violations up to 0.1%. The geometrical constraint is more challenging to satisfy due to its penalty formulation. As a result, GP and RGP methods strongly violate the GC1 constraint. Our results show, that the usage of linear approximation dramatically improves the accuracy of the GC1 constraint. Adaptive step size, BB or QN-BB, further reduces the violations of all constraints, stabilizes the behavior, and precisely finds the local minimum.

5.7 Large Shape Optimization Example

Large shape optimization problems are solved to demonstrate the performance of the proposed methods on the example of industrial importance. The commercial solvers, Altair Optistruct (structural) and Siemens StarCCM (CFD), have been used to do primal and adjoint simulations. The optimization framework ShapeModule (BMW Group) is used as an optimizer where the proposed methods are implemented.

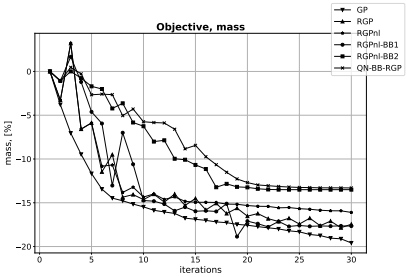
5.7.1 Structural optimization

5.7.1.1 Problem description

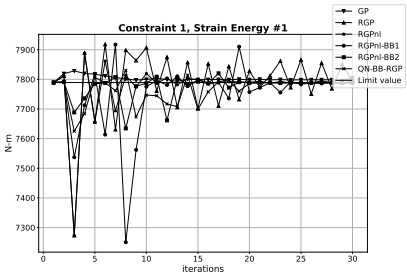
The structural optimization problem is described in Publication I, Section 4.3. The problem has 144423 design variables (48141 surface nodes with x-, y-, z- spatial directions), one objective function (mass), one physical constraint (maximum displacement), and one aggregated packaging constraint (see Section 6.2.1).

5.7.1.2 Applied methods

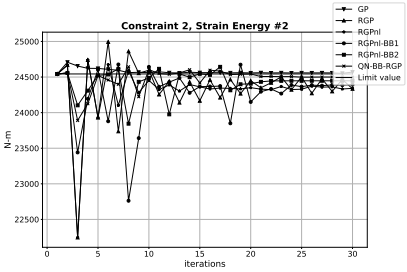
The optimization problem has been solved using three methods: RGP with constant step size $\alpha = 0.5$ mm, RGP with relaxed BB method, eq. 5.3 (including inner loop with linearized constraint values), and QN-BB-RGP method. The relaxed BB method avoids large jumps in the step size (from



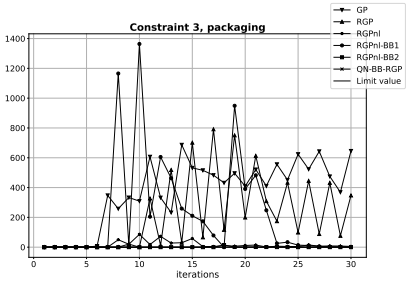
(a) mass



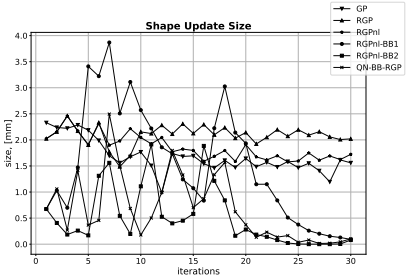
(b) Strain energy, load case 1



(c) Strain energy, load case 2



(d) Packaging response



(e) Shape update size

Figure 5.3: Hook benchmark, comparison of the optimization methods

lower boundary to upper), and it computes relaxed step size α_r based on the proposed and previous step length:

$$\alpha_r^{(i)} = 0.5\alpha^{(i-1)} + 0.5\alpha^{(i)} \tag{5.14}$$

5.7.1.3 Results

The optimization results are shown in Fig. 5.4. The QN-BB-RGP method has found the best-performing feasible result with the least iterations. Similarly, the BB-RGP method has increased the step size α up to 1.5 mm till the maximum displacement constraint is non-active, iteration 7. Afterward, the suggested step size is dramatically reduced. As a result, the RGP method with constant step size has a better improvement rate until the maximum displacement is violated. At iteration 35, when the maximum displacement is satisfied, the BB method suggests increased step size, which leads to a violation of the constraint. Due to mesh quality, the simulation has been stopped at iteration 37.

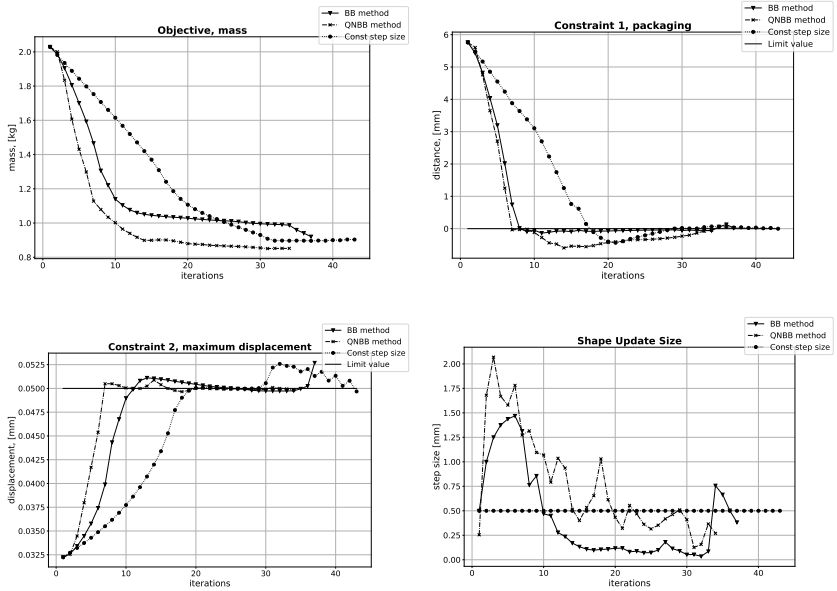


Figure 5.4: Optimization results

Fig. 5.5 compares the final shapes. From top to bottom, the methods are RGP with constant step size, RGP with BB method, and QN-BB-RGP. The solutions found by the RGP method with constant step size and the BB method are similar to each other. In contrast, the QN-BB-RGP method finds a different local minimum with a smaller mass. As a result, the proposed solution has bigger holes and smaller cross-sections in the trusses.

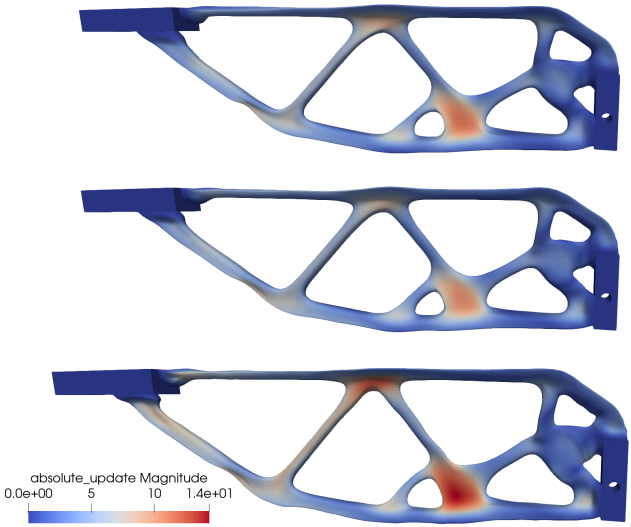


Figure 5.5: Optimization results comparison: RGP (top), BB method with RGP (middle), QN-BB-RGP (bottom)

5.7.2 CFD-Based Shape Optimization

5.7.2.1 Problem description

Publication II shows a large industrial CFD-based shape optimization problem of the full car model. The goal of the optimization is to improve the efficiency of the racing car BMW M4 GT4 aerodynamic package. The drag force is used as an objective function. At the same time, the downforce is applied as an inequality constraint referring to the initial value. The CFD model is well described in Publication II, Chapter 6. The design surface is the rear wing and front splitter, Fig. 5.6.

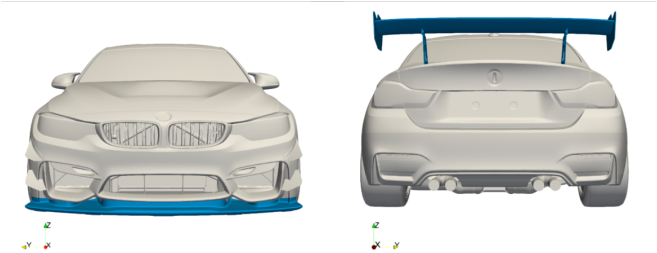


Figure 5.6: Design surface, marked with blue color

5.7.2.2 Applied methods

The problem is solved using three strategies: RGP method with constant step size $\alpha = 1$ mm, RGP method with relaxed BB method "BB-RGP" (see eq. 5.14) and QN-BB-RGP method.

5.7.2.3 Results

The optimization results are shown in Fig. 5.7. The modified design surfaces are compared from optimization iteration 10 in Fig 5.8. RGP reduces the drag force by 1.91% and finds a feasible solution with respect to downforce. BB-RGP suggests small step sizes because the response functions are highly non-linear and the gradients change dramatically from iteration to iteration. During the optimization process, BB-RGP has the slowest improvement rate, and the constraint violations are comparable to the RGP run. QN-BB-RGP method has found the best drag force reduction, $4,03\%$, and it founds a feasible solution. During the optimization process, the constraint violation is more than 3%, which is corrected at the last optimization iterations. The shape update size computed by QN-BB-RGP is in the range $[0.42 - 3.35]$ mm.

The BB-RGP run is done on a different HPC configuration compared to other runs. As a result, there is a small deviation in the fluid solution. The relative difference in the initial values is 0.16% for SCx and 0.19% for SCz.

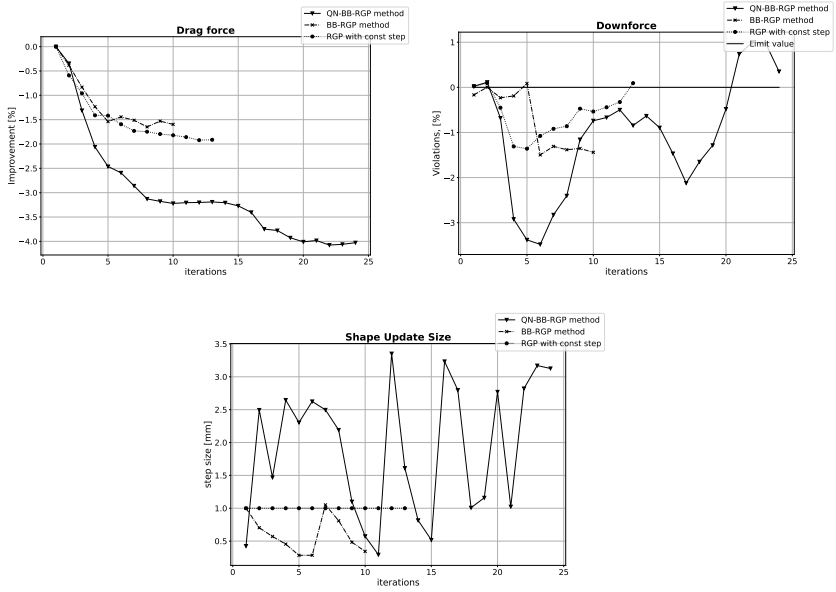
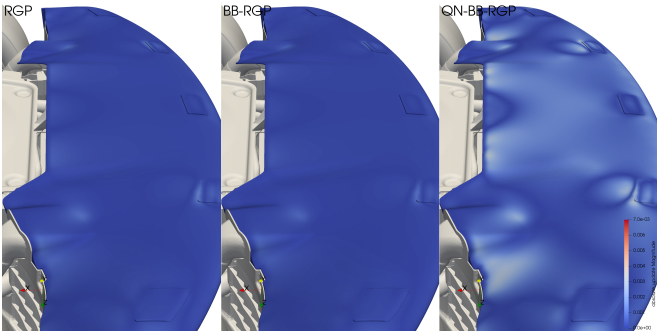
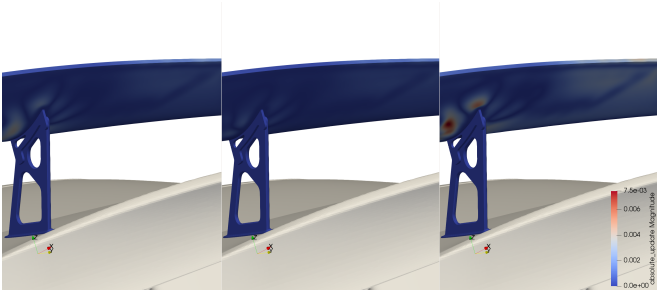


Figure 5.7: Optimization results



(a) Front spliter



(b) Rear wing

Figure 5.8: Absolute update, optimization iteration 10. Left – RGP, middle – BB-RGP, right – QN-BB-RGP

Chapter 6

Shape optimization in additive manufacturing application

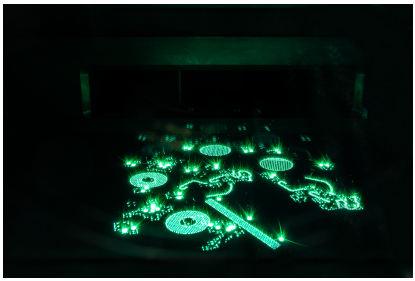
6.1 Overview

In modern engineering, numerical free-form optimization methods allow to find new high-performance parts bringing the products to a new level. Typically, these solutions have a complex geometry that is hard or expensive to manufacture with the "classical" manufacturing methods, such as stamping, casting, hydroforming, etc. Alternatively, one may apply manufacturing constraints that greatly limit the design space. In contrast to "classical" methods, additive manufacturing (AM), reduces the geometric restrictions to a large extent and allows the manufacturing of almost any topology. Additionally, AM efficiency and fabrication cost are not very sensitive to geometric complexity. In AM, the product is manufactured layer-by-layer by a wide range of techniques: Fused Deposition Modeling (FDM)or Stereolithography (SLA) for plastic printing, direct metal laser sintering (DMLS) or Laser Powder Bed Fusion (LPBF) for metal printing, Fig. 6.1. As a result, AM can easily create parts based on the freeform design of topology and shape optimization, Ghantasala et al. [28] and Liu et al. [40].

Besides the before mentioned advantages, there are AM-specific limitations to printed geometry. The most discussed one is that of *self-supporting* structures (overhang-free). If the geometry is not self-supported and additional supporting structures are not applied, the printed part may break during the printing process or have poor surface quality. Therefore, the support structures are used, which leads to additional material consumption and process time to design the required support structures and to remove the support structures and attachments from actual parts. According to Liu et al. [40], the cost of the not self-supported geometry is increased by 40-70 % according to their internal data.

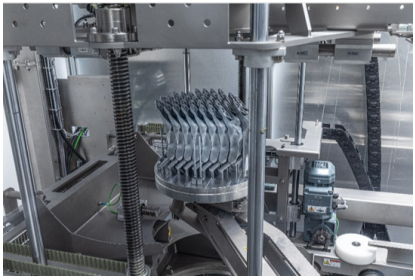
Another AM-specific property of the geometry is *stackability*: the property of how close the geometries can be packed one into another. The term

(a) 3D printing machine



(b) 3D printing process

(c) Internal working space



(d) 3D printed parts

Figure 6.1: Additive manufacturing process, Laser Powder Bed Fusion (LPBF). Source: [Successful industrialization and digitalization of additive manufacturing, https://www.press.bmwgroup.com]

stackabilization is firstly introduced by Li et al. [37], and it names the process of improving the stackability of the geometry. The impact of the stackability on the efficiency of additive manufacturing is evident: due to limited space inside the printing area, Fig. 6.1c, it is possible to produce more parts during one process. As a result, the average manufacturing time and cost per part can be reduced. The first formulation of the stackabilization process for node-based shape optimization with Vertex-Morphing is formulated by A. Ghantasala and R. A. Najian, Ghantasala et al. [27].

6.2 Stackabilization

Stackibilization is a process to improve the stackability of the geometry in a specific direction, Li et al. [37]. There are two key parameters to improve stackability: the shape of the model and a stackable direction. In this work, the main focus is to modify the shape of the initial geometry to improve stackability in a given stackable direction d.

Stackibilization starts with creating two copies of the geometries, left and right, which are offset by the given distance a in the stackable direction d. The next process is adjusting the packaging geometry's position. Initially, it serves to find an initial position of the packaging geometries, where they touch the main geometry but don't penetrate with a given tolerance. The process to find new position of the packaging geometries is summarized in Fig. 6.2 and it contains following steps:

1. Compute g_i values of the nodal packaging response function, eq. 6.2.
2. Move the packaging geometry in the stackable direction d by small step α_s. Repeat it k times till the self-penetration is detected.
3. Compute gain in stackability Δg as a traveled distance of the packaging geometry till self-penetration is detected:

$$\Delta g = -\alpha_s k \tag{6.1}$$

4. Compute packaging gradients ∇g, eq. 6.5.

The overall stackabilization process is summarized in Fig. 6.3. The following section 6.2.1 describes the packaging response, that is used in Step 4.

6.2.1 Packaging response

Following Geiser et al. [25] and Najian Asl et al. [45] the packaging response defines geometrical constraints to prevent a final design surface from penetrating arbitrary packaging geometry by dividing the design space into a feasible and infeasible domain. Fig. 6.4 shows a shape optimization problem with discrete packaging geometry (gray box). The surface normals of the packaging geometry point into the feasible domain. The nodal value of the packaging response at the node i is a projection of the distance vector to the closest packaging node onto the surface unit vector of that node:

$$g_i = (\boldsymbol{cp}_i - \boldsymbol{x}_i)^T \boldsymbol{n}_{cpi} \tag{6.2}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{cp}_i$ are spatial coordinates of the node i and its closest node on the packaging surface. g_i is positive if node i is inside an infeasible node, zero if it lies on the boundary, and negative if it is inside the feasible domain. The nodal gradient of node i is the negative unit normal of the packaging's closest point at the position i and zero elsewhere:

$$\nabla g_i = [0, ... - \boldsymbol{n}_{cpi}, ..., 0] \tag{6.3}$$

The node-wise formulation of packaging response creates n geometrical constraints for each node i, Allaire et al. [2]. One can use aggregation techniques to reduce the number of constraints to aggregate nodal constraints into a global one. It satisfies the nodal constraints globally and in a less numerically expensive way. Allaire et al. [2], Brelje et al. [10], and Damigos

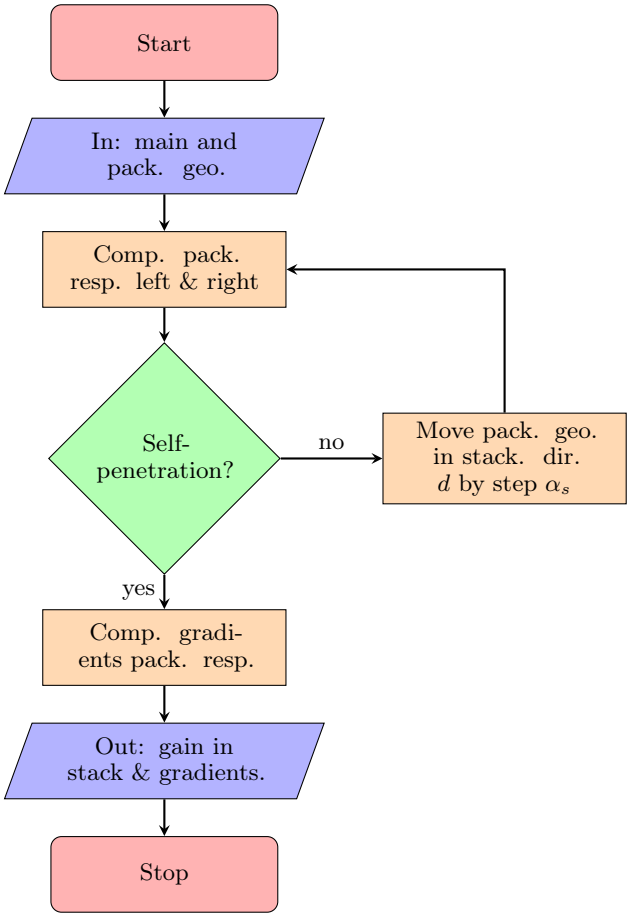


Figure 6.2: Process to adjust the position of packaging geometries, compute gain in stackability and packaging gradients

et al. [16] have previously applied aggregation formulations for geometric constraints in shape optimization. One possibility to aggregate the nodal constraints is to take the square sum of all nodal constraints that are infeasible ($g_i^+ > 0$):

$$g = \sum_{i=1}^{n} (g_i^+)^2 \tag{6.4}$$

where the gradient of the aggregated constraint is:

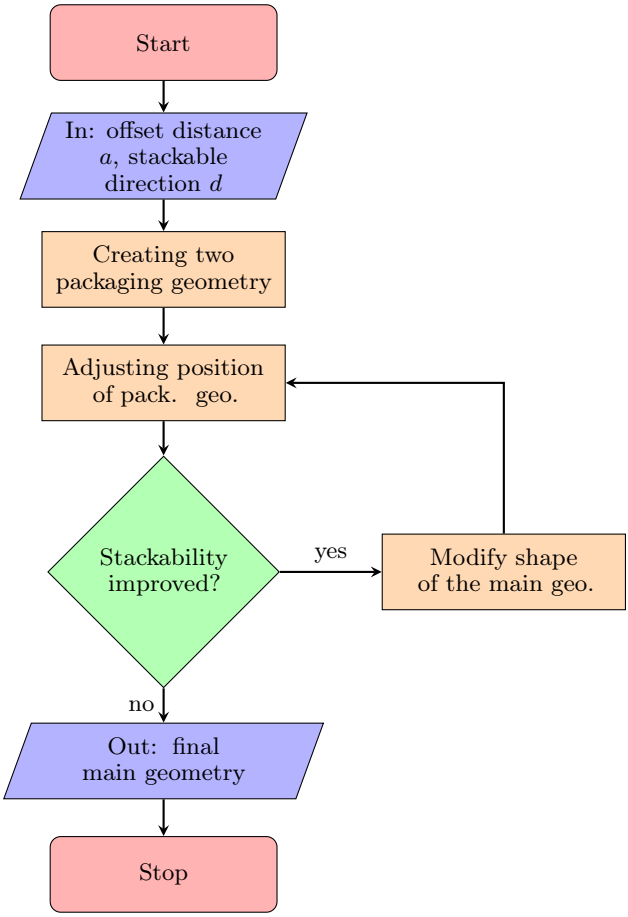


Figure 6.3: Stackabilization process (pack. geo. – packaging geometry)

$$\nabla g = 2g_i^+ \boldsymbol{n}_{cpi} \tag{6.5}$$

6.3 Self-support (overhang-free) constraint

6.3.1 Identification of self-supporting nodes

There are two conditions to check if the node is self-supported:

1. Based on the critical angle condition;

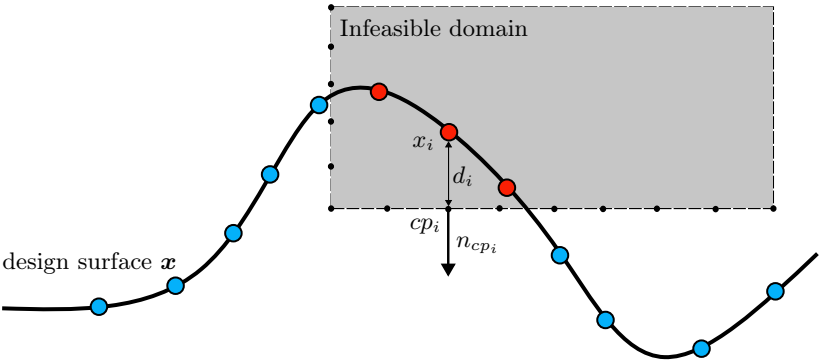


Figure 6.4: Nodal packaging constraint. Feasible nodes are highlighted in blue, infeasible – red

2. Based on the distance to supporting nodes.

The identification workflow of overhang-free nodes is summarized in Fig. 6.5.

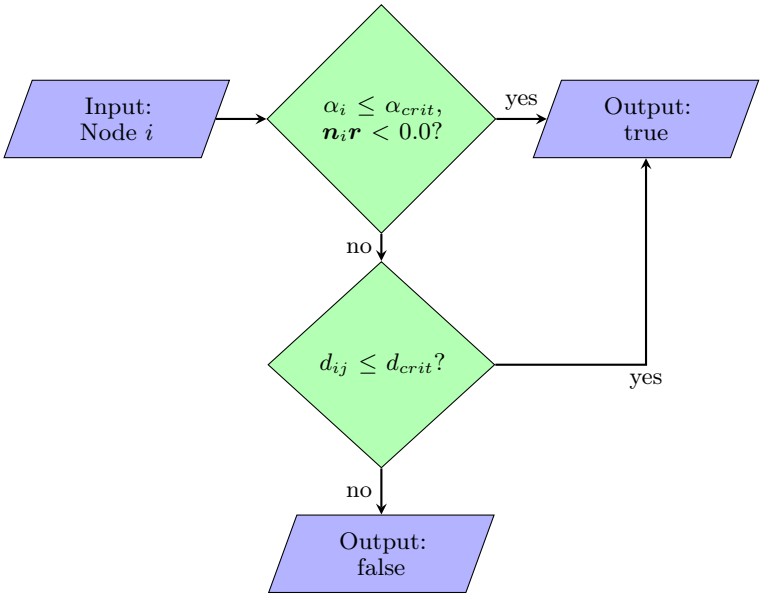


Figure 6.5: Process to determine if a node is self-supported

6.3.1.1 Critical angle criterion

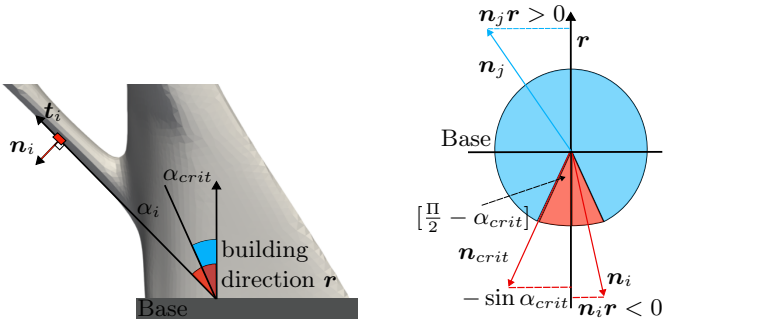
Following the ideas of Garaigordobil et al. [24], we compute the angle α_i for the node i, where the angle is between building direction r and tangential vector $\boldsymbol{t}_i$. In contrast to topology optimization, the boundary is always defined in shape optimization. Therefore, it is straightforward to compute the angle α_i and compare it to the critical angle α_{crit}. The criterion for node i to be feasible is:

$$\begin{cases} \alpha_i \leq \alpha_{crit} \\ \boldsymbol{n}_i \boldsymbol{r} < 0.0 \end{cases} \tag{6.6}$$

The criterion can be reformulated as in terms of the projection of the normal vector $\boldsymbol{n}_i$ onto the building direction r:

$$-\boldsymbol{n}_i \boldsymbol{r} - \sin \alpha_{crit} \leq 0.0 \tag{6.7}$$

where $-\sin \alpha_{crit}$ is a size of the critical projection. Fig. 6.6a shows the graphical representation of the eq. 6.6 and Fig. 6.6b – the eq. 6.7. In Fig. 6.7, the feasible and infeasible nodes are shown for critical angles $\alpha_{crit} = [10°, 30°, 45°]$. All the nodes where the projection of the normal vector onto the building direction is lower than -0.5 are marked red (as infeasible nodes). The number of infeasible nodes is reduced if the critical angle is increased due to larger feasible normal vector orientations, Fig. 6.7. The build direction is $r = [0.56, 0.043, 0.83]$ in all comparison examples.



(a) Angle and critical angle description

(b) Normal vector projection

Figure 6.6: Self-support constraint, critical angle criteria. Infeasible node is highlighted in red

6.3.1.2 Distance criterion

The second condition is applied to the infeasible nodes to check if they are positioned close to the support structure. The condition can be formulated as

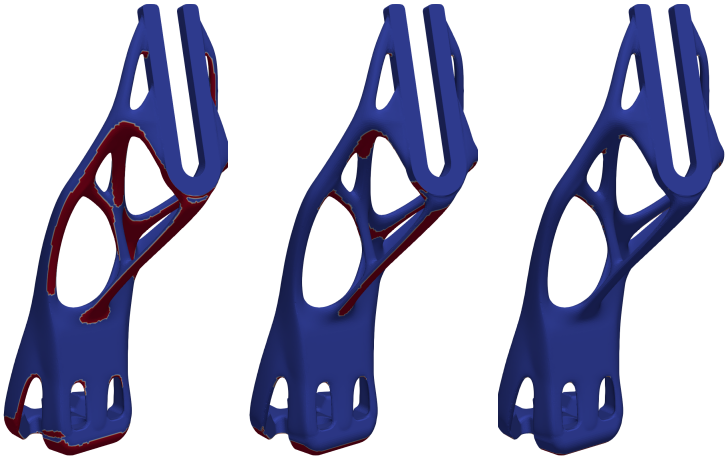


Figure 6.7: Infeasible elements (red color) with respect to critical angle: $\alpha_{crit} = 10°, 30°, 45°$ (from left to right)

a comparison of the distance between a node and the supporting node with critical distance:

$$\begin{aligned} d_{ij} &\leq d_{crit} \\ d_{ij} &= \|x_i - x_j\| \end{aligned} \tag{6.8}$$

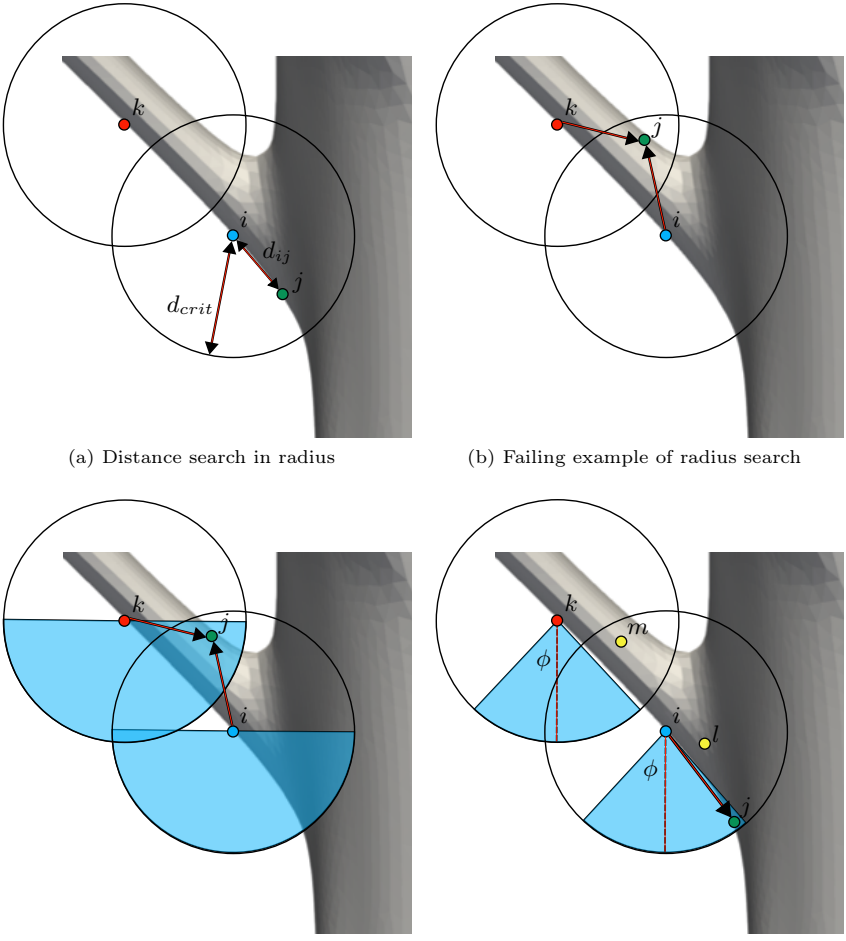
where d_{ij} is an Euclidean distance between nodes i and j, x_i and x_j are spatial coordinates of the nodes i and j respectively. The process to find the supporting nodes is following:

1. Find neighbor nodes j around node i within radius $r = d_{crit}$;
2. Check if there is a node j that is feasible with respect to angle condition (*necessary* condition), eq. 6.6. See Fig. 6.8a;
3. Check if the found node j can support node i, *sufficient* condition.

Step 1 fulfills the distance condition, eq. 6.8. Step 2 checks if there is a candidate that can support the node i. In Fig. 6.8a the node i finds a candidate j, while the node k has no candidates. Step 2 is a necessary condition but not sufficient. In Fig. 6.8b, there is a critical case where node i finds a candidate j which is above node i in the building direction. Hence, node j can't support node i because it isn't built at this point. Therefore, step 3 is introduced to verify the candidate j. There are two possible methods to check:

1. Half-sphere method checks that the support-candidate is below the supported node. In Fig. 6.8c the candidate j is valid for node k, but is invalid for node i;

2. Cone method checks that the support-candidate is inside the cone with height d_{crit} and angle ϕ. This condition avoids support candidates from "sides". In Fig. 6.8d, the node j can support the node i, while the nodes l and m don't support nodes i or k, in contrast to the half-sphere condition.



(a) Distance search in radius

(b) Failing example of radius search

(c) Search technique in a lower half-sphere

(d) Search in a cone

Figure 6.8: Support-node search strategies

6.3.2 Response function formulation

The constraint is formulated as a penalty function where the possible value is positive or zero. The value is defined as a sum of the square element normal projection of the infeasible elements. An infeasible element is an element where at least one node is considered infeasible. If the element is feasible, the contribution to the sum is zero:

$$\begin{aligned} g &= \sum_i (g_i^+)^2 \\ g_i^+ &= \begin{cases} \boldsymbol{n}_i \boldsymbol{r}_i; \text{ if elem}_i \text{ is infeasible} \\ 0; \text{ if elem}_i \text{ is feasible} \end{cases} \end{aligned} \tag{6.9}$$

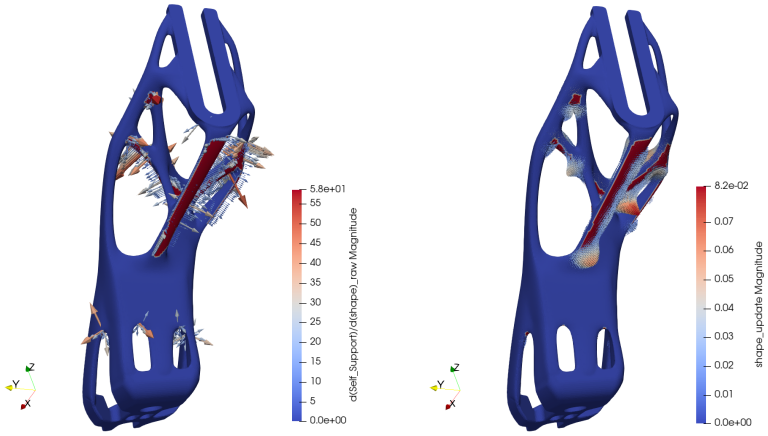
The derivative of the response function is computed using finite difference, where the node's spatial coordinates of the infeasible element are moved by a small change, and the difference of the elemental value g_i is computed:

$$\begin{aligned} \nabla g_i &= 2 g_i(x) \frac{g_i(x+\delta) - g_i(x)}{\delta} \\ \nabla g &= \sum_i \nabla g_i \end{aligned} \tag{6.10}$$

Fig. 6.9 shows the sensitivities computed using finite difference (left) and a smooth shape update computed using the steepest descent algorithm and Vertex Morphing. As a result, the shape update tries to change its angle with respect to the building direction. Fig. 6.10 shows the computed shape update (left) and the corrected geometry (right) to find an overhang-free shape for the corner case. One can see that the fixed geometry has a more narrow base (green boxes), and the upper position of the corner (blue box) is overhang-free concerning critical angle and distance criteria.

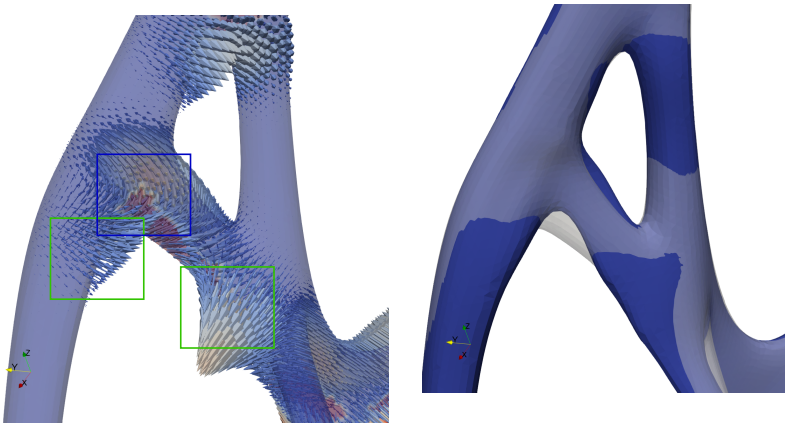
6.4 Numerical examples

This section presents several numerical experiments to test the proposed AM-specific responses. All simulations are done using the geometry for the fixture of a soft-top attachment in the BMW i8 Roadster, Fig . 6.11a. The proposed methods are implemented in the optimization framework ShapeModule (BMW Group). Altair Optistruct ™ software is used to solve the numerical models' structural primal and adjoint analysis. All optimization problems are solved using adaptive Vertex Morphing parametrization and the Quasi-Newton relaxed gradient projection method, which have been discussed in previous chapters. The size of the optimization problems is 145326 (number of surface nodes $\times 3$). Fig. 6.11b shows the FE model, where the nodes with red-colored are the design nodes that the optimizer can update. The blue nodes are the non-design nodes. The Vertex Morphing with adaptive filtering radius, Chapter 3 parametrization technique is applied without additional filtering size settings. The QN-BB-RGP (Chapter 5) is used as an optimizer for



(a) Raw sensitivities (b) Smooth shape update

Figure 6.9: Self-support constraint



(a) Gain in stackability (b) Constraints handling

Figure 6.10: Corner modification for overhang-free geometry: left – shape update based on response gradients, right – overhang-free corner (blue) and an initial geometry (transparent gray)

constrained and unconstrained problems. The convergence criteria are:

$$\sum_{j=i-5}^{j=i} \frac{f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^{(0)})}{f(\boldsymbol{x}^{0})} \leq 1e^{-5} \tag{6.11}$$

$$i \leq 200 \tag{6.12}$$

(a) Photo of real part (b) Numerical model

Figure 6.11: Geometry of fixture for soft-top attachment. a) real part, photo via BMW Group; b) Numerical model: red – design parts, blue – non-design parts

6.4.1 Stackabilization

The aforementioned model is optimized to improve the stackability of the geometry. The stackable direction $\boldsymbol{d}$ is $[0.56, 0.043, 0.83]$ and it remains unchanged during the whole optimization process. The update steps α_s to move the packaging geometry is set to 0.05 mm. Fig. 6.12 shows the stackability improvement during optimization. After iteration 38, the optimizer can't improve the shape of the model to improve further because the bottom non-design parts (See. Fig. 6.11b) get in contact with each other. The final shape and the positions of the packaging geometries are shown in Fig. 6.13. The shape changes are applied locally at the regions where initial geometries have self-penetrations. Therefore, the final geometry is similar to the original one. Still, the applied local changes allow improving the stackabilization by 40% see Fig. 6.13.

6.4.2 Overhang-free geometry

In this section, the geometry is optimized to be overhang free for the building direction $\boldsymbol{r} = [-0.18, 0.0, 0.46]$, the critical distance is $d_{crit} = 2$ mm, and the searching angle is $\phi = 45°$. In graph 6.14, the response value minimization is shown. The function is highly non-linear and has penalty behavior. As a result, the reduction of the function can't be ensured at every iteration. Nevertheless, the optimizer can reduce the overhang-free response value by

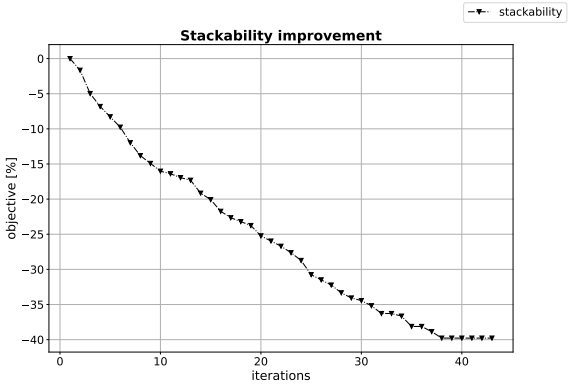


Figure 6.12: Stackabilization improvement through optimization iterations

100%, which means that all infeasible nodes are corrected. In Fig. 6.15, the final shape is compared to the initial one.

6.4.3 Combined example

In this section, the combined optimization problem is solved. The stackabilization process is applied as an objective function, while the overhang response, mass, and two compliance functions are added as constraints. The load is applied at the top fixed zone in z- and y- directions, while the bottom part of the geometry is fixed. The stackable direction is $\boldsymbol{d} = [0.56, 0.043, 0.83]$, the building direction is $\boldsymbol{r} = [-1, 0, 1]$, the searching angle is $\phi = 45^\circ$ and the critical distance is $d_{crit} = 2$ mm.

Fig. 6.16 shows the improvement in the stackability function while the constraints are satisfied. The starting design is infeasible with respect to the self-support response function, which is corrected during the optimization process. In the first steps, the optimizer reduces the self-support violation a lot as the correction part for it is maximal. The constraint is first corrected at the optimization iteration 18, but it violates in the next iterations. It happens because the middle bar gets bent to improve stackability, and at the same time, it becomes non-self-supported. The final result is an overhang-free design. In Fig. 6.17, the final shape and packaging geometry positions are shown. The corners/arches are lifted up similar to Fig. 6.10 to have supports that are overhang-free, and they can support the corner/arches nodes as well. Additional modifications close to the fixed zones reinforce the structure and satisfy the required compliance.

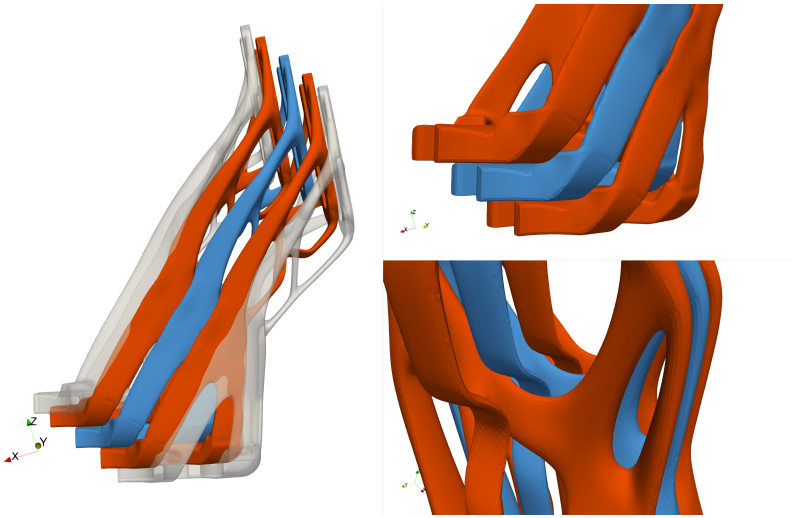


Figure 6.13: Final shape of the geometry. Blue — main geometry, orange — packaging geometries, gray — initial position and shape of packaging geometries

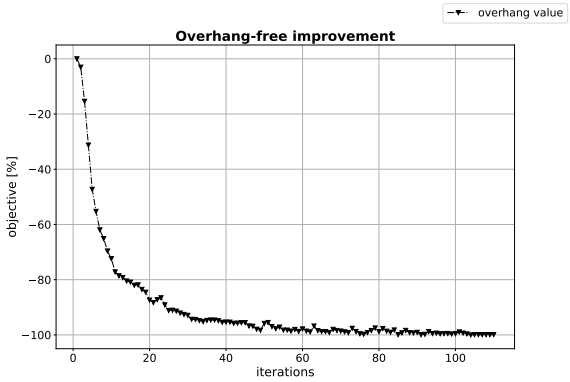


Figure 6.14: Reduction of the overhang response value during optimization

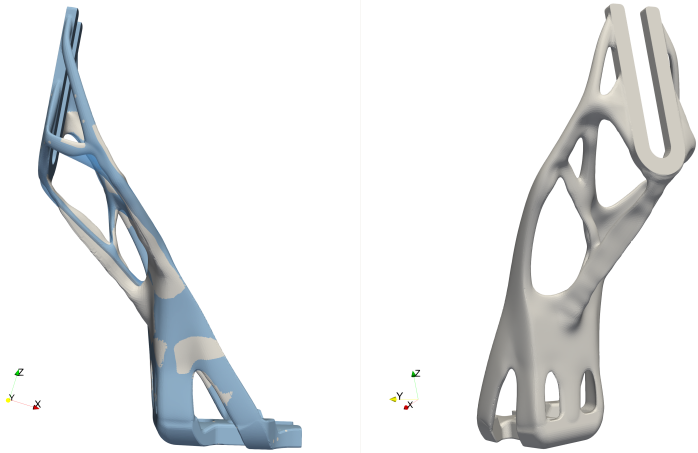
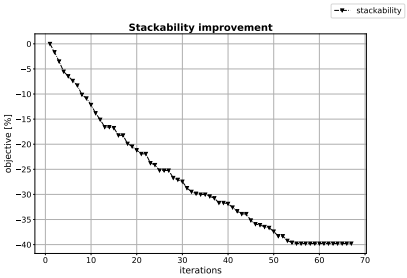
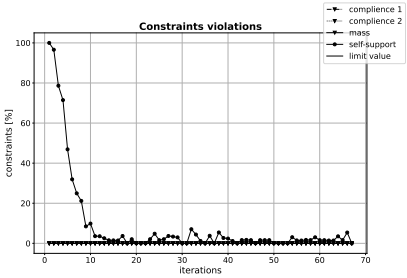


Figure 6.15: Overhang-free geometry. Transparent blue is the initial shape, and gray – final shape



(a) Gain in stackability



(b) Constraints handling

Figure 6.16: Optimization results

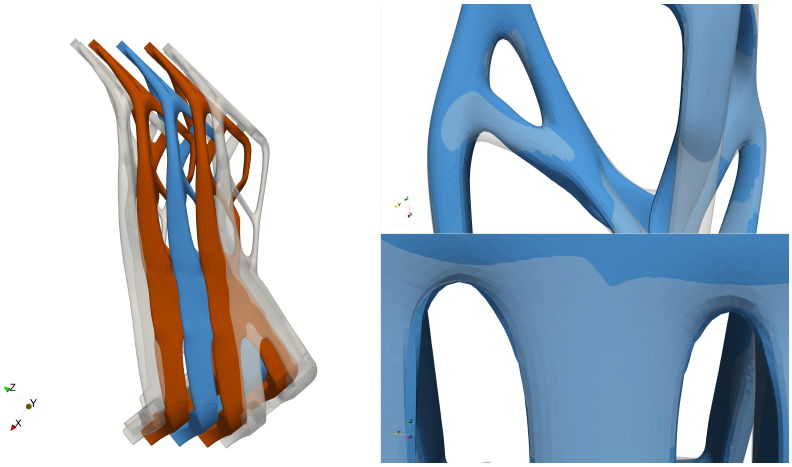


Figure 6.17: Stackable and overhang-free geometry. Right: Blue — main geometry, orange — packaging geometries; left: transparent gray is initial shape, blue – final shape

Chapter 7

Conclusions and outlook

7.1 Vertex Morphing with adaptive filtering radius

Vertex Morphing with adaptive filtering radius extends the original Vertex Morphing to improve its usability, flexibility, and robustness.

Usability. AVM computes the radius field based on the mesh size to ensure the filtering properties and improve the surface quality. With the radius field, AVM doesn't need any user input regarding radius size to do optimization. As a result, AVM is easy to be applied on a unknown complex model. It also simplifies the automatization of the optimization process in the design cycle. Chapter 6 shows the examples for additive manufacturing where AVM is applied without user input.

Robustness. AVM updates the radius field every optimization iteration to insure the filtering properties. Academic structural example from Chapter 3 demonstrates, that this property is not required in case of small shape changes and its necessity if the large shape changes are required and allowed.

Flexibility. AVM allows to set of different filtering radii for every node; consequently, it allows finding different local minima. As a result, a designer can locally adjust the parametrization to explore the design space and find various solutions, that are trusted, esthetically pleasant and satisfying manufacturing criteria. Publication II shows the full-car optimization problem, where the different sizes of the radius is applied and it allows to find good shape modes for different parts of the car.

The smoothing algorithm of the AVM method is required for the radius field to avoid jumps in the shape updates. Publication II studies the effect of the smoothing algorithm on the final surface quality. Publication II provides the practical values for the AVM parameters to achieve a good quality of the results.

Outlook. In future research, one can extend the base to compute the filtering radius size to use surface curvature or spatial derivatives of the shape sensitivities, etc. Publication II shows the CFD-based examples, where AVM doesn't filter well the wrong sensitivities at the flow-separation points. The

additional conditions on the mapped sensitivities may be found to avoid this scenario. Additionally, the method can be extended to support non-matching grids by mapping the radius fields from the control space onto the shape surface to insure the conjugation of the shape updates across the meshes.

7.2 Optimization Algorithm

7.2.1 Relaxed gradient projection method

In Publication I, the relaxed gradient projection method is introduced. It reduces the zig-zagging behavior of the constraints along the design boundaries by keeping them active in the critical zone. As a result, the relaxed gradient projection method can handle various physical, geometric, and manufacturing constraints. Publication I demonstrates the structural optimization problems, where RGP speeds up the objective function reduction by factor 1.7 compare to the gradient projection method.

Chapter 5 shows the Hook benchmark, which is poorly solved by the RGP method. The reason for high violations is a small projection compare to correction part. The modified RGP method with inner loop, that checks the linear approximation of the constraints improves the behavior and it is used as a base for the Quasi-Newton method.

Outlook. The relaxed gradient projection algorithm has been used as a default optimization algorithm in the ShapeModule optimization framework (BMW Group) since 2019. It has been successfully applied to numerous constrained shape optimization problems in different industrial applications by engineers at BMW Group. In future research, one can better formulate the inner loop and solve it in a more efficient way by following ideas of SLP methods, Vanderplaats [55].

7.2.2 Barzilai-Borwein method

In this work, the possibility to apply the Barzilain-Borwein method for shape optimization problems is studied. Our results show that the method is easy to implement and performs well in engineering optimization problems. The found issue of the method is, that it may compute a negative step length, that is not acceptable. Therefore, we propose to apply an absolute operator to the step size.

Furthermore, the Quasi-Newton Barzilai-Borwein method is introduced. It provides the adaptive step size for each design variable based on the Barzilai-Borwein formula. The Hook benchmark, Chapter 5 shows improvement in the constraint handling and accuracy. It also offers the possibility of having a local maximum step size for each design variable, which is important for AVM parameterization. The CFD-based full car optimization problem shows the “scaling” properties of the QN-BB method, where the front region with smaller sensitivities has been strongly modified due to larger step sizes at these nodes. As a result, the found solution is better compare to solutions, which are found with the Barzilain-Borwein method and the constant step size.

Outlook. In future research, one can study the QN-BB method's weak efficiency in small optimization problems. Our results show that it requires more optimization iterations to converge compare to original method in small analytical problems.

7.3 Shape Optimization for Additive Manufacturing

Two AM-specific constraints are discussed in this work. The stackabilization process is overviewed and tested on the real-world numerical examples. The process is based on the idea of sequential movement of the packaging geometries in the stackable direction by a small constant amount. The process is formulated as an objective function and it can be combined with various constraints or objective functions.

The self-supported constraint is firstly formulated for node-based shape optimization problems. The constraint identifies the infeasible nodes by two conditions: well-known critical angle and new distance criteria. The distance criteria use the cone-search strategy, where the critical angle and distance are given as input. one can define the input parameters based on the real printed parts examples, where unsupported parts are analyzed and classified as acceptable or not. The input parameters have to be found for specific manufacturing machines and materials. The self-support constraint can be used as objective or constraint function and with any constraint or objective functions.

Real world example. Fig. 7.1 shows the manufactured parts with improved stackability by BMW Group. The parts have been designed using the implementation in ShapeModule (BMW Group).

The self-support constraint still requires the verification of the model. The real world experiments have to be done to compare the simulation results and real world examples.

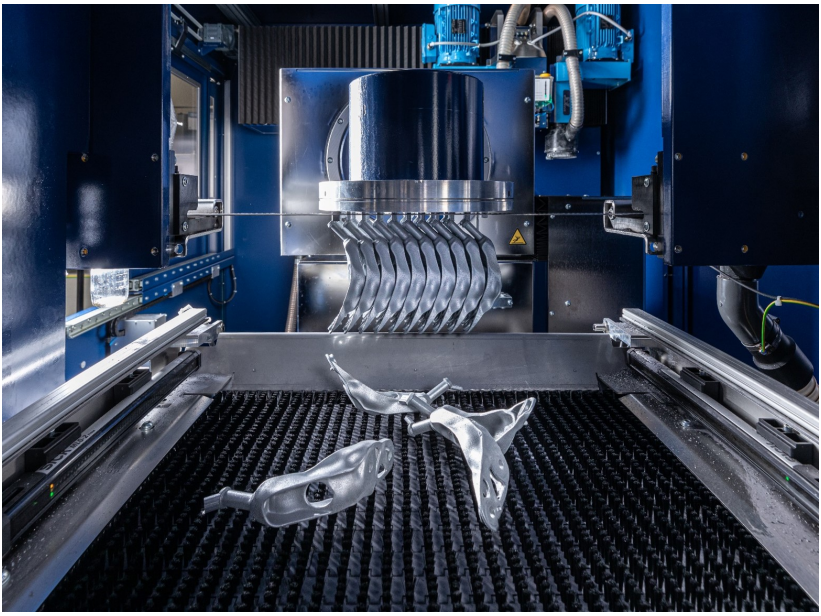


Figure 7.1: 3D printed parts with improved stackability. Photo via BMW Group

Appendix A

Publication I

Structural and Multidisciplinary Optimization (2021) 63:1633–1651
https://doi.org/10.1007/s00158-020-02821-y

RESEARCH PAPER

Relaxed gradient projection algorithm for constrained node-based shape optimization

Ihar Antonau[1] · Majid Hojjat[2] · Kai-Uwe Bletzinger[1]

Received: 4 August 2020 / Revised: 13 November 2020 / Accepted: 9 December 2020 / Published online: 10 February 2021

Abstract
In node-based shape optimization, there are a vast amount of design parameters, and the objectives, as well as the physical constraints, are non-linear in state and design. Robust optimization algorithms are required. The methods of feasible directions are widely used in practical optimization problems and know to be quite robust. A subclass of these methods is the gradient projection method. It is an active-set method, it can be used with equality and non-equality constraints, and it has gained significant popularity for its intuitive implementation. One significant issue around efficiency is that the algorithm may suffer from zigzagging behavior while it follows non-linear design boundaries. In this work, we propose a modification to Rosen's gradient projection algorithm. It includes the efficient techniques to damp the zigzagging behavior of the original algorithm while following the non-linear design boundaries, thus improving the performance of the method.

Keywords Gradient-based constrained optimization · Shape optimization · Vertex Morphing · Rosen's gradient projection algorithm · Node-based shape parametrization

1 Introduction

The aim of this paper is to propose a modified algorithm for constrained node-based shape optimization. It has good potential to improve the objective function by finding a new design through the modification of the shape of the initial model. In our paper, we are interested in iterative optimization methods, where a continuous evolution of the design produced. Shape optimization is successfully used in many fields of application: aerospace engineering (Kroll et al. 2007; Kenway et al. 2014; Palacios et al. 2012), automotive industry (Najian Asl et al. 2017; Hojjat et al. 2014), structural mechanics (Chen et al. 2019; Haftka and Grandhi 1986; Firl and Bletzinger 2012), fluid-structure interaction (FSI) (Hojjat et al. 2010; Heners et al. 2017), etc.

General, constrained shape-optimization problems can be formulated as follows:

$$\begin{aligned} &\textbf{minimize}: f(\boldsymbol{x}) \\ &\textbf{design variables}: \boldsymbol{x} \\ \text{s.t.:} &g_j(\boldsymbol{x}) \leq 0, \text{ where } j = 1..n_g \\ &h_k(\boldsymbol{x}) = 0, \text{ where } k = 1..n_h \end{aligned} \tag{1}$$

where $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{x}$ is the vector of design parameters, $g_j(\boldsymbol{x})$ are inequality constraints, and $h_k(\boldsymbol{x})$ are equality constraints. An important step in optimization is the choice of the design variables. In the optimization of the shape (and topology), there are two main types of shape parametrization: explicit and implicit. Implicit parametrization can be presented, for instance, by the free-form deformation (FFD) approach (Sieger et al. 2012) or a level-set method (Wang and Luo 2020; Luo et al. 2008). Alternatively, in the explicit parametrization, such as Vertex Morphing (Bletzinger 2017; Hojjat et al. 2014) or CAD-based parametrization (Xu et al. 2014; Agarwal et al. 2018; Hardee et al. 1999), the representation of the geometry is directly used as a design parameter field. In this work, we are using Vertex Morphing parametrization.

Responsible Editor: Ming Zhou

✉ Ihar Antonau
ihar.antonau@tum.de

[1] Structural Analysis, Technical University Munich, Munich, Germany

[2] BMW Group, Munich, Germany

The main advantage of the Vertex Morphing is no additional optimization model is needed. The analysis model is used directly, where the coordinates of the surface nodes are the design parameters. Isogeometric parametrization (Ummidivarapu and Voruganti 2017; Ummidivarapu et al. 2020) is a good alternative to the Vertex Morphing. Both methods have similarities, and the difference is typically in the number of design variables. Vertex Morphing uses surface nodes of the FE model as a design parameters; therefore, there is a large number of variables. That allows us to find new unknown solutions by changing the parametrization settings. On the other hand, with Vertex Morphing, it is challenging to apply boundaries and geometrical constraints to the design parameters (Najian Asl et al. 2017). The interested reader can find more details about Vertex Morphing and form-finding in Bletzinger (2017), Baumgärtner et al. (2016), Hojjat et al. (2014).

Solving industrial problems is state of the art. The main focuses groups researching shape optimization problems are developing industrial applications, deriving sensitivity analysis w.r.t shape design variables, and finding new designs of the models. In most cases, they use well-established optimization algorithms, such as steepest descent, gradient projection, augmented Lagrangian, or trust-region algorithms. Nonetheless, classical algorithms may suffer from poor efficiency due to the specific properties of the problems. For instance, the active-set methods may suffer from the zigzagging phenomenon (Fletcher 2013; Sun and Yuan 2006) because constraints repeatedly enter and leave the active set. Therefore, it results in slow convergence of the method (Gallagher and Zienkiewicz 1977). Typical properties of the node-based shape optimization problem are:

- A large number of design variables. In the Vertex Morphing practice, the "usual" number is around $10e5 - 10e6$. That makes solving the optimization problem not straightforward;
- The objectives, as well as the physical constraints, are non-linear in state and design;
- The sensitivity analysis for different objective or constraint functions cannot always be solved analytically; thus, they are solved with a tolerance;
- The sensitivities of the different responses have to be scaled due to the different physical units. Scaling may mean that information regarding the size of the raw sensitivities is lost;
- Calculation of the $f(\boldsymbol{x}), \nabla f(\boldsymbol{x}), g(\boldsymbol{x}), \nabla g(\boldsymbol{x})$ is computationally expensive. Doing physical analysis may take up $\approx$ 50–80% of the one optimization iterations computational time;
- Algorithms such as gradient projection that require extra calculations of the response functions to calculate the correction step precisely (we discuss this in the details in the Sections 2.1 and 2.2). This can be numerically expensive or may require additional assumptions and simplifications.
- Line search techniques can be numerically expensive or non-accurate for highly non-linear functions. In practice, a constant step size may be preferred.

In this work, we propose a relaxed gradient projection method. The method is a modification of the classical Rosen's gradient projection algorithm (Rosen 1960, 1961). In this context, "relaxed" means that constraints can be in the transient stage between active and non-active. The relaxation and correction factors mildly control the relaxation and violation of the constraints. In the proposed method, we introduce the buffer (critical) zone to calculate the relaxation factor and the correction term violated constraints. As a result, the algorithm has efficient techniques to damp zigzagging behavior when it follows the design boundaries and has stable performance.

The paper is structured as follows: First, the Rosen's gradient projection algorithm is reviewed as the reference method. Then the proposed algorithm and its simplified version are described. The next section describes the numerical experiments and shows a detailed analysis of the performance of the proposed and reference methods. Finally, conclusions are drawn from the work.

2 Rosen's gradient projection algorithm

This section describes the Rosen's gradient projection algorithm (GP), its advantages, and disadvantages in the context of shape optimization problems. The method is used as the reference algorithm in our studies.

2.1 Gradient projection method

The gradient projection algorithm calculates feasible search direction by projecting the steepest descent direction into the tangent subspace to the active constraints. Detailed description can be found in the article by Rosen (1960) and Du et al. (1990). We will describe the way to calculate the projected search direction for optimization problems with linear constraints by following Haftka and Kamat (1990). The problem can be formulated as follows:

$$\begin{aligned} \textbf{minimize} : f(\boldsymbol{x}) &= \sum_i \omega_i f_i(\boldsymbol{x}), \text{ where } i = 1..n_f \\ \text{s.t.} : g_j(\boldsymbol{x}) &= \boldsymbol{a}_j \boldsymbol{x} - b_j <= 0, \; j = 1, ..., n_g \\ h_k(\boldsymbol{x}) &= \boldsymbol{a}_k \boldsymbol{x} - b_k = 0, k = 1, ..., n_h \end{aligned} \tag{2}$$

If we select only the r active constraints, we can define an n by r matrix N, such that the columns of this matrix are the

gradients of active constraints. The basic assumption of the gradient projection method is that $\boldsymbol{x}$ lies on the tangential subspace to the boundary of the active constraints. If our solutions $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(i+1)}$ at the iteration i and $i+1$ satisfy the constraints, then the constraints can be rewritten as:

$$N^T \boldsymbol{s} = 0 \tag{3}$$

where $\boldsymbol{s}$ is a search direction. If we want to project the steepest descent direction $-\nabla f$ on the tangent subspace of the active set of constraints, we can redefine problem (2) as follows:

$$\begin{aligned} &\textbf{minimize} : \boldsymbol{s}^T \nabla f \\ &\quad \text{s.t.} : N^T \boldsymbol{s} = 0, \\ &\qquad \text{and } \boldsymbol{s}^T \boldsymbol{s} = 1 \end{aligned} \tag{4}$$

where the second condition bounds the solution. The Lagrangian function is:

$$L(\boldsymbol{s}, \boldsymbol{\lambda}, \mu) = \boldsymbol{s}^T \nabla f - \boldsymbol{s}^T \boldsymbol{N} \lambda - 2\mu(\boldsymbol{s}^T \boldsymbol{s} - 1) \tag{5}$$

The condition for L to be stationary is

$$\frac{\partial L}{\partial \boldsymbol{s}} = \nabla f - \boldsymbol{N}\lambda - 2\mu \boldsymbol{s} = 0 \tag{6}$$

We can find the Lagrangian multiplier λ by multiplying (6) with N^T and using condition from (3):

$$\lambda = (\boldsymbol{N}^T \boldsymbol{N})^{-1} \boldsymbol{N}^T \nabla f \tag{7}$$

and the feasible search direction s:

$$\boldsymbol{s} = \frac{1}{2\mu} [\boldsymbol{I} - \boldsymbol{N}(\boldsymbol{N}^T \boldsymbol{N})^{-1} \boldsymbol{N}^T] \nabla f \tag{8}$$

In Najian Asl et al. (2017) and Haftka and Kamat (1990), the authors observe that the factor $\frac{1}{2\mu}$ does not play an important role in the determination of search direction because it scales the vector and does not change its direction. The final search direction to minimize the objective function can be changed with sign factor "-".

To find the Lagrangian multiplier λ in (7), the linear system of equation of size $r \times r$ needs to be solved. Depending on the number of active constraints r and design variables n, the constraint matrix N can be sparse and large. The condition number of such a system can be large; therefore, special attention should be paid to the choice of an efficient and robust linear solver. The reader may refer to Najian Asl et al. (2017) for more details on solving (7). After finding the feasible search direction, new shape $\boldsymbol{x}_{i+1}$ can be found. A line-search can be used to find the step size $\alpha^{(i)}$ that sufficiently reduces the objective function or a constant step size can be used. Design update can be found as follows:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \alpha^{(i)} \boldsymbol{s} \tag{9}$$

2.2 Reduced gradient projection algorithm

Rosen's work (1961) provides an extension to the gradient projection algorithm to handle non-linear constraints. The main idea is to calculate the correction (restoring) move that can bring violated constraints back into the feasible domain. To calculate the restoring move, we linearize the constraint:

$$g_j \approx g_j(\boldsymbol{x}^{(i)}) + \nabla g_j^T (\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}) \tag{10}$$

Using the linearized equation of the constraints, we can find the correction move:

$$\begin{aligned} \bar{\boldsymbol{x}}^{(i+1)} - \tilde{\boldsymbol{x}}^{(i+1)} &= -\boldsymbol{N}(\boldsymbol{N}^T \boldsymbol{N})^{-1} \boldsymbol{g_a} \\ g_{a,j} &= g_j(\tilde{\boldsymbol{x}}^{(i+1)}) \end{aligned} \tag{11}$$

where $\tilde{\boldsymbol{x}}^{(i+1)}$ is a new design after minimizing in the tangential direction, (8), $\bar{\boldsymbol{x}}^{(i+1)}$ is the corrected design, and $\boldsymbol{g_a}$ is a vector which contains the violations of the active constraints. Equation (11) is based on the linear approximation and therefore should be repeated several times, until $\boldsymbol{g_a}$ is sufficiently small. In addition, the matrix N should be re-evaluated for each point, which means all physical solvers should be deployed again in order to undertaken a sensitivity analysis for active constraints. In the industrial case, the deployment of physical solvers can be very time consuming. Hence, it can be computationally expensive to calculate a correction move several times, or even just once. To reduce computation cost, one can use the violation of the constraint g_j at the beginning of the iteration, $g_{a,j} = g_j(\boldsymbol{x}^{(i)})$. With this assumption, the correction move might not bring the $\boldsymbol{x}_{i+1}$ back on the design boundary. Therefore, in one or more optimization iterations, the active constraint would become non-active ($g_j(\boldsymbol{x}^{(i+1)}) < 0$). In this case, the algorithm would perform a steepest descent step, like other feasible direction methods (Vanderplaats 2007), and violate the constraint again in the next iteration. That leads to zigzagging behavior of the algorithm, and the reduction of its performance (Fletcher 2013; Sun and Yuan 2006).

2.3 Numerical example

Figure 1 gives the typical diagram with the zigzagging behavior of the gradient projection algorithm with a constant step size. After some initial iterations, the non-linear design boundary is reached and overshot. On the next iteration, the algorithm calculates the projected direction and applies the correction move to bring the solution back

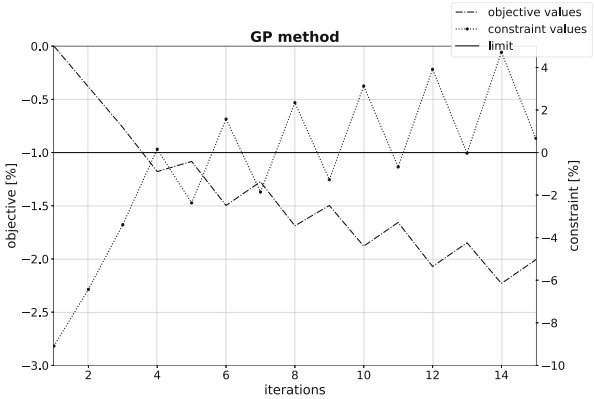


Fig. 1 Zigzagging behavior of GP method

to the feasible side. This leads to the zigzagging of the objective and constraint values.

3 Relaxed gradient projection algorithm

To overcome issues with gradient projection methods in our optimization problems, we introduce the proposed method, a relaxed gradient projection algorithm (RGP). It incorporates techniques for damping the zigzagging behavior of the algorithm, while following non-linear active constraints. This section describes the RGP method and its simplified version (SRGP).

3.1 Buffer (critical) zone

The gradient projection algorithm has an issue with switching on and off the constraint while following the design boundary. To avoid switching, we introduce the buffer (critical) zone. The buffer (critical) zone is the region where the constraint is considered as active. Inside this zone, we calculate the buffer coefficient $\omega_j^{(i)}$, which defines how "strongly" the constraint should be considered. If the constraint value has not reached the limit value, the $\omega_j^{(i)}$ coefficient makes the constraint "weaker." On the other hand, if the constraint value is on the limit or has violated the limit, the $\omega_j^{(i)}$ coefficient makes the constraint fully active. It smoothly varies from zero to two, where "zero" means that constraint is non-active, and "one" means that the constraint value has reached its limit value. If the buffer coefficient is more than one, the constraint is violated, and the correction part should be applied. We use linear distribution through the buffer zone for the buffer coefficient. Non-linear distribution is non-applicable because the algorithm varies the buffer coefficient in a non-linear way, and it reduces the stability of the method. Based on the size of buffer and its central position, one can calculate the buffer coefficient $\omega_j^{(i)}$ for inequality constraints ($g_j(\boldsymbol{x}_i) \leq 0$):

$$
\begin{aligned}
LBV_j^{(i)} &= CBV_j^{(i)} - BS_j^{(i)} \\
\omega_j^{(i)} &= \frac{g_j(\boldsymbol{x}^{(i)}) - LBV_j^{(i)}}{BS_j^{(i)}}
\end{aligned}
\tag{12}
$$

or for equality constraints ($h_j(\boldsymbol{x}_i) = 0$):

$$
\omega_j^{(i)} = 1 + \frac{abs[g_j(\boldsymbol{x}^{(i)}) - LV_j]}{BS_j^{(i)}} \tag{13}
$$

where $LBV_j^{(i)}$ is a value "lower buffer value," $BS_j^{(i)}$ is a value "buffer size," $CBV_j^{(i)}$ is a value "central buffer value," $g_j(\boldsymbol{x}^{(i)})$ is a constraint value, and LV_j is a limit value. All values are calculated for the jth constraint at the ith iteration.

With (12) and (13), CBV_j and BS_j should first be defined. Initially, $CBV_j^{(i)}$ can be set to be the same as the corresponding constraint limit value. Finding suitable $BS_j^{(i)}$ requires the use of historical information. In the first step, BS_j can be initialized as some small value, for instance $1e-12$ or 1% of the constraint limit value. Starting from the second iteration, we can calculate the maximum change in the constraint value $\Delta g_j^{(i)}$ during the optimization process. By the multiplying maximum change by the buffer size factor BSF, we can estimate the $BS_j^{(i)}$:

$$
\begin{aligned}
BS_j^{(i)} &= BSF \cdot \max_k(\Delta g_j(\boldsymbol{x}^{(k)})) \\
\Delta g_j^{(i)} &= abs(g_j(\boldsymbol{x}^{(i)}) - g_j(\boldsymbol{x}^{(i-1)}))
\end{aligned}
\tag{14}
$$

In general, the buffer factor should be more than one ($BSF > 1.0$) and can be changed during the optimization

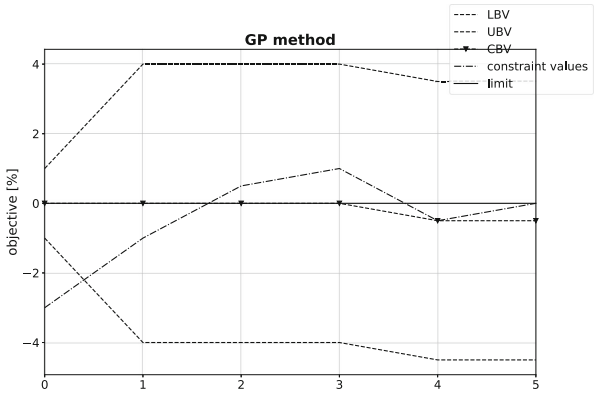


Fig. 2 Buffer zone around constraint limit

process via the buffer adaptation functions. Initially, in our examples, the buffer factor *BSF* is set to 2 because then the algorithm has at least one optimization iteration inside the buffer zone before the constraint value reaches its limit. To sum up, the buffer zone controls the active set of constraints through the constraint's value. In Fig. 2, there is a graph to demonstrate the typical buffer zone around the constraint value.

3.2 Search direction

The RGP algorithm inherits from Rosen's gradient projection algorithm the projection part of the feasible direction, and can rotate it to the direction of the active constraints gradients. The buffer coefficient $\omega_j^{(i)}$ is divided into two components. The first part is relaxation, whereby the constraint is "relaxed" when it is in the feasible domain. The $\omega_j^{r,(i)}$ relaxation coefficient is calculated as follows:

$$\omega_j^{r,(i)} = \begin{cases} \omega_j^{(i)}, & \text{if } \omega_j^{(i)} \leq 1.0 \\ 1, & \text{if } \omega_j^{(i)} > 1.0 \end{cases} \tag{15}$$

If the constraint is equality, the relaxation coefficient is always equal to one, $\omega_j^{r,(i)} = 1.0$. The second component, the $\omega_j^{c,(i)}$ correction coefficient is:

$$\omega_j^{c,(i)} = \begin{cases} BSF_{init}(\omega_j^{(i)} - 1), & \text{if } 1.0 < \omega_j^{(i)} < \omega^{max} \\ 0, & \text{if } \omega_j^{(i)} \leq 1.0 \\ BSF_{init}\omega^{max}, & \text{if } \omega_j^{(i)} \geq \omega^{max} \end{cases} \tag{16}$$

where the factor BSF_{init} is the initial buffer size factor, and ω^{max} is the maximum correction coefficient. If the problem starts from an infeasible domain, the correction coefficient can be very high and may cause numerical issues. The $\omega^{max} = 2$ limits the correction coefficient to the values inside the buffer zone and can work in the most cases. Nonetheless, if the problem starts well inside the infeasible domain, the ω^{max} can be increased to a relatively large value, for instance 10 or 100. If we combine the relaxation and correction components, we can define the search direction:

$$\begin{aligned} \boldsymbol{p}^{(i)} &= -[\boldsymbol{I} - \boldsymbol{N}\boldsymbol{\omega}^{r,(i)}(\boldsymbol{N}^T\boldsymbol{N})^{-1}\boldsymbol{N}^T]\nabla f^{(i)} \\ \hat{\boldsymbol{s}}^{(i)} &= \boldsymbol{p}^{(i)} - \boldsymbol{N}\boldsymbol{\omega}^{c,(i)} \\ \boldsymbol{s}^{(i)} &= \frac{\hat{\boldsymbol{s}}^{(i)}}{||\hat{\boldsymbol{s}}^{(i)}||_{max}} \end{aligned} \tag{17}$$

where $\boldsymbol{\omega}^{r,(i)}$ is an r by r diagonal matrix; $\omega_j^{r,(i)}$ is placed in the main diagonal; $\boldsymbol{\omega}^{c,(i)}$ is a vector with size r, which consist of $\omega_j^{c,(i)}$ buffer coefficients; and $\boldsymbol{p}^{(i)}$ is the relaxed projected direction. All vectors, $\nabla f(\boldsymbol{x}^{(i)})$ and $\nabla g_j(\boldsymbol{x}^{(i)})$ are scaled using max norm ($\nabla f \leftarrow \frac{\nabla f}{||\nabla f||_{max}}, \nabla g_j \leftarrow \frac{\nabla g_j}{||\nabla g_j||_{max}}$). The first equation in (17) calculates the relaxed projected direction $\boldsymbol{p}^{(i)}$, which is similar to the (8). The $\boldsymbol{\omega}^{r,(i)}$ relaxation coefficient can be understood as a factor to control how strongly the steepest direction should be turned into the projected direction. The second equation in (17), the correction equation, is different to the correction move from (11). In contrast to the correction move (11), the correction part rotates the projected direction to point towards the feasible domain, instead of calculating the design update. If the violation of the constraint is higher, the rotation is more significant. If there are several violated constraints and the $\omega_j^{c,(i)}$ correction coefficients are large, the final search direction can have high norm. To avoid it, the third equation normalizes (bounds) the search direction.

Figure 3 shows the possible search directions, calculated using the RGP method. As it is shown, if the constraint value is not inside the buffer zone ($\omega_j^{(i)} = 0$), the search direction

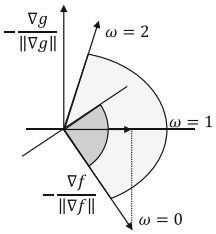


Fig. 3 Possible search direction inside buffer zone, calculated using RGP method

is the same as the steepest descent. If the $\omega_j^{(i)} = 1$, the search direction is the projection of the steepest direction onto the tangential subspace of the active constraint (8). If the $\omega_j^{(i)} > 1$, the search direction is rotated towards the feasible domain. The dark gray sector shows the search directions, which can improve the objective functions.

3.3 Buffer adaptation functions

The performance of the proposed algorithm strongly depends on the buffer size, because the buffer size is used to calculate the feasible search direction. There is a chance that the initial size and central position of the buffer zone may become non-optimal during the optimization process. Therefore, the algorithm requires functions in order to correct the buffer size in two cases: the zigzagging behavior around the constraint limits for the case where constraint violation increases.

In the first case, if the zigzagging for the constraint is detected, the buffer size factor is increased by the following rule:

$$BSF_j^{(i+1)} = BSF_j^{(i)} + abs(\omega_j^{(i)} - \omega_j^{(i-1)}) \cdot \text{factor} \qquad (18)$$

where factor is a positive number that scales the update of the buffer size factor. In all numerical examples, factor is set to one. With an increase in the buffer size factor $BSF_j^{(i+1)}$, the buffer size $BS_j^{(i+1)}$ is increased respectively. If we calculate the constraint value $g_j(\boldsymbol{x}^{(i+1)})$ and its respective correction coefficient $\omega_j^{c,(i)}$ using previous and new buffer sizes $BS_j^{(i)}$, $BS_j^{(i+1)}$, it can be seen that the correction coefficient $\omega_j^{c,(i)}$ is smaller with the new buffer size factor than with the previous one. In contrast, the relaxation coefficient $\omega_j^{r,(i)}$ is greater with the new buffer size. Therefore, the value of the constraint will be changed less from iteration to iteration. Alternatively, the buffer size factor can be modified by various rules. For instance, buffer size factor can be doubled, if zigzagging is detected.

The zigzagging behavior can be detected in different ways. One of them is comparing n number of constraint values with the constraint limit value. If the constraint values are sequentially greater and lower than the limit value, the zigzagging around limit is detected. Alternatively, the sign of the result of the multiplication of the difference in constraint values $\Delta g_j^{(i)}$ at n previous iterations can be checked. If we consider 4 previous iterations, the zigzagging criteria is:

$$\begin{cases} \Delta g_j^{(i)} \cdot \Delta g_j^{(i-1)} < 0 \\ \Delta g_j^{(i-1)} \cdot \Delta g_j^{(i-2)} < 0 \\ \Delta g_j^{(i)} = g_j(\boldsymbol{x}^{(i)}) - g_j(\boldsymbol{x}^{(i-1)}) \end{cases} \qquad (19)$$

Besides the zigzagging behavior, the constraint value can move away from a limit value in the infeasible direction. The condition to detect this in the case of inequality constraints $g_j \le 0$ is:

$$\begin{cases} g_j(x^{(i)}) > 0 \\ g_j(x^{(i-1)}) > 0 \\ \Delta g_j^{(i)} \ge 0 \\ \Delta g_j^{(i)} = g_j(\boldsymbol{x}^{(i)}) - g_j(\boldsymbol{x}^{(i-1)}) \end{cases} \qquad (20)$$

To bring the design back into feasible domain, the algorithm moves the buffer center in the direction of the feasible domain. That makes the buffer zone non-symmetric around the limit value and increases the $w_j^{(i)}$ buffer coefficient. This function can be understood as moving the real boundaries deeper inside the feasible domain. The new buffer center CBV can thus be found as:

$$CBV_j^{i+1} = CBV_j^{i+1} - (g_j(\boldsymbol{x}^{(i-1)}) - LV_j) \qquad (21)$$

With similar schema (20, 21), the buffer center CBV can be restored back to the limit value, in case the constraint correction factor is too strong. In case of the equality constraint, the condition (20) can be extended as follows:

$$\begin{cases} h_j(x^{(i)}) > 0 \\ h_j(x^{(i-1)}) > 0 \\ \Delta h_j^{(i)} \ge 0 \end{cases} \text{ or } \begin{cases} h_j(x^{(i)}) < 0 \\ h_j(x^{(i-1)}) < 0 \\ \Delta h_j^{(i)} \le 0 \end{cases} \qquad (22)$$

Numerical example Figure 4 shows a typical diagram of the smooth application of the active constraint with a constant step size. In the first 3 iterations, the buffer zone was adjusted to refer to the history of the constraint values. In contrast to Rosen's gradient projection, there is no zigzagging behavior along the constraint limit and no jumps in the objective function values.

3.4 Simplified relaxed gradient projection algorithm

In addition to the relaxed gradient projection method from the previous section, there is a simplified version of the proposed algorithm (SRGP). The SRGP method

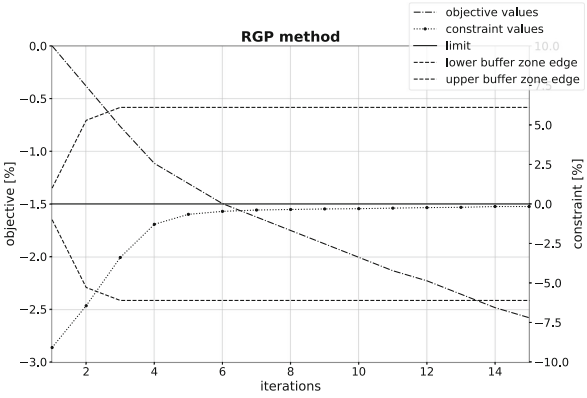


Fig. 4 Smooth adding constraint into the active set using RGP method

does not solve the linear system of equations (7). Instead, it subtracts weighted constraint gradients from negative objective gradient. The weights are calculated in the same way as the buffer coefficient $\omega_j^{(i)}$, (12). In case of a single constraint, the feasible direction is:

$$\boldsymbol{s} = -(1-\omega^{(i)})\nabla f - \omega^{(i)}\nabla g \tag{23}$$

where the $\omega_j^{(i)}$ is the buffer coefficient. In contrast to RGP method, it differs in the range [0; 1] and can be calculated as:

$$\omega^{(i)} = \frac{g(\boldsymbol{x}_i) - LBV}{2 \cdot BS} \tag{24}$$

In the case of multiple constraints, the search direction can be found as follows:

$$\begin{aligned} \boldsymbol{s} &= -(1-\omega^{(i)})\nabla f - \boldsymbol{N\omega}^{(i)}, \\ \omega^{(i)} &= \max_j(\omega_j^{(i)}), \text{ or} \\ \omega^{(i)} &= \sum_j^n(\omega_j^{(i)})/n \end{aligned} \tag{25}$$

where $\boldsymbol{\omega}^{(i)}$ is a vector with size r, which consist of the buffer coefficients $\omega_j^{(i)}$. The weight $\omega^{(i)}$ of the objective gradients can be defined in several ways. It can be the maximum or the average of the $\boldsymbol{\omega}_j^{(i)}$ buffer coefficients.

The simplified relaxed gradient projection method uses the same buffer zone (3.1) and buffer-adaptation functions (Section 3.3) as the RGP method to damp the zigzagging. Unlike the RGP method, SRGP does not calculate the projection direction. Therefore, it cannot follow the linear constraints. In general, the simplified method oscillates constraint and objective values more, and requires more iterations to damp the zigzagging behavior. The advantages of the simplified method are the lack of the need to solve the Lagrangian multipliers (7); that it can work with a large number of constraints; and that it is easy to implement in the optimization framework. All vectors, $\nabla f(\boldsymbol{x}^{(i)})$ and $\nabla g_j(\boldsymbol{x}^{(i)})$, are scaled using max norm. Figure 5 shows the possible directions calculated via the SRGP method (light gray sector). In contrast to the RGP method, the light gray sector is larger and the direction, which is calculated when the constraint value is on the limit ($\omega_j^{(i)} = 0.5$), points towards the feasible domain. Above all, the simplified relaxed gradient projection method can be effectively used in different optimization problems.

Numerical example In Fig. 6, one can see the typical diagram of the smoothing applied on the active constraint with constant step size. In the first 3 iteration, the method adjusts the buffer zone by referring to the history of the constraint values, like in the RGP case. There is a small violation of the constraint while following it. Therefore, the buffer zone slightly changes through iterations. The performance is similar to the RGP case in this simple example.

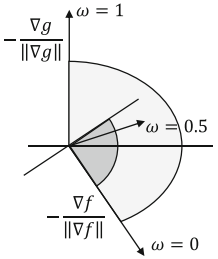


Fig. 5 Possible search direction inside buffer zone, calculated by SRGP method

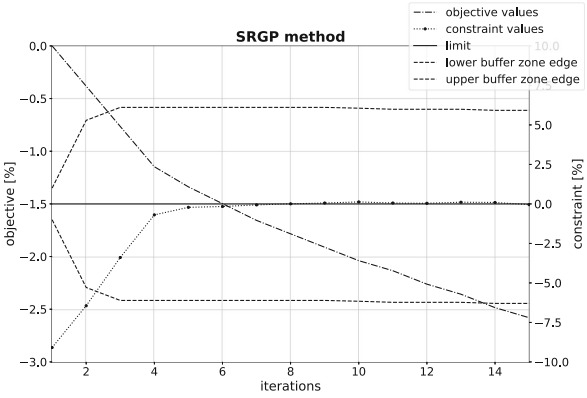


Fig. 6 Smooth adding constraint into the active set with simplified RGP method

3.5 Optimization algorithm

The Algorithm 1 "Relaxed Gradient Projection" is written as the simplified pseudo-code to highlight important steps and to show their order. An interested reader can see more details in the provided python script (see additional materials).

Algorithm 1 Relaxed gradient projection.

$i \leftarrow 0$
// Optimization Loop
for $i = 1, 2, \ldots$ **do**
  Calculate $f(\boldsymbol{x}^{(i)})$, $g_j(\boldsymbol{x}^{(i)})$
  UpdateBufferZones() // (14)
  CalculateBufferCoefficient() // (12, 13)
  ApplyBufferAdaptiveFunctions() //(18, 21)
  ComputeGradients() $\nabla f, \nabla g_j$
  NormalizeSensitivities() $\frac{\nabla f}{||\nabla f||_{max}}, \frac{\nabla g}{||\nabla g||_{max}}$
  ComputeSearchDirection() $(\boldsymbol{s})^{(i)}$ // (17)
  UpdateDesignParameters() $(\boldsymbol{x})^{(i+1)} \leftarrow (\boldsymbol{x})^{(i)} + \alpha(\boldsymbol{s})^{(i)}$
  CheckConvergence()
  $i \leftarrow i + 1$
end for

4 Numerical examples

To demonstrate the performance of the proposed method, several optimization problems are solved. The main focus in the practical problems is to compare the gradient projection method with its modified "relaxed" version. Results of the simplified relaxed gradient method are shown, but they are not in the main focus of discussions. All methods are implemented in the optimization framework "ShapeModule" (BMW Group). Altair Optistruct™ software is used to solve the structural primal and adjoint analysis of the numerical models.

4.1 Analytical optimization problems

Solution of the test examples for nonlinear programming codes from Hock and Schittkowski (1981) is solved. Description, the reference solutions, and RGP solutions are shown in Tables 1, 2, and 3. All problems are solved with the constant step size. No scaling for the sensitivities is used. The results show that the RGP method is not efficient in solving analytical problems. The main reason for that is a constant step size. After several iterations, the parameter update is extremely low. Using line search techniques can improve the performance of the method. In practical cases, rather than accurately converging to the minima, it is needed to find sufficient improvement. For instance, in the Problem # 43, after 50 iterations, all constraints are satisfied and the objective value is $f(x) = -42.75$.

4.2 Structural optimization problem

Our first solved optimization problem is the typical mass reduction of the model, while the compliance of the model should satisfy the given limit value. The experiment shows the difference in the performance of the gradient projection and relaxed gradient projection when constraints are activated during the optimization process.

Case description In Fig. 10, the geometry of the model can be seen from 2 different sides. It is the fixture for soft-top attachment in the BMW i8 Roadster. The blue color indicates the parts of the model that are damped, which means the optimization algorithm cannot move them.

Table 1 Example 1

Problem	#2
Classification	PBR-T1-2
Number of variables	$n = 2$
Number of constraints	$m = 1$
Objective function	
	$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$
Constrained function	
	$1.5 \leq x_2$
Start	
	$x_0 = (-2, 1)$
	$f(x_0) = 909$
Reference solution	
	$x^* = (2a\ \cos(\frac{1}{3}\ \arccos(1/b)), 1.5)$
	$a = (598/1200)^{0.5}$
	$b = 400a^3$
	$f(x^*) = 0.0504261879$
RGP solution	
	$x^* = (1.22437007, 1.49999902)$
	$f(x^*) = 0.05042600898328847$
	$g(x^*) = 9.765624997548628e - 07$
	$n_{\text{iter}} = 261$
RGP settings	
step size	$5e - 4$

Table 2 Example 2

Problem	#22
Classification	QQR-T1-6
Number of variables	$n = 2$
Number of constraints	$m = 2$
Objective function	
	$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$
Constrained function	
	$-x_1 - x_2 + 2 \geq 0$
	$-x_1^2 + x_2 \geq 0$
Start	
	$x_0 = (2, 2)$
	$f(x_0) = 1$
Reference solution	
	$x^* = (1.0, 1.0)$
	$f(x^*) = 1.0$
RGP solution	
	$x^* = (0.99999962,\ 1.00000126)$
	$f(x^*) = 1.0000007616095747$
	$g_1(x^*) = -8.769388442075865e - 07$
	$g_2(x^*) = 2.0193504708387877e - 06$
	$n_{\text{iter}} = 102$
RGP settings	
step size	$5e - 2$

The red parts are design variables. There is a transition zone between blue and red parts, where the model can be modified to maintain continuity between damped and design parts. The optimization problem can be formulated as follows:

$$\begin{aligned} &\textbf{minimize} : \text{mass}(\boldsymbol{x}) \\ \text{s.t.} : &\text{compliance}_1(\boldsymbol{x}) \leq \text{compliance}_1(\boldsymbol{x}^{(0)}) * 1.1 \\ &\text{compliance}_2(\boldsymbol{x}) \leq \text{compliance}_2(\boldsymbol{x}^{(0)}) * 1.1 \end{aligned} \tag{26}$$

where $\text{mass}(\boldsymbol{x})$ is the mass response function, $\boldsymbol{x}$ is a vector of the design parameters, $\boldsymbol{x}^{(0)}$ is the initial state, and $\text{compliance}_1(\boldsymbol{x})$ and $\text{compliance}_2(\boldsymbol{x})$ are the compliance responses according to the different load cases. The load case 1 is applied in y-direction and load case 2 in z-direction. Both loads are applied on the upper blue zone while the bottom blue part of the FE model is fixed (Fig. 10a). In the case of node-based parametrization, the surface nodes are used as design variables. The size of the $\boldsymbol{x}$ is 145,326. Our stopping criteria is a maximum number of iterations (50). Further optimization steps are not useful, as the mesh quality is not acceptable. The maximum shape update magnitude is constant and equals 0.15 mm.

Optimization method settings Table 4 shows the settings we have used for the methods. The correction coefficient scales the restoring move (11) of the violated constraint. As the shape update is limited, the coefficient helps to balance the impact of the violated constraint with respect to other responses to the final shape update. The parameters are tuned in a way that violated constraints can be corrected in one step. With smaller coefficients, the violations are initially smaller, but after some iterations, method diverges because it can not handle constraints anymore.

Results Figure 7 gives the graphs with the compared objective and constraint values through optimization iterations. The gradient projection algorithm detects the violated constraint $\text{compliance}_2(\boldsymbol{x})$ at iteration 6, adds it to the active set of the constraints, and applies the correction move to bring the solution back into the feasible domain. At iteration, 7, the solution is feasible, and the constraint is removed from the active set of constraints. Hence, the algorithm does not consider it. Therefore, the constraint value is again violated in the next iteration. After this point, marked zigzagging behavior can be seen for objective and constraint values. In contrast, a relaxed gradient projected algorithm has no zigzagging behavior after iteration 6 or 7. If we compare constraint values precisely, in Fig. 7b, c, one can see that up until iteration 4, both algorithms calculate the same shape update because both methods perform the steepest descent step. At iteration 5, the RGP method detects the constraint

Table 3 Example 3

Problem	#43
Classification	QQR-T1-11
Number of variables	$n = 4$
Number of constraints	$m = 3$
Objective function	
	$f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2$ $-5x_1 - 5x_2 - 21x_3 + 7x_4$
Constrained function	
	$8 - x_1^2 - x_2^2 - x_3^2 - x_4^2 -$ $x_1 + x_2 - x_3 + x_4 \geq 0$
	$10 - x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2$ $+x_1 + x_4 \geq 0$
	$5 - 2x_1^2 - x_2^2 - x_3^2$ $-2x_1 + x_2 + x_4 \geq 0$
Start	
	$x_0 = (0, 0, 0, 0)$
	$f(x_0) = 0$
Reference solution	
	$x^* = (0.0, 1.0, 2.0, -1.0)$
	$f(x^*) = -44$
RGP solution	
	$x^* = (1.31159684e-07, 1.0,$ $2.0, -9.99999586e-01)$
	$f(x^*) = -43.999999009762654$
	$g_1(x^*) = 9.520910907445668e-07$
	$g_2(x^*) = 1.0000019930022122$
	$g_3(x^*) = 1.9072899259953147e-08$
	$n_{\text{iter}} = 2766$
RGP settings	
step size	$5e-2$

compliance$_2(\boldsymbol{x})$ as active and adds it to the active set of constraints with the relaxation coefficient. At iteration 9, RGP adds constraint compliance$_1(\boldsymbol{x})$ in the same way. Therefore, the RGP method smoothly and in advance adds constraints to the active set. Still, both constraints were violated during the optimization process, but the amount of the violation is lower than in GP. For instance, after 20 iterations, the constraint compliance$_2(\boldsymbol{x})$ starts to zigzag around the limit, but the RGP method is able to damp the oscillations by using the adaptation function (21) (see Fig. 8). While constraint compliance$_2(\boldsymbol{x})$ is oscillating, the objective function is still improving with nearly the same speed as in the previous iterations. With regard to the results, the objective function is improved faster using the RGP method compared to GP (Fig. 9). During $5 - 50$ optimization iterations, while both optimization algorithms are traveling along the design boundaries, the GP method is able to improve our objective function by 9.5% and RGP by 14.4%. Hence, the RGP method improves the objective function 1.7 times faster than GP in this case. SRGP method sees nearly the same improvement of the objective function as RGP, but the oscillations of the constraint value around the limits are greater. In Fig. 10b, the comparison between initial and optimum design (RGP method) can be seen.

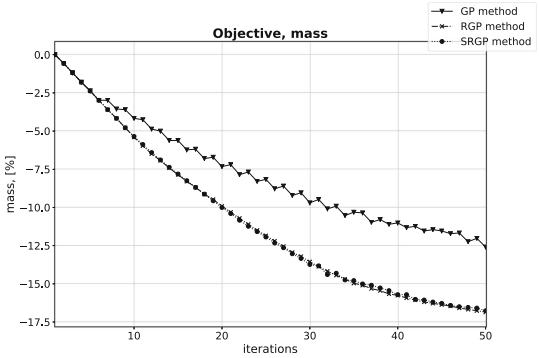
Table 4 Optimitation method settings

Gradient projection	
Step size	0.15
Compliance constraint corr. coeff.	1.0
Compliance constraint corr. coeff.	1.0
RGP and SRGP	
Step size	0.15
Buffer scale factor eq. (18)	1
Initial BSF	2.0

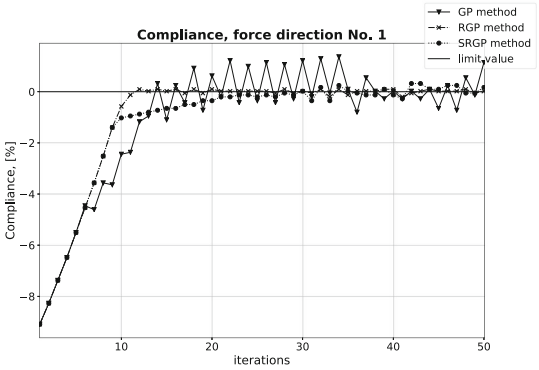
In Fig. 11, there is a comparison in the shape updates between GP and RGP methods. At iteration 1, both methods perform the steepest descent step; therefore, the shape updates are similar in both cases. The difference occurs at iteration 5. The GP method does not detect the constraint compliance$_2(\boldsymbol{x})$ and continues performing steepest descent step. RGP already detects the constraint and calculates "constrained" shape update. At iterations 6–8, the GP method strongly modifies the shape updates: at iteration 6, there is a shape update with a correction move, at iteration 7, there is the steepest descent update, and at iteration 8, there is again a shape update with a correction move. The behavior continues during the whole optimization process. In contrast, the RGP method smoothly modifies the shape updates during iterations 5–8 and beyond. The RGP method tries to find the feasible search direction that will not significantly oscillate the constraints. Therefore, there are fewer oscillations compare to GP.

Figure 12 shows the difference in the final shapes. There is no big difference between SRGP and RGP generated shapes. The shape, generated by the GP method, has less modification of the design due to problems with zig zagging. All methods has similar pattern of the update, therefore all of them converging to similar local optima. The reason for that is that all methods are trying to follow similar optimization path along the active sets of the constraints. If there are no active constraints, all method use steepest descent direction. If the initial design or the parametrization will be changed, new final design can be found.

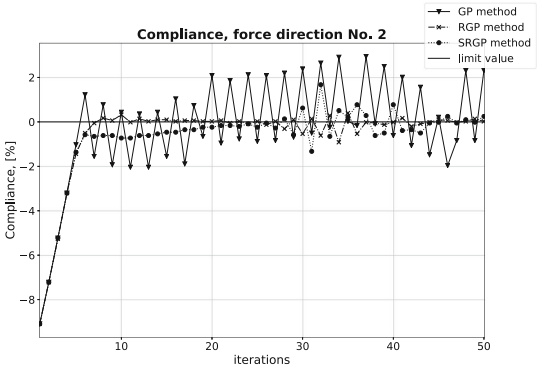
Conclusion The RGP method demonstrates good performance in this case, compared to the GP method. It was able to smoothly activate the constraints as they approached the limit values. If the zigzagging of the constraint is detected,



(a)



(b)



(c)

Fig. 7 GP vs RGP vs SRGP method comparison. **a** Objective values. **b** Constraint compliance1(x) . **c** Constraint compliance2(x)

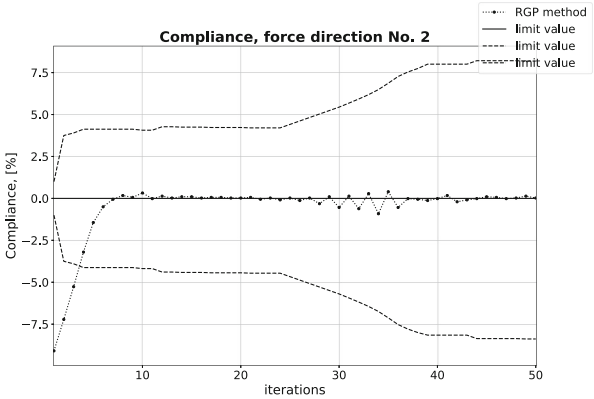


Fig. 8 RGP method, constraint compliance2($\boldsymbol{x}$) with buffer zone

the method is able to damp them, while the objective function keeps nearly the same ratio of improvement. The SRGP method exhibits similar performance to the RGP method, although the constraint violations are greater in number.

4.3 Structural optimization problem with geometrical constraint

In the second numerical example, the structural optimization problem with a geometrical constraint should be solved. In practical cases, it is common to apply geometrical constraints because the designed part exists within a given space so as not to collide with neighboring parts; therefore, the shape boundaries should be fixed. Details of the packaging constraint can be found in Najian Asl et al. (2017).

Case description The objective function is to reduce the mass of the initial model with respect to two constraints: the model should be inside a packaging box (geometrical constraint) and the maximum displacement of any point should be below the given limit. The problem can be defined as follows:

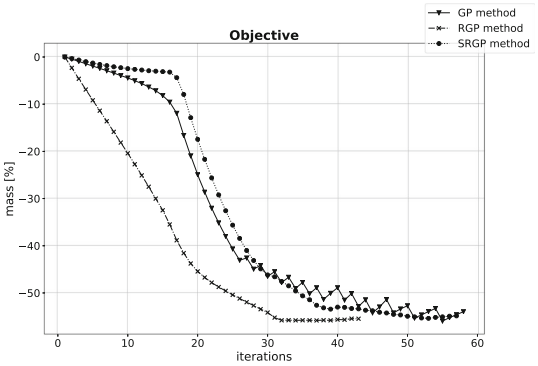
$$\begin{aligned} &\textbf{minimize: } \text{mass}(\boldsymbol{x}) \\ &\quad \textbf{s.t. } d_i \leq d^{crit} \\ &\qquad x_i \in \boldsymbol{V^c} \end{aligned} \tag{27}$$

where $\boldsymbol{x}$ is a vector of the design parameters, mass($\boldsymbol{x}$) is a mass response, d_i is the displacement at the point i, and $\boldsymbol{V^c}$ is the packaging constraint. The initial configuration can be seen in Fig. 13a, where the red zones are the design nodes and the blue zones are damped and cannot be moved. Figure 13b and c show which parts initially violate the geometrical constraint and should be corrected during the optimization process. The stopping criteria are the maximum number of optimization iterations (100) or the lack of sufficient improvement in the objective function (less than 0.1% in the last 10 iterations). The maximum shape update magnitude is constant and equals to 0.5 mm.

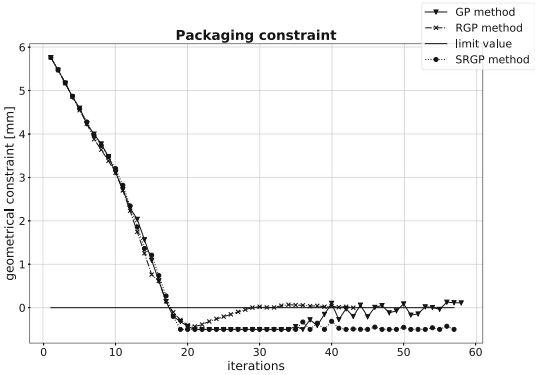
Optimization method settings Table 5 shows the settings we have used for the methods. The correction coefficients were tuned in a way that packaging constraint is less dominant during the optimization process.

Results In Fig. 9, there is a comparison of the objective and constraint values. Initial design is infeasible with respect to packaging constraint; therefore, all methods perform their first iterations to correct the packaging constraint. On graph Fig. 9a, it can be seen that the RGP method improves the objective function much faster than the other two methods. In the case of SRGP, the method performs the "steepest descent steps" in the direction of the violated geometrical constraint ($\boldsymbol{s} = 0 \cdot \nabla f - 1 \cdot \boldsymbol{N}$). The GP method has calculated a correction move; hence, most of the shape update is performed to correct the constraint rather than improve the objective function. In the RGP method, the correction term is limited by the correction coefficient $\omega^{max} = 2$ (see Section 3.2); therefore, the correction is not as high as for the other, and the objective function is improved faster. When the packaging constraint value is corrected, the speed of objective improvement increases.

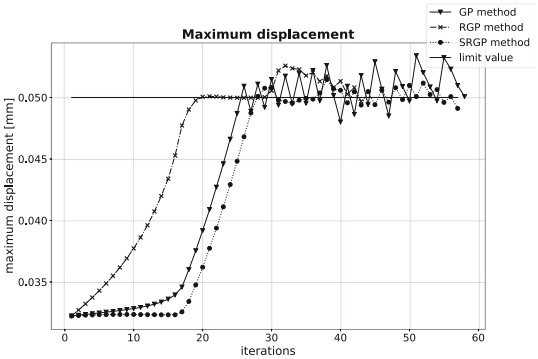
Figure 14d provides a comparison of the initial and final design of the model. The initial design is in transparent blue, and the optimized design is in white. Visible are the zones where the mass was reduced and the zone where the shape was modified to satisfy the packaging constraint. All methods converged towards a similar optimized design,



(a)



(b)



(c)

Fig. 9 GP vs RGP vs SRGP methods compared. **a** Objective values. **b** Constraint compliance1(x). **c** Constraint compliance2(x)

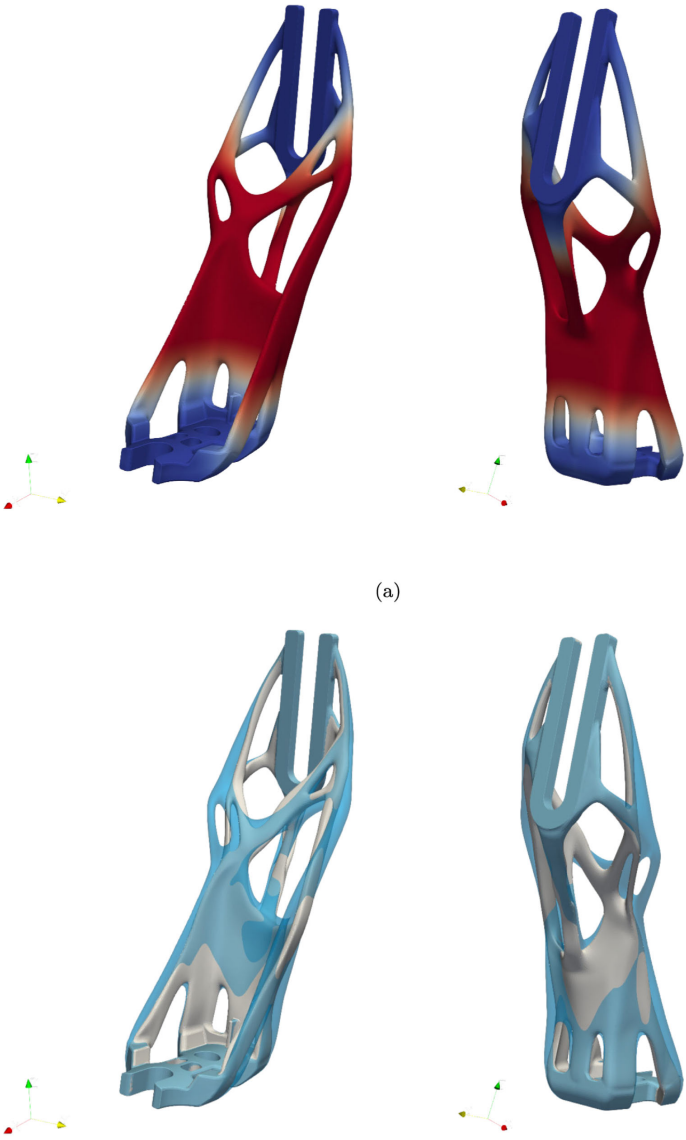


Fig. 10 Optimization model. **a** Geometry of the optimization model, red design part, blue damped part. **b** Initial design (transparent blue), optimized design (white) via RGP method

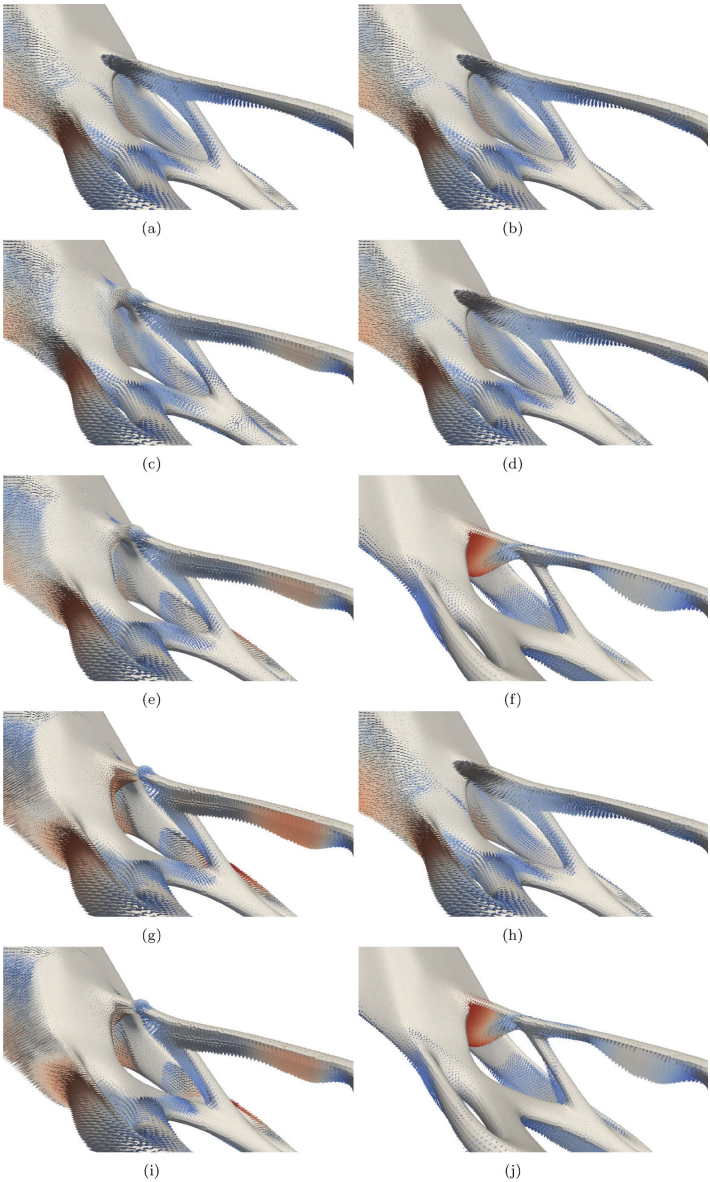


Fig. 11 Comparison of the shape update (scaled), calculated by GP and RGP methods. **a** Iteration 1, RGP. **b** Iteration 1, GP. **c** Iteration 5, RGP. **d** Iteration 5, GP. **e** Iteration 6, RGP. **f** Iteration 6, GP. **g** Iteration 7, RGP. **h** Iteration 7, GP. **i** Iteration 8, RGP. **j** Iteration 8, GP

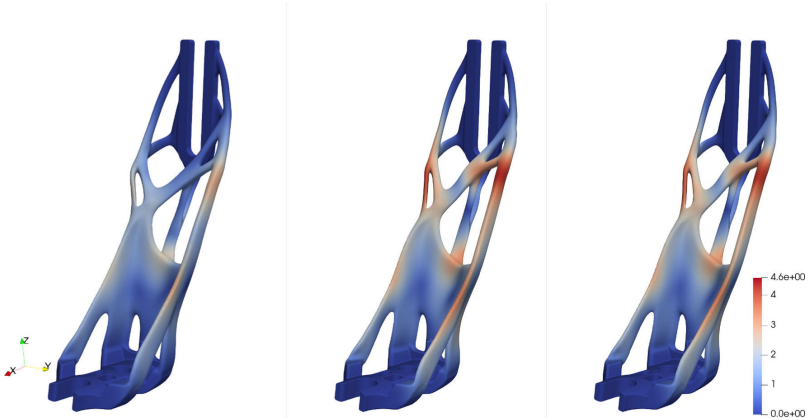


Fig. 12 Comparison of the optimized designs. Data filed shows absolute update in (mm). From left to right: GP, RGP, SRGP

(see Fig. 14a–c). The optimization process was stopped by the convergence criteria because there was no more sufficient improvement in the objective function, and the constraints are satisfied within the given tolerance.

An important difference in performance between the RGP method and others is that in this case the RGP method is much more stable and is able to maintain its geometrical constraint very accurately. There is a violation in the maximum displacement constraint at the 28th iteration, but the method is able to correct the constraint. In case of the SRGP method, there are more oscillations of the constraint values, but the violations are lower. The method was not as efficient as RGP due to the initially slow improvement of the objective. When the geometrical constraint was corrected, the method performed well, and in a stable manner. The GP method displayed similar issues with zigzagging as in the previous examples. Nonetheless, the GP method finds the solution with the same number of iterations as the SRGP method. It is important to note that the computational time for one iteration for each method is similar because there is the same number of calls for analysis and they take up the most time.

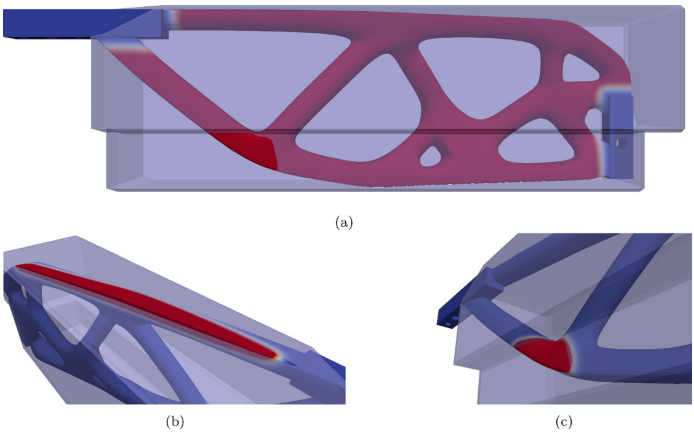


Fig. 13 Optimization model, packaging constraint(light purple). **a** Design model, red design part, blue damped part. **b** Geometrical constraint violation of the initial design, upper side (red). **c** Geometrical constraint violation of the initial design, lower side

Table 5 Optimitation method settings

Gradient projection	
Step size	0.5
Maximum displacement constraint corr. coeff.	1.0
Packaging constraint corr. coeff.	0.05
RGP and SRGP	
Step size	0.5
Buffer scale factor (18)	1
Initial BSF	2.0

Table 6 Computation time

Method	GP	RGP	SRGP
Aver. time, 1 Iter, s	247	250	248
Full time, s	14326	10750	14136

4.4 Computational time

Table 6 shows the comparison of the computational time needed to solve the optimization problem from the Section 4.3. Intel Xeon(R) CPU E5-1650 v4 with 6 cores was used in cooperation with 64 GiB RAM. 137 s is needed to solve primal and adjoint solutions and less than 0.7 s to find shape update with constant step size. Rest of the time is needed to compute parametrization, mesh update, and save the output files.

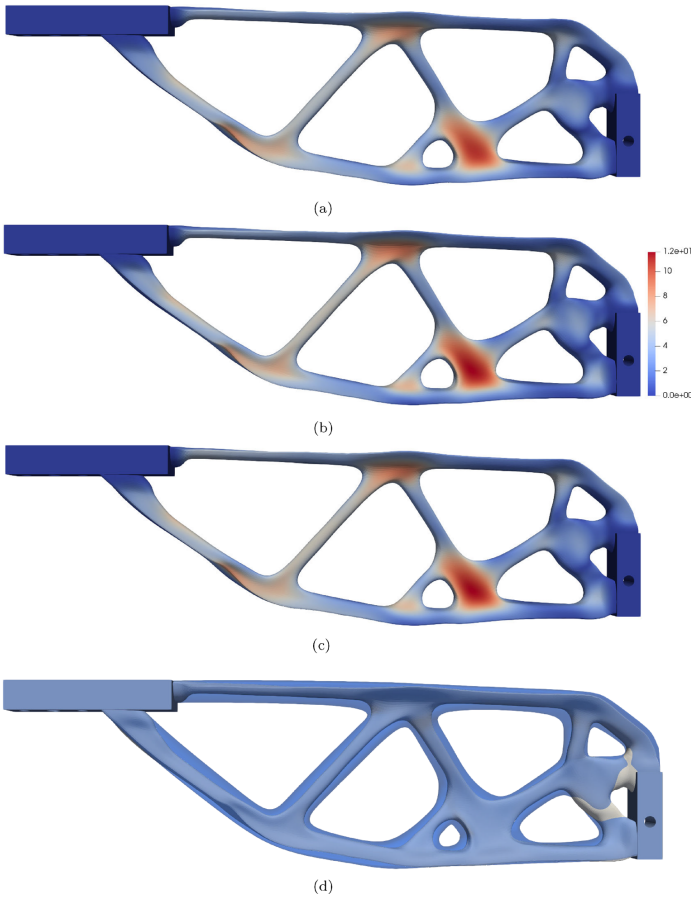


Fig. 14 Comparison of the optimized designs. Data filed shows absolute update in (mm). **a** Gradient Projection method method **b** Relaxed Gradient Projection method **c** Simplied Relaxed Gradient Projection method **d** Initial design (transparent blue), optimized design (white) via RGP method

5 Conclusions

In this paper, the relaxed gradient projection method is introduced. The proposed modifications to Rosen's gradient projections show good speed up in the rate of the objective improvement, and in avoiding marked zigzagging behavior. The proposed method can activate the constraint in advance before the limit value is reached, and it has techniques to reduce the zigzagging behavior while following the constraint boundaries. It does not require accurate parameter set up; therefore, it is easier and stable in daily practice. Further research can be done to find efficient line-search techniques that can be efficiently used in the practical applications, and computing more accurately the correction part of the search direction. In conclusion, we see the relaxed gradient projection algorithm as being one of the group of feasible direction methods. We do not claim that the proposed method is the best option for shape optimization problems in general. The proposed algorithm should be considered as a good alternative to other successful optimization methods, such as inner-point (Chen et al. 2019) or trust-region algorithms (Yuan 1999).

Supplementary Information The online version contains supplementary material available at 10.1007/s00158-020-02821-y.

Acknowledgments Open Access funding enabled and organized by Projekt DEAL. The authors wish to thank the ShapeModule project (BMW Group) for the support. Thanks are also due to S. Stahl M.Sc. (Design for Additive Manufacturing, BMW Group) for providing numerical models.

Funding This paper is based on a part of the research sponsored by the BMW group.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results The proposed method is implemented in the optimization framework "ShapeModule" (BMW Group, shapemodule@bmw.de). The code has no public access, as well as shown models. Following suggestions from Haftka et al. (2019), to improve the replication of results, the proposed method is implemented in the python script (see additional materials). The script has all significant steps of the relaxed gradient projection method, and it solves a constrained quadratic problem. In the script, all methods have references to the corresponding equations. Additionally, the constant parameter update and scaling of the sensitivities are applied. An interested reader can contact the corresponding author for the respective explanations.

References

Agarwal D, Robinson T, Armstrong C, Kapellos C (2018) Enhancing cad-based shape optimisation by automatically updating the cad model's parameterisation. Struct Multidiscip Optim 59:11. https://doi.org/10.1007/s00158-018-2152-7

Baumgärtner D, Viti A, Dumont A, Carrier G, Bletzinger K-U (2016) Comparison and combination of experience-based parametrization with vertex morphing in aerodynamic shape optimization of a forward-swept wing aircraft 06. https://doi.org/10.2514/6.2016-3368

Bletzinger K-U (2017) Shape optimization, pp 1–42. ISBN 9781119176817. https://doi.org/10.1002/9781119176817.ecm2109

Chen L, Bletzinger K-U, Geiser A, Wüchner R (2019) A modified search direction method for inequality constrained optimization problems using the singular-value decomposition of normalized response gradients. Struct Multidiscip Optim 06:1–19. https://doi.org/10.1007/s00158-019-02320-9

Du D, Wu F, Zhang X (1990) On rosen's gradient projection methods. Ann Oper Res 24:9–28, 12. https://doi.org/10.1007/BF02216813

Firl M, Bletzinger K-U (2012) Shape optimization of thin walled structures governed by geometrically nonlinear mechanics. Comput Methods Appl Mech Eng 237–240:107–117, 09. https://doi.org/10.1016/j.cma.2012.05.016

Fletcher R (2013) General linearly constrained optimization, chap 11. Wiley, New York, pp 259–276. https://doi.org/10.1002/9781118723203.ch11. ISBN 9781118723203

Gallagher R, Zienkiewicz O (1977) Optimum structural design—theory and applications 01

Haftka RT, Grandhi RV (1986) Structural shape optimization—a survey. Comput Methods Appl Mech Eng 57(1):91–106. https://doi.org/10.1016/0045-7825(86)90072-1. ISSN 0045-7825

Haftka R, Kamat M (1990) Element of structural optimisation, vol 1 01. https://doi.org/10.1007/978-94-015-7862-2

Haftka RT, Zhou M, Queipo NV (2019) Replication of results. Struct Multidiscip Optim 60(2):405–409. https://doi.org/10.1007/s00158-019-02298-4. ISSN 1615-1488

Hardee E, Chang K-H, Tu J, Choi K, Grindeanu I, Yu X (1999) A cad-based design parameterization for shape optimization of elastic solids. Adv Eng Softw 30:185–199, 03. https://doi.org/10.1016/S0965-9978(98)00065-9

Heners J, Radtke L, Hinze M, Düster A (2017) Adjoint shape optimization for fluid-structure interaction of ducted flows. Comput Mech 61:259–276, 08. https://doi.org/10.1007/s00466-017-1465-5

Hock W, Schittkowski K (1981) Test examples for nonlinear programming codes, vol 187. Springer, Berlin. https://doi.org/10.1007/978-3-642-48320-2

Hojjat M, Stavropoulou E, Gallinger T, Israel U, Wüchner R, Bletzinger K-U (2010) Fluid-structure interaction in the context of shape optimization and computational wind engineering. 73: 351–381, 09, Fluid Structure Interaction II, Springer, Berlin Heidelberg

Hojjat M, Stavropoulou E, Bletzinger K-U (2014) The vertex morphing method for node-based shape optimization. Comput Methods Appl Mech Eng 268:494–513, 01. https://doi.org/10.1016/j.cma.2013.10.015

Kenway G, Kennedy G, Martins J (2014) Aerostructural optimization of the common research model configuration, 06. ISBN 978-1-62410-283-7. https://doi.org/10.2514/6.2014-3274

Kroll N, Gauger N, Brezillon J, Dwight R, Vollmer D, Becker K, Barnewitz H, Schulz V, Hazra S (2007) Flow simulation and shape optimization for aircraft design. J Comput Appl Math 203:397–411, 06. https://doi.org/10.1016/j.cam.2006.04.012

Luo Z, Wang M, Wang S, Wei P (2008) A level set-based parameterization method for structural shape and topology optimization. Int J Numer Methods Eng 76:1–26, 10. https://doi.org/10.1002/nme.2092

Najian Asl R, Shayegan S, Geiser A, Hojjat M, Bletzinger K-U (2017) A consistent formulation for imposing packaging constraints in shape optimization using vertex morphing parametrization. Struct Multidiscip Optim 56:1–13, 10. https://doi.org/10.1007/s00158-017-1819-9

Palacios F, Economon T, Alonso J (2012) Optimal shape design for open rotor blades 06. https://doi.org/10.2514/6.2012-3018

Rosen J (1960) The gradient projection method for nonlinear programming. Part i. Linear constraints. J Soc Ind Appl Math 8:03. https://doi.org/10.1137/0108011

Rosen J (1961) The gradient projection method for nonlinear programming: part ii. SIAM J Appl Math - SIAMAM 9:01

Sieger D, Menzel S, Botsch M (2012) A comprehensive comparison of shape deformation methods in evolutionary design optimization

Sun W, Yuan Y-X (2006) Optimization theory and methods. Nonlinear Program 1:01. https://doi.org/10.1007/b106451

Ummidivarapu V, Voruganti H (2017) Shape optimisation of two-dimensional structures using isogeometric analysis. Int J Eng Syst Model Simul 9:169, 01. https://doi.org/10.1504/IJESMS.2017.085080

Ummidivarapu V, Voruganti H, Khajah T, Bordas S (2020) Isogeometric shape optimization of an acoustic horn using the teaching-learning-based optimization (tlbo) algorithm. Comput Aided Geom Des 80:101881, 05. https://doi.org/10.1016/j.cagd.2020.101881

Vanderplaats G (2007) Multidiscipline design optimization. Vanderplaats Research & Development Inc

Wang M, Luo Z (2020) Shape and topology optimization for compliant mechanisms using level set-based parameterization method, pp 18–21, 05

Xu S, Jahn W, Mueller J-D (2014) Cad-based shape optimisation with cfd using a discrete adjoint. Int J Numer Methods Fluids 74:01. https://doi.org/10.1002/fld.3844

Yuan Y-X (1999) A review of trust region algorithms for optimization. In: ICM99: proceedings of the fourth international congress on industrial and applied mathematics 09

Appendix B

Publication II

Structural and Multidisciplinary Optimization (2022) 65:198
https://doi.org/10.1007/s00158-022-03279-w

RESEARCH PAPER

Latest developments in node-based shape optimization using Vertex Morphing parameterization

Ihar Antonau[1] · Suneth Warnakulasuriya[1] · Kai-Uwe Bletzinger[1] · Fabio Michael Bluhm[2] · Majid Hojjat[3] · Roland Wüchner[4]

Received: 26 February 2022 / Revised: 12 May 2022 / Accepted: 16 May 2022

Abstract
The latest updates on the Vertex Morphing technique for large optimization problems are shown in this work. Discussions about the challenges of node-based shape optimization in academic and industrial applications are included. The adaptive Vertex Morphing technique is demonstrated, which is easy to use in practice and allows the full exploitation of the potential of node-based shape optimization to find new designs in large-scale applications. We also show an efficient optimization method to handle different physical responses with many geometrical constraints. A state-of-the-art example of industrial importance supports the work.

Keywords Gradient-based constrained optimization · Barzilai–Borwein method · Relaxed gradient projection method · Adaptive Vertex Morphing

1 Introduction

In many industrial applications, adjoint-based shape optimization application with response functions, computed by computational fluid dynamics (CFD) analysis, has become an important analysis tool in the design process of the products (Papoutsis-Kiachagias and Giannakoglou 2014; Müller et al. 2021). In shape optimization, the aim is to find an optimal shape of the model regarding a physical quantity, for instance, drag force.

The choice of the design parameters (parameterization) plays a key role in successful results in realistic optimization problems. There are different methods and strategies to parameterize the design space of large problems. There are two major groups of parameterization techniques: CAD and CAD-free (or parameter-free) methods. CAD methods control the position of "many" surface points based on "few" CAD parameters(Xu et al. 2014; Agarwal et al. 2018). In contrast, CAD-free methods use the surface nodes directly as the design parameters (Firl and Bletzinger 2012; Stück and Rung 2013; Bletzinger 2017; L. A. G and Guillaume 2018). CAD-free methods have difficulties attaining the final shape without rough/noisy boundaries (Stück and Rung 2011). In this regard, several techniques are proposed to increase the regularity of the shape update (Stavropoulou et al. 2014; Kröger and Rung 2015).

Vertex Morphing is a successful CAD-free technique introduced by Hojjat et al. (2014) and Bletzinger (2014), Bletzinger (2017). The main characteristics of the technique are as follows:

1. The method uses the filtering operation to generate smooth design updates and control the surface mesh quality.
2. No extra optimization model is needed. The Finite Element (FE) model is used directly. The Vertex Morphing parameterization is easy to set up for complex models and geometries, and it is a good alternative to well-

Responsible Editor: Kyriakos Giannakoglou

Topical Collection: Flow-driven Multiphysics
Guest Editors: J Alexandersen, C SAndreasen, K Giannakoglou, K Maute, K Yaji

✉ Ihar Antonau
ihar.antonau@tum.de

[1] Technical University of Munich, Munich, Germany

[2] RWTH Aachen University, Aachen, Germany

[3] BMW Group, Munich, Germany

[4] Institut für Statik und Dynamik, Technische Universität Braunschweig, Braunschweig, Germany

Published online: 06 July 2022

known parameterization strategies (Baumgärtner et al. 2016);
3. The Vertex Morphing method can be successfully integrated into the multi-disciplinary optimization framework. An example of such implementation can be found in Ghantasala et al. (2021), Baumgärtner (2020);
4. Vertex Morphing parameterization provides a rich design space, allowing new solutions to be explored. (Bletzinger 2017);
5. Different design constraints, such as symmetry, axis-symmetry, or damped (non-design) zones, can be consistently integrated into the parameterization. This ensures the satisfaction of the given requirements without using advanced constrained optimization algorithms (Najian Asl 2019).

The main parameter to adjust Vertex Morphing is a filter radius ("filtering intensity"). It is an additional design parameter to modify generated shape update modes and define shape features of the initial geometry that will be preserved. In the work of Hojjat et al. (2014), it has been shown that the optimizer converges to different target local optimums by adjusting the filtering radius. The role of the filtering radius can be defined as follows:

1. The optimizer is driven to a certain local minimum by choice of the filter radius;
2. The filter radius directly controls the final shape (smoothness, wavelength);
3. All the initial features of the geometry which are smaller than the filter radius are preserved;
4. *A priori* the "good" size of the filter radius is unknown;
5. During the optimization process, the large deformations of the design surface can change the surface mesh size dramatically. Therefore, the filter radius can become too small concerning surface mesh size, and it will cover only a few layers of the elements. As a result, the consequential following shape updates are not smooth anymore. In contrast, the filter radius may become too large if the model shrinks. Hence, the big parts of the model move as a rigid body.

The adaptive Vertex Morphing is proposed to address the challenges mentioned above. The adaptive Vertex Morphing method computes the smallest radius required for appropriate filtering on each node. As a result, adaptive Vertex Morphing can be used without any user's input, or it can correct the given user's input. Additionally, in contrast to the initial method, in adaptive Vertex Morphing, the user can provide not only "global" radius size but also "local" sizes. In this paper, CFD shape optimization problems are solved using the adaptive Vertex Morphing technique.

Structural optimization problems with Vertex Morphing parameterization require robust and efficient optimization methods to handle many design variables and different physical and geometric constraints. Also, the efficient line search strategy can improve the computation of descent improvement of the objective function and keep the design feasible. Typically, a structural optimization problem with Vertex Morphing has the following properties:

1. A large number of design variables. The "usual" number is $1e4 - 1e6$. Therefore, the sensitivities of the response functions have to be computed using adjoint sensitivity analysis (Najian Asl 2019);
2. For typical engineering optimization tasks, computing the response values of the objective and constraint functions requires numerically expensive CFD (or structural) analysis. Therefore, optimization methods must be robust and efficient to reduce the number of evaluations of response values as much as possible;
3. The objective and constraints functions are typically highly non-linear (Firl and Bletzinger 2012);
4. The sensitivities of the response functions have to be scaled due to the different physical units (m, kg, N, etc.). Therefore the information regarding the magnitude of the raw sensitivities can be lost;
5. Due to a large number of design variables, geometric constraints and design bounds lead to a large number of constraints. An efficient aggregation method may be required (Geiser et al. 2021);
6. Line search techniques can be numerically expensive or non-accurate for highly non-linear functions. In practical application, a constant step size may be preferred.

Generally, a constraint shape optimization problem is formulated as follows:

$$\begin{aligned} \textbf{minimize} &: f(\boldsymbol{x}) \\ \textbf{design variables} &: \boldsymbol{x} \\ \textbf{subject to} &: \\ g_j(\boldsymbol{x}) \leq 0, \text{ where } j &= 1..n_g \\ h_k(\boldsymbol{x}) = 0, \text{ where } k &= 1..n_h \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ represents design parameters that define the design surface, n_g and n_h are numbers of inequality ($g(\boldsymbol{x}) \leq 0$) and equality ($h(\boldsymbol{x}) = 0$) constraints.

The algorithms that have been successfully used with Vertex Morphing parameterization are gradient projection (Najian Asl et al. 2017; Ertl 2020), the relaxed gradient projection (RGP) method (Antonau et al. 2021), and the modified search direction method (Chen et al. 2019; Chen 2021). All methods are first-order direct optimization methods that

can find good search directions to improve the objective functions and handle constraints.

In the mentioned works, a constant step size has been used. However, the good step size is an unknown *a priori* and may lead to poor performance or higher computational cost. There are various methods to calculate the exact or approximate step length to find a minimum of the objective function or sufficient reduction along the descent search direction. For instance, Cauchy methods may require calculating the Hessian matrix, which is not always available or very expensive to compute (Zhou et al. 2006). Besides, Armijo's backtracking schemes try several step sizes until the acceptance criteria are satisfied (Ahookhosh and Ghaderi 2017). In large CFD optimization problems, additional functional evaluation may excessively increase the computational cost of each optimization iteration.

On the other hand, the Barzilai–Borwein (BB) method (Barzilai and Borwein 1988) attracts many research groups because of its simplicity and surprising efficiency in unconstrained optimization problems. The method's main advantage is that it does not require any costly computational operations to approximate the step size. There are various modifications of the technique: Projected Barzilai–Borwein method (Dai and Fletcher 2005), Adaptive Barzilai–Borwein method (Zhou et al. 2006), Stabilized Barzilai–Borwein method (Oleg Burdakov and Dai 2019), and accelerated Barzilai–Borwein method (Huang et al. 2022).

The Quasi-Newton Barzilai–Borwein (QN–BB) method is introduced in this work. In contrast to the original and modified versions, the QN–BB method computes each design variable's step size independently. Therefore, each design parameter has its step size based on the local sensitivity information. The QN–BB method has been coupled with the relaxed gradient projection method (QN–BB–RGP). The QN–BB–RGP algorithm uses a linear approximation of the constraints to compute feasible design updates. It allows for solving efficiently large optimization problems with localized constraints. Additionally, in this work, the maximum-value aggregation technique is introduced. It combines a large number of nodal constraints into one, where each node has an individual correction based on the nodal constraint value. The QN–BB method was first time shown at Eccomas Congress 2020 & 14th WCCM; the record of the presentation talk can be found under the link (https://slideslive.com/38944933).

The paper is structured as follows: First, the Vertex Morphing method is reviewed, and the proposed adaptive Vertex Morphing is introduced. Then the QN–BB–RGP method is described with all its components: QN–BB, RGP, and the max-value aggregation technique. The following sections describe the academic and industrial optimization problems and show a detailed analysis of the performance of the proposed methods. Finally, conclusions are drawn from the work.

2 Vertex Morphing

Without appropriate regularization measures, node-based shape optimization produces high-frequency, noisy geometries. Therefore, one means of choice is to subject the raw geometry to smoothing using filters. In the context of Vertex Morphing, thus, the structural geometry $\boldsymbol{x}$ is indirectly controlled by a control field $\boldsymbol{s}$ and a kernel or filter function A, for example, on the surface Γ with surface coordinates (ξ, η, ζ):

$$\boldsymbol{x}(\xi_0, \eta_0, \zeta_0) = \int_\Gamma A(\xi - \xi_0, \eta - \eta_0, \zeta - \zeta_0)\boldsymbol{s}(\xi, \eta, \zeta)d\Gamma. \quad (2)$$

Vertex Morphing belongs to the direct filtering techniques as opposed to the indirect ones, such as Sobolev smoothing (Jameson 1988, 1995; Pironneau 1974), where the filter is applied to the actual geometry $\boldsymbol{x}$. There is great freedom to choose kernel functions. For the choice of simple polynomials on compact support (including a piecewise linear hat function and splines), it is shown that Vertex Morphing is identical to a generalized CAD-based approach with indirectly defined spline base functions (Bletzinger 2017). When taking the Gauss bell-shaped distribution function, the technique has additional equivalent properties compared to indirect smoothing (Stück and Rung 2011).

After discretization of the structural geometry $\boldsymbol{x} = [x_1^x, x_1^y, x_1^z, ..., x_n^x, x_n^y, x_n^z]$ and control function $\boldsymbol{s} = [s_1^x, s_1^y, s_1^z, ..., s_m^x, s_m^y, s_m^z]$ by standard techniques as the finite element method, Vertex Morphing appears as follows:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \quad (3)$$

where $\boldsymbol{x}$ is the vector of coordinates of nodes where the spatial coordinates in $x-$, $y-$, and $z-$ direction of a node are arranged sequentially. $\boldsymbol{A}$ is the filter operator matrix, and $\boldsymbol{s}$ is the vector of discrete control field parameters, again arranged sequentially. The most straightforward approach is to add control parameters to every node, i.e., vertex, of the finite element model, which motivates the term "Vertex Morphing."

The entries A_{ij} of $\boldsymbol{A}$ reflect the filter effect as the interaction between two different nodes i and j, their spatial position vectors x_i and x_j, and their Euclidean distance $\left|x_i - x_j\right|$. For the case of the Gauss distribution as kernel and approximating integration by summation it holds:

$$A_{ij} = \frac{F(x_i, x_j)}{sum}$$
$$sum = \sum_j F(x_i, x_j), \tag{4}$$

where

$$F(x_i, x_j) = \begin{cases} \exp\left\|(-\left|x_i - x_j\right|^2 / 2r^2)\right\| & \left|x_i - x_j\right| < r \\ 0.0 & \left\|x_i - x_j\right\| \geq r \end{cases} \tag{5}$$

and r is the filter radius. Different from the initial version of Vertex Morphing (Hojjat et al. 2014), the same filter operations are applied of any item assigned to a discrete node, which are in particular each component of the spatial coordinates. As a consequence, the entries $\boldsymbol{A}_{ij}$ appear as scalar matrices in $\boldsymbol{A}$:

$$\boldsymbol{A}_{ij} = A_{ij} * \boldsymbol{I}$$
$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{n1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{1m} & \cdots & \boldsymbol{A}_{nm} \end{bmatrix}, \tag{6}$$

where $\boldsymbol{I}$ is an identity matrix 3 by 3. This technique is equivalent to the finite element method interpolating every nodal coordinate by the same shape function assigned to the individual node. Consequently, Vertex Morphing simultaneously controls the shape growth in normal surface direction and the mesh adaptation tangential to the surface without further considerations.

3 Adaptive Vertex Morphing

This section introduces the AVM technique and its properties: computation of radius field, smoothing process, and radius control by the user.

3.1 Radius field computation

The size of the radius plays a crucial role in the filtering process. If the radius does not cover enough elements, the final design may not be smooth and have kinks and poor quality surface mesh. Even if we choose a suitable radius size for the initial model, after n optimization iterations, the model can be deformed dramatically, and the radius size becomes inappropriate. Therefore, it is essential to check if the radius has a proper size during optimization. The adaptive Vertex Morphing computes the minimum required radius for every node during the optimization process. Consequently, it can adapt the radius size for each node to keep the filtering property. The required radius for node k is computed based on the distances to the neighboring nodes j as follows:

$$r_k = C \cdot \max_j(d_j), \tag{7}$$

where $d_j = \left\|\boldsymbol{x}_k - \boldsymbol{x}_j\right\|$ is a distance between node k and j. The constant C can be understood as a number of element layers the filtering includes. From various numerical experiments, the good values for C are in the range [4, 10] (Firl et al. 2012; Hojjat et al. 2014; Stavropoulou et al. 2014).

Firl et al. (2012) show that any filtering introduces the filtering error: the gap between the optimal shape and the "true" optimum. The error disappears, when $r \to 0$. The adaptive Vertex Morphing method allows finding the solution with the smallest possible radius for the given mesh. Hence, it generates the smallest filtering error. Figure 1 shows the computed radii (radius field) by eq. (7) with $C = 7$ as a field, where the value refers to the filter radius at the node.

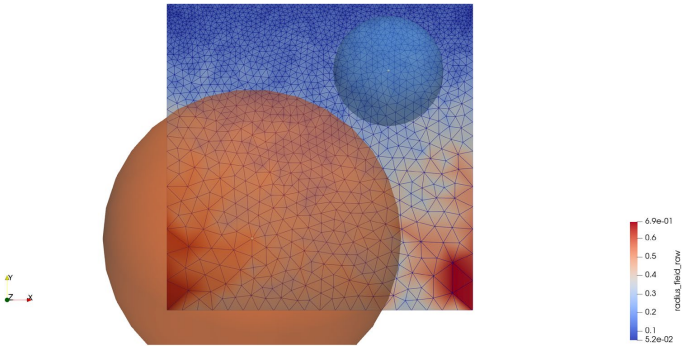


Fig. 1 Individual radii for each node of the mesh, orange—radius of the big element, blue—radius of the small element

3.2 Radius field smoothing process

In the above-computed radius field, the radii size at neighboring nodes varies a lot because the plate has an unstructured mesh. As it is shown in Geiser (2017), the Vertex Morphing generates a gap in the shape update on the border between two different radii. The proposed solution is to have a smooth transition in space from one radius size to another. The adaptive Vertex Morphing method smoothens the computed radius field from eq. 7 so that the radii at the neighboring nodes have similar radii (smooth transition). In this work, an adaptive Vertex Morphing mapping operator $\boldsymbol{A}$ with linear kernel function and local radius size has been applied to smooth the radius field, Algorithm 1.

Algorithm 1 Smoothing process of the radius field

$\bar{\boldsymbol{r}} = \boldsymbol{r}$
for $iter = 1, 2, ...N_{iter}$ **do**
  for $k = 1, 2, ...End_{nodes}$ **do**
    if $r_k < \bar{r}_k$ **then**
      $\bar{r}_k = \bar{r}_k$
    else
      $\bar{r}_k = r_k$
    end if
    Find all neighbor nodes j of node k in the radius $\bar{r}_k/(C-1)$, M - number of neighbors
    $w_k^j = \max(1 - \|x_k - x_j\| / \bar{r}_k, 0) \cdot A^j$, A^j is a nodal area of the node j
    $r_k = (\bar{r}_i + \sum_{j=0}^{M} w_k^j * \bar{r}_j)/(1 + \sum_{j=0}^{M} w_k^j)$
  end for
end for

In the smoothing process, it is required to do multiple filtering iterations because the initial radius field is discrete. The number of smoothing iterations can vary for different examples. Figure 2 compares the raw and smoothed radii for different N_{iter} numbers. The results can be summarized as follows:

1. In case $N_{iter} = 0$, there is no smoothing of the radius field, Fig. 2a, b;
2. In case $N_{iter} = 1$, the radius field is non-smooth in the region with higher radius values, Fig. 2c;
3. In case $N_{iter} = [2, 10]$, the radius field is smooth. If the N_{iter} number increases, the high radius values diffuse into the regions with low radius values, Fig. 2d, e;
4. In case $N_{iter} = 100$, the radius field has very small changes in the values [0.54, 0.6], almost a constant field, Fig. 2f. If $N_{iter} \to \inf$, the radius fields converges to a constant field.

3.3 Local and global radius control

In the initial and adaptive Vertex Morphing methods, the filtering radius size is considered as an additional design parameter that strongly affects the final shape. Minimal required radii computed by the adaptive Vertex Morphing method are not always the best choice. Due to manufacturing limitations, weak performance, or unaesthetic design, one may need to change the radius size to find a new design in the next optimization process. The adaptive Vertex Morphing technique allows setting a global or local minimum radius. Hence, equation 7 is modified as follows:

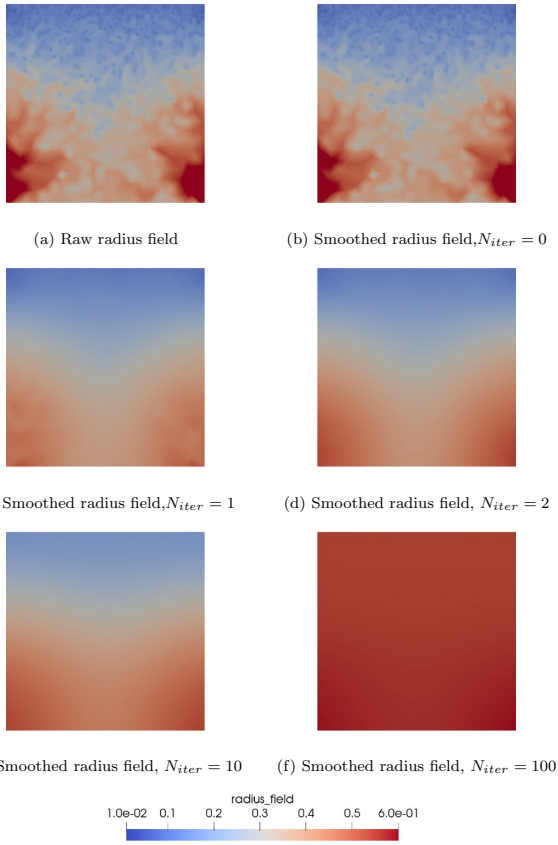
$$r_k = \max(C \cdot \max_j(d_j), r_{min,k}), \tag{8}$$

where the $r_{min,k}$ is a given minimal radius for a node k. Figure 3 shows how the "minimum radius" ($r_{min,k} = 0.3$) modifies the resulting radius field.

The modification in equation (8) extends the design features of the adaptive Vertex Morphing method. On one side, the adaptive Vertex Morphing method is straightforward to use, and on the other side, it is very powerful and flexible parameterization. The workflow with adaptive Vertex Morphing parameterization with a new unknown model is as follows:

1. Run the first cycle of the optimization using only the computed radius field without any additional input for the filter radius.
2. Based on the outcoming results from the first run, adjust the sizes of the filtering radius in the regions where the final shape modes are not suitable or lead to bad performing design.

The adaptive Vertex Morphing method increases the mesh dependency of the parameterization because if the finite element model is discretized with a new mesh, the minimum required radii will also change (see eq. 7). If the optimization problem is convex, the final shape will always converge to an unique solution, independent of the filter radius constant



(a) Raw radius field

(b) Smoothed radius field,$N_{iter} = 0$

(c) Smoothed radius field,$N_{iter} = 1$

(d) Smoothed radius field, $N_{iter} = 2$

(e) Smoothed radius field, $N_{iter} = 10$

(f) Smoothed radius field, $N_{iter} = 100$

Fig. 2 Radius field comparison with respect to N_{iter} smoothing iterations

or variable sizes. In contrast, if the optimization problem is non-convex, the choice of the filter radius will guide the optimizer to the different local optima; hence, the adaptive Vertex Morphing method may find different local minima for different discretizations. The interested reader is referred to Firl et al. (2012), Hojjat et al. (2014) to peruse mesh-independency in FE-based parameterizations.

3.4 Simple 3D plate example

The 3D plate example is prepared to demonstrate the influence of the computed radius field on the design surface's quality. The discrete sensitivity field has been computed and used as a sensitivity field on the plate geometry (Fig. 1). 10 optimization iterations of the steepest descent algorithm with constant step size have been applied to find the deformed plate. The sensitivity field is defined as follows:

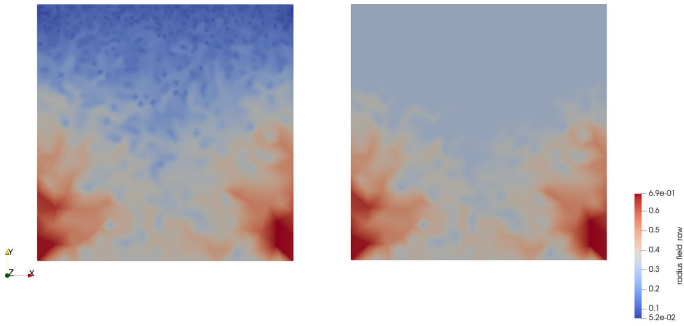
$$\nabla f(x_i) = \boldsymbol{n}_i A_i, \tag{9}$$

where n_i and A_i are the nodal normal and area, respectively. The steepest descent shape update is computed as follows:

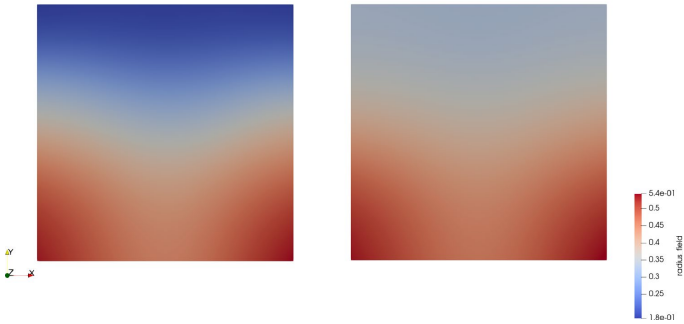
$$\Delta \boldsymbol{x} = \boldsymbol{A}\left(\frac{\alpha}{||\boldsymbol{s}||}\boldsymbol{s}\right), \tag{10}$$

where $\boldsymbol{A}$ is an adaptive Vertex Morphing filtering matrix, which is computed using the smoothed radius field.

In Figs. 4 and 5, the deformed plate is shown with respect to different values of C and N_{iter}. If $N_{iter} = [0, 1]$, the radius field is non-smooth and the deformed plate has



(a) Raw radius field



(b) Smoothed radius field

Fig. 3 Comparison of radius fields: left—original, right—with minimum required radius, ($r_{min,k} = 0.3, N_{iter} = 10, C = 7$)

rough surface with kinks. In contrast, if $N_{iter} \geq 2$, the radius field is smooth, and the deformed plate has a smooth surface. These results correlate with the statements from Geiser (2017). Similarly, if $C < 4$, the filter radius size covers only a few elements; hence the adaptive Vertex Morphing does not compute smooth shape change.

4 Quasi-Newton relaxed gradient projection method

This section introduces the Quasi-Newton Barzilai–Borwein and Quasi-Newton relaxed gradient projection methods and max-value aggregation techniques.

4.1 Relaxed gradient projection method

The relaxed gradient projection method (RGP method) is a modification of a well-known Rosen's gradient projection method (Rosen 1960, 1961). The main idea of the RGP algorithm is to use information regarding the values of the constraints from previous optimization iterations to compute a buffer (critical) zone around the constraint boundary and keep the constraint active if the value is inside the buffer zone. For convenience, the basic formulas are presented below. For more details, the reader should refer to Antonau et al. (2021).

The buffer coefficient $\omega_j^{(i)}$ can be computed based on the constraint value $g_j(\boldsymbol{x}^{(i)})$ and the buffer size $BS_j^{(i)}$:

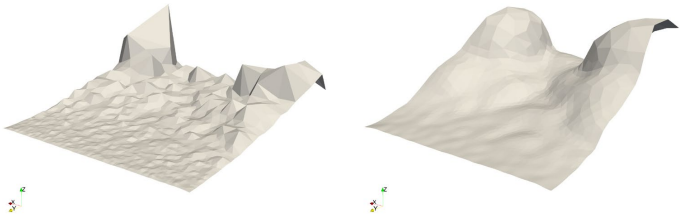
$$\omega_j^{(i)} = \frac{g_j(\boldsymbol{x}^{(i)}) - LBV_j^{(i)}}{BS_j^{(i)}} \qquad LBV_j^{(i)} = CBV_j^{(i)} - BS_j^{(i)} \tag{11}$$

or for equality constraints ($h_j(\boldsymbol{x}_i) = 0$):

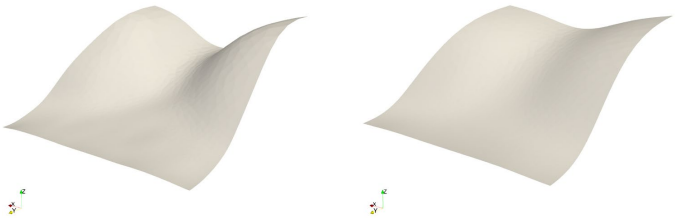
(a) $C = 7,\ N_{iter} = 0$ (b) $C = 7,\ N_{iter} = 1$

(c) $C = 7,\ N_{iter} = 2$ (d) $C = 7,\ N_{iter} = 10$

Fig. 4 Deformed plate with respect to different N_{iter} numbers



(a) $C = 1,\ N_{iter} = 7$ (b) $C = 2,\ N_{iter} = 7$



(c) $C = 4,\ N_{iter} = 7$ (d) $C = 7,\ N_{iter} = 7$

Fig. 5 Deformed plate with respect to different C constant

$$\omega_j^{(i)} = 1 + \frac{abs[g_j(\boldsymbol{x}^{(i)}) - LV_j]}{BS_j^{(i)}}, \tag{12}$$

where $LBV_j^{(i)}$ ("lower buffer value") is a lower boundary of the buffer zone, $BS_j^{(i)}$ ("buffer size") is a size of the buffer zone, $CBV_j^{(i)}$ ("central buffer value") is a value of buffer zone's center, and LV_j is a constraint limit value. The buffer size BS is based on constraints values from previous iterations:

$$\begin{aligned} BS_j^{(i)} &= BSF^{(i)} \cdot \max_k(\Delta g_j(\boldsymbol{x}^{(k)})) \\ \Delta g_j^{(i)} &= abs(g_j(\boldsymbol{x}^{(i)}) - g_j(\boldsymbol{x}^{(i-1)})) \end{aligned}, \tag{13}$$

where $BSF^{(i)}$ can be adjusted by the buffer adaptation functions (Antonau et al. 2021). The buffer coefficient can be separated into two components: "relaxation" and "correction" coefficient. The first part, "relaxation," is calculated as follows:

$$\omega_j^{r,(i)} = \begin{cases} \omega_j^{(i)}, & \text{if } \omega_j^{(i)} \le 1.0 \\ 1, & \text{if } \omega_j^{(i)} > 1.0 \end{cases} \tag{14}$$

If the constraint is equality, the relaxation coefficient is always equal to one, $\omega_j^{r,(i)} = 1.0$. The second component, "correction," $\omega_j^{c,(i)}$ is

$$\omega_j^{c,(i)} = \begin{cases} BSF^{(0)}(\omega_j^{(i)} - 1), & \text{if } 1.0 < \omega_j^{(i)} < \omega^{max} \\ 0, & \text{if } \omega_j^{(i)} \le 1.0 \\ BSF^{(0)}(\omega^{max} - 1), & \text{if } \omega_j^{(i)} \ge \omega^{max} \end{cases}, \tag{15}$$

where the factor $BSF^{(0)} = 2$ is the initial buffer size factor, and ω^{max} is the maximum correction coefficient. If the problem starts from an infeasible domain, the correction coefficient can be very high and may cause numerical issues. The $\omega^{max} = 2$ limits the correction coefficient to the values inside the buffer zone and works in most cases. The search direction can be defined as follows:

$$\begin{aligned} \boldsymbol{p}^{(i)} &= -[\boldsymbol{I} - \boldsymbol{N}\boldsymbol{\omega}^{r,(i)}(\boldsymbol{N}^T\boldsymbol{N})^{-1}\boldsymbol{N}^T]\nabla f^{(i)} \\ \hat{\boldsymbol{s}}^{(i)} &= \boldsymbol{p}^{(i)} - \boldsymbol{N}\boldsymbol{\omega}^{c,(i)} \\ \boldsymbol{s}^{(i)} &= \frac{\hat{\boldsymbol{s}}^{(i)}}{||\hat{\boldsymbol{s}}^{(i)}||_{max}} \end{aligned} \tag{16}$$

The last equation scales the search direction using the max norm and can be skipped if the line search method works with an unscaled search direction. However, max scaling is required in the practical optimization application, where the shape can be changed by a certain amount (constant or limited).

4.2 Barzilai–Borwein method

The Barzilai–Borwein method (BB method) suggests a step size approximation using current and previous sensitivity information. The Barzilai–Borwein method computes a new step size as follows:

$$\alpha^{(i)} = \frac{\boldsymbol{y}^{(i),T}\boldsymbol{d}^{(i-1)}}{\boldsymbol{y}^{(i),T}\boldsymbol{y}^{(i)}} \tag{17}$$

or

$$\alpha^{(i)} = \frac{\boldsymbol{d}^{(i-1),T}\boldsymbol{d}^{(i-1)}}{\boldsymbol{d}^{(i-1),T}\boldsymbol{y}^{(i)}}, \tag{18}$$

where $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i-1)}) - \nabla f(\boldsymbol{x}^{(i)})$ is a change in the sensitivities of the objective function and $\boldsymbol{d}^{(i-1)} = \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i-1)}$ is the previous update of the design variables. Therefore, if $\boldsymbol{s}^{(i)}$ is a search direction at iteration i, the design update is

$$\Delta\boldsymbol{x}^{(i)} = \alpha^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{19}$$

A modification to the original Barzilai–Borwein method is introduced in this work, the Quasi-Newton Barzilai–Borwein method (QN–BB). The main idea of our modification is to compute the step size for each design variable instead of one step size for the full search direction vector. The design update can be found as follows:

$$\boldsymbol{H}^{(i)} = [\alpha_k^{(i)}] \tag{20}$$

$$\alpha_k^{(i)} = \min\left(abs\left[\frac{\boldsymbol{y}_k^{(i),T}\boldsymbol{d}_k^{(i-1)}}{\boldsymbol{y}_k^{(i),T}\boldsymbol{y}_k^{(i)}}\right], \alpha_{k,max}^{(i)}\right) \tag{21}$$

$$\Delta\boldsymbol{x}^{(i)} = \boldsymbol{H}^{(i)} \cdot \boldsymbol{s}^{(i)} \quad , \tag{22}$$

where $\boldsymbol{s}^{(i)}$ is a search direction computed by the relaxed gradient projection method at iteration i and $\alpha_{k,max}^{(i)}$ is a maximum allowed step size at node k. If $\boldsymbol{s}^{(i)}$ is normalized by equation (16), $\alpha_{k,max}^{(i)} = r_k^{(i)}/5$.

4.3 Quasi-Newton relaxed gradient projection method

The Quasi-Newton relaxed gradient projection method combines the Quasi-Newton Barzilai–Borwein method and the relaxed gradient projection method. Linear approximation of the constraints is used to improve the constraint handling. In Fig. 6, the Quasi-Newton relaxed gradient projection method is shown. The QN–BB–RGP method is performed as follows:

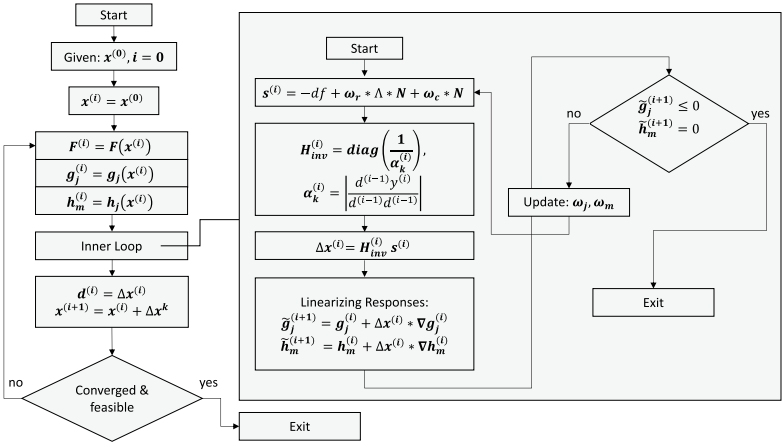


Fig. 6 Flowchart of the Quasi-Newton relaxed gradient projection method

1. Compute response values at the current design state: $f(x^{(i)})$, $g(x^{(i)})$;
2. Compute gradients of the objective function and active constraints: $\nabla \boldsymbol{f}(x^{(i)})$, $\nabla \boldsymbol{g}(x^{(i)})$;
3. Find shape update $\boldsymbol{\Delta x}^{(i)}$:
   (a) Compute search direction $\boldsymbol{s}^{(i)}$, eq. (16);
   (b) Compute shape update $\boldsymbol{\Delta x}^{(i)}$, eq. (20);
   (c) Compute linear approximation to response function for the computed shape update: $\tilde{g}(\boldsymbol{x}^{(i+1)})$, $\tilde{h}(\boldsymbol{x}^{(i+1)})$;
   (d) If $\tilde{g}(\boldsymbol{x}^{(i+1)}) <= 0$ and $\tilde{h}(\boldsymbol{x}^{(i+1)}) = 0$, then the feasible shape update is found. The inner loop is converged;
   (e) If $\tilde{g}(\boldsymbol{x}^{(i+1)}) > 0$ and $\tilde{h}(\boldsymbol{x}^{(i+1)})! = 0$, then the feasible shape update is not found. Update the buffer coefficients $\omega_j^{(i)} += 0.02$ and repeat the inner loop process;
4. Save current $\boldsymbol{\Delta x}^{(i)}$, $\boldsymbol{s}^{(i)}$;
5. Check if the optimization algorithm has converged. If not, go to Step 1.

The constant to increase the $\omega_j^{(i)}$ is based on the numerical experiments, and it shows a good compromise between accuracy and cost. It has no effect on $\omega_j^{(i+1)}$.

4.4 Maximum-value constraint aggregation technique

The RGP and QN–BB–RGP methods compute the buffer coefficient based on the constraint value and buffer size using equation (11). For the nodal constraints, the buffer coefficients are computed for each node. For constraint $g_j(x_k^{(i)})$ of node k at the optimization iteration i, the buffer coefficient is computed as follows:

$$w_{j,k}^{(i)} = \frac{g_j(x_k^{(i)}) - LBV_j^{(i)}}{BS_j^{(i)}} \tag{23}$$

If $\nabla g_j(x_k^{(i)})$ is a gradient vector of the constraint for node k, the aggregated constraint can be formulated as follows:

$$\begin{aligned} g_j(x^{(i)}) &= \max_k(g_j(x_k^{(i)})) \\ \nabla g(x^{(i)}) &= \sum_k w_{j,k}^{(i)} \cdot \nabla g_j(x_k^{(i)}). \\ w_j^{(i)} &= \max_k(w_{j,k}^{(i)}) \end{aligned} \tag{24}$$

Linear approximation to a constraint function at the new design point $x_k^{(i+1)}$ is

$$\tilde{g}(x_k^{(i+1)}) = g(x_k^{(i)}) + \nabla g_j(x_k^{(i)} \boldsymbol{\Delta x}^{(i)}). \tag{25}$$

If the approximated value $\tilde{g}(x_k^{(i+1)}) > 0$, the QN–BB–RGP method increases the $w_{j,k}^{(i)}$ to modify the computed search direction $\boldsymbol{s}^{(i)}$, equation (16).

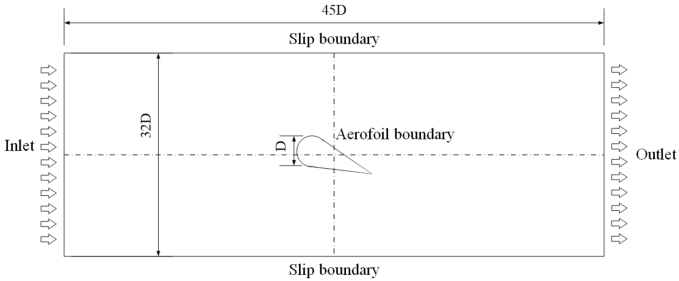


Fig. 7 Aerofoil problem configuration used in RANS in 2D

5 Academic experiment

This numerical investigation is designed to illustrate the applicability of the methods above in a highly nonlinear shape optimization problem. Reynolds-averaged Navier–Stokes (RANS) equations are used to solve for flow variables in the fluid domain utilizing the $k-\omega-sst$ two-equation turbulence model. The reader is referred to Warnakulasuriya (2021) for more information on the specific implementations of the two-equations turbulence model in a Finite Element (FE) context.

5.1 Experimental set-up

The fluid domain $\Omega = (-24.5D, 24.5D) \times (-16D, 16D) \subset R^2$ chosen after a domain size study is illustrated in Fig. 7, where $D = 0.1\,m$. The inlet (i.e., Γ_{inlet}) is applied with a constant velocity (i.e., u_{inlet}), and turbulence quantities are determined using a turbulence intensity of 5% and a turbulent mixing length of $45D$. The Reynolds number is $Re = 10e5$. The outlet (i.e., Γ_{outlet}) is applied with a $0\,Pa$ Dirichlet boundary condition for P, zero gradient boundary conditions for variables u, k, ϵ, ω. The slip condition (i.e., Γ_{far}) is applied on the top and bottom slip boundary for variable u, and all other variables are applied with a zero gradient boundary condition. Linear-log law wall functions developed by Launder and Spalding (1983) are used on the aerofoil boundary (i.e., Γ_s) to accommodate a wide range of meshes with $y^+ \in [0, 300]$ in the first element near the wall boundary.

Figure 8 illustrates the overall mesh (refer to Fig. 8a) and enlarged view of the same mesh near the initial aerofoil geometry (refer fig. 8b) consisting of 20, 183 triangle elements.

5.2 Optimization procedure

Drag and lift forces are the interested scalar QOI in this numerical experiment because the aerofoil's usefulness depends on having maximum lift with minimum drag force. Equation (26) describes the optimization problem of interest.

$$\min_{s} \quad J_{drag}(\boldsymbol{w}(\boldsymbol{s}), \boldsymbol{s})$$

subjected to

$$\begin{aligned} &\boldsymbol{R}_{\phi} = \boldsymbol{0} \quad \forall \phi \in \{u, v, w, p, k, \omega\} \\ &G_{centroid}(\boldsymbol{s}) = 0 \\ &J_{lift}(\boldsymbol{w}(\boldsymbol{s}), \boldsymbol{s}) - J_{lift}\left(\boldsymbol{w}(\boldsymbol{s}_{initial}), \boldsymbol{s}_{initial}\right) \geq 0.0 \end{aligned} \quad , \tag{26}$$

where $J_{lift}\left(\boldsymbol{w}(\boldsymbol{s}_{initial}), \boldsymbol{s}_{initial}\right) = 0.89$ in all experiments.

$G_{centroid}$ is computed by averaging all nodal coordinates along the aerofoil boundary as illustrated in equation (27) where N represents the number of nodes in Γ_s. It is applied to constrain aerofoil geometry to be present at the center of Ω for all design iterations. AVM is used to smooth the noisy sensitivity field on the aerofoil boundary, and QN–BB–RGP

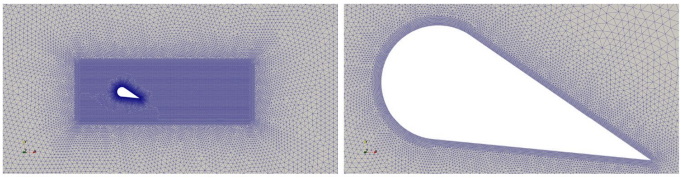


Fig. 8 Initial mesh for a 2D aerofoil optimization problem

(a) Initial overall mesh (b) Initial enlarged mesh

is used to obtain the next aerofoil boundary for the optimization problem

$$G_{centroid} = \left\| \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_{current,i} - \boldsymbol{x}_{\Omega_{center}}) \right\|_2^2, \quad (27)$$

where $\boldsymbol{x}_{\Omega_{center}}$ is the center of Ω.

5.3 Results

The results of the academic experiment are presented hereafter. The academic experiment is carried out with adaptive Vertex Morphing (AVM) and different radius set-ups: adaptive radius ($C = 7$), 20 *mm* and 50 *mm*. The QN–BB–RGP algorithm is used in all experiments to solve the optimization problems. The focus of this experiment is to show the importance of the filtering radius. All experiments have done 500 optimization iterations without further convergence criteria.

Figure 9 illustrates drag force variation with each design iteration. The experiment with constant Vertex Morphing radii of 50 *mm* shows oscillations and does not depict an overall improvement in the drag force reduction. However, the experiments with adaptive and 20 *mm* radii show improvement over the design iterations where 20 *mm* radius set-up finds the best performing design.

Geometric constraint variations with respect to optimization iterations are illustrated in Fig. 10. It depicts that all the proposed designs satisfy the geometric constraint, which enforces keeping the aerofoil design in the center of the fluid domain.

However, the lift constraint as depicted in Fig. 11 shows oscillating behavior, thus with constraint violations. The QN–BB–RGP method cannot precisely predict the constraint value of the non-linear constraints such as lift by using a linear approximation. However, the QN–BB–RGP can correct the violated constraint values in all experiments. Hence, there are improved feasible designs during the optimization process. Table 1 summarizes the results of the experiments and shows the performance of the last and the best-feasible designs. Table 2 summarizes the computational time.

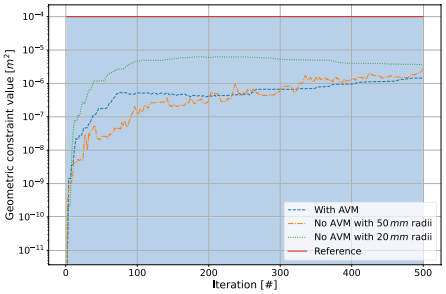


Fig. 10 Geometric constraint variation with optimization design iterations

To further investigate the final design from each experiment, Fig. 12 illustrates the velocity and pressure distributions of the optimized designs. It can be observed that the experiments with adaptive and constant Vertex Morphing radii of 20 *mm* try to reduce the frontal area to reduce drag force acting upon the aerofoil. The optimized design obtained by the experiment with radii of 50 *mm* does not significantly reduce the frontal area, and it is due to restricting sensitivity information by having higher constant Vertex Morphing radii. In all experiments, the final designs have smooth boundaries. In the case of adaptive radius, the final design has smaller local changes on the lower surface and the trailing edge, see Fig. 12.

Figure 13 illustrates the effect of the filtering radii on the generated shape update. At iteration 1, in all experiments,

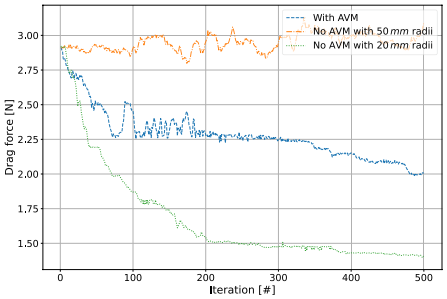


Fig. 9 Drag force variation with optimization design iterations

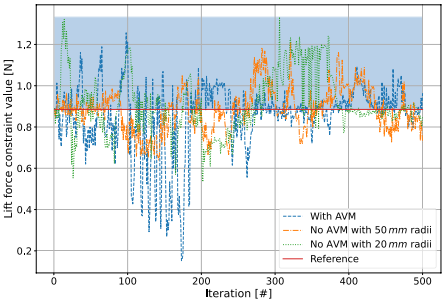


Fig. 11 Lift constraint variation with optimization design iterations

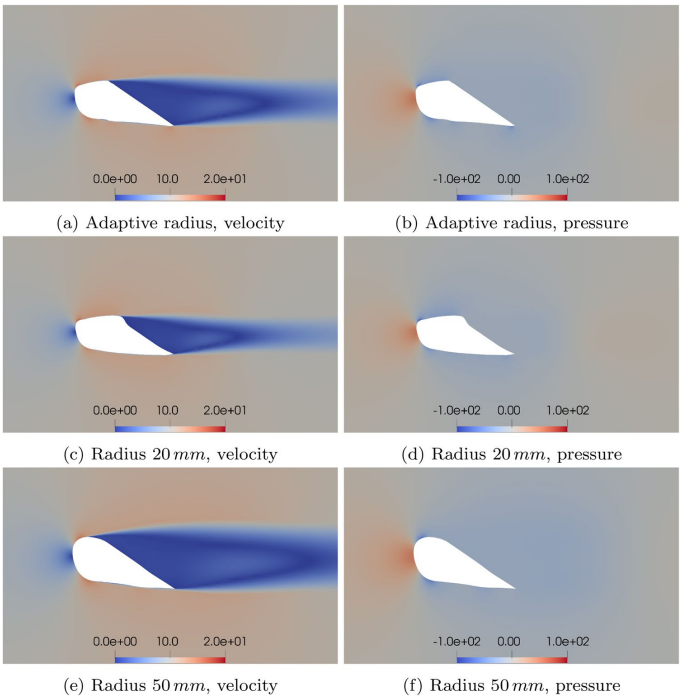
Table 1 Optimization results of the aerofoil, in red—best-feasible designs

Radius setting	Drag force	Improvement [%]	Lift force	Violation [%]	Iteration
Adaptive radius	2.007	−31.1%	+0.964	+8.31%	500
Radius 20*mm*	1.407	−51.87%	0.849	−4.61%	500
Radius 20*mm*	1.416	−51.55%	0.894	+0.45%	491
Radius 50*mm*	3.09	+6.16%	0.726	−18.43%	500
Radius 50*mm*	2.809	−3.53%	0.982	+10.34%	176

Table 2 Computation time

Computation time	
Aver. primal analysis	792.6 s
Aver adjoint analysis per response	30.3 s
Aver. time per optimization iteration	827 s
Aver. time to find $\Delta \boldsymbol{x}$	4.1 s
Overall optimization time	413526 s
CPU hours	1380

the raw drag sensitivities are identical, as well as the steepest descent step ($\Delta \boldsymbol{x}^{(1)} = -\alpha^{(1)} \cdot \nabla \boldsymbol{f}^{(1)}$). It can be observed that the radius of 20 *mm* smoothens the non-accurate sensitivity on the trailing edge (flow separation point) better than a smaller (adaptive) radius (7 *mm* at the point). Therefore, the large filter helps to reduce the local error of the sensitivities and generates shape update changes that modify the aerofoil profile more "globally." As a result, the optimizer finds a better-performing design with a 20 *mm* radius. In contrast,

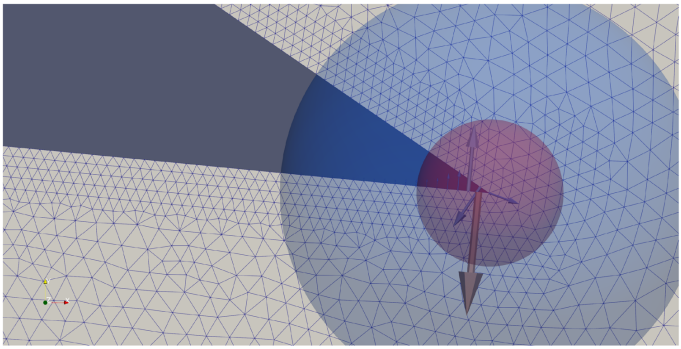


(a) Adaptive radius, velocity (b) Adaptive radius, pressure

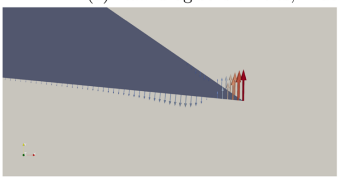
(c) Radius 20 mm, velocity (d) Radius 20 mm, pressure

(e) Radius 50 mm, velocity (f) Radius 50 mm, pressure

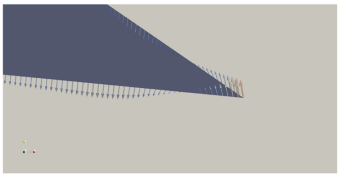
Fig. 12 Optimized aerofoil design, on left—velocity $[ms^{-1}]$ and on right—pressure $[Pa]$ distributions



(a) Raw drag sensitivities, Blue - 20 mm, Red - adaptive, 7 mm



(b) Shape update (scaled), adaptive radius 7 mm



(c) Shape update (scaled), constant radius 20 mm

Fig. 13 Drag sensitivities and shape updates at the wing's tip, iteration 1

the optimizer finds a weak design with a radius of 50 mm due to high filtering error. As it is discussed in Firl et al. (2012), the filtering intensity is always a compromise between filtering error and smoothing of sensitivity error.

6 Large industrial example

An industrial CFD optimization problem is solved to demonstrate the full potential and flexibility of the adaptive Vertex Morphing technique and the robustness of the QN–BB–RGP algorithm. The goal of the optimization is to reduce the drag force of the BMW M4 GT4 car, while the downforce has to be equal to or larger than an initial value. The example has been prepared in cooperation with BMW Group, Motorsport division. All methods above are implemented in the optimization framework "ShapeModule" (BMW Group). Siemens STAR-CCM+™ software (Version 2020.2) is used to do primal and sensitivity analysis of the numerical model.

6.1 Problem description

Table 3 summarizes the properties of the CFD model/analysis, optimization problem, and parameterization. All geometrical constraints are aggregated into one, as shown in Section 4.4. The details of the turbulence models, solvers,

Table 3 Optimization problem description

Primal & Adjoint Analysis	
Time integration	Steady
Turbulence model	K-Omega SST (Menter)
Adjoint Model	Adjoint Frozen Turbulence
Mesh & Domain	
Domain size	$80 \times 40 \times 30$ [m]
Number of cells	114,681,935
Smallest element size	4 [mm]
Element type	hex dominant
Optimization Problem	
Design variables	position of the surface nodes: $[x, y, z]$
Number of design Variables	1,950,141
Objective function	drag force
Physical constraint type	downforce
Number of physical constraints	1
Geometrical constraint type	max shape update at node
Number of geometrical constraints	115,830
Parameterization	
Part's name	Radius size [mm]
Overall Design Surface	30
Rear wing	50
Trunk lid	200

and adjoint analysis can be found in the documentation of the Siemens STAR-CCM+^{TM}.

Figure 14 shows the CFD model of the full car, where the design surface is highlighted in blue. The car's exterior surface is chosen as a design surface: front flaps, front splitter, and rear wing. All these features have different physical and mesh sizes. Hence, they require different radius sizes. Figure 15 shows the radius field over the design surface and Table 3 gives the radius sizes.

The CFD model is highly detailed, and it contains a lot of non-design parts on the exterior surface and inside the car as well. For instance, non-design parts are the door handles, radiator, mirrors, lights, engine, suspension, gearbox, etc. Therefore, the geometrical constraints are needed to avoid the penetration between design and non-design parts. Figure 16 shows the design surface, where the geometrical constraints are applied.

6.2 Results and discussions

The QN–BB–RGP method solves this problem successfully without any parameter tuning of the optimization algorithm. It is an important point because finding suitable parameters is a very expensive and time-consuming process. The simulation is run on a 12-node HPC cluster, where each node has 2 x AMD 24-Core EPYC 7402 with 252 GB RAM. In total, simulation takes 210 hours or 120960 CPUh. Figure 17 shows the response values change during the optimization process. The optimization process is stopped by a maximum number of iterations due to the time limit. From the results given in Table 4, the most consuming operation is primal and adjoint CFD analysis. Finding the search direction takes around 1 hour, which is less than other operations: mesh motion, file saving, etc. One inner iteration takes approximately 3 min, where computing the search direction takes

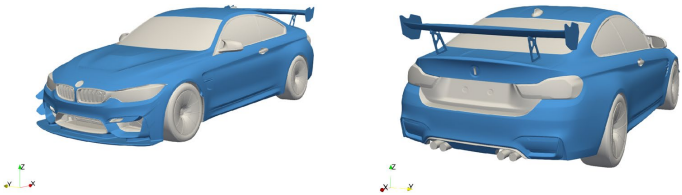


Fig. 14 Geometry of the CFD model: design surface—blue, non-design parts—light gray

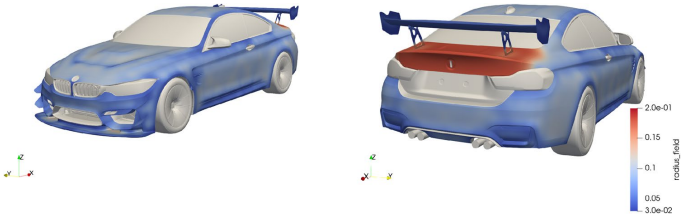


Fig. 15 Radius field over the design surface

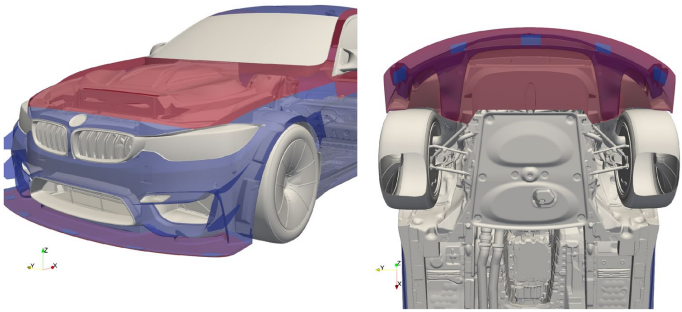


Fig. 16 Geometrical constraint: constrained zone—red, free design surface—blue, non-design parts—light gray

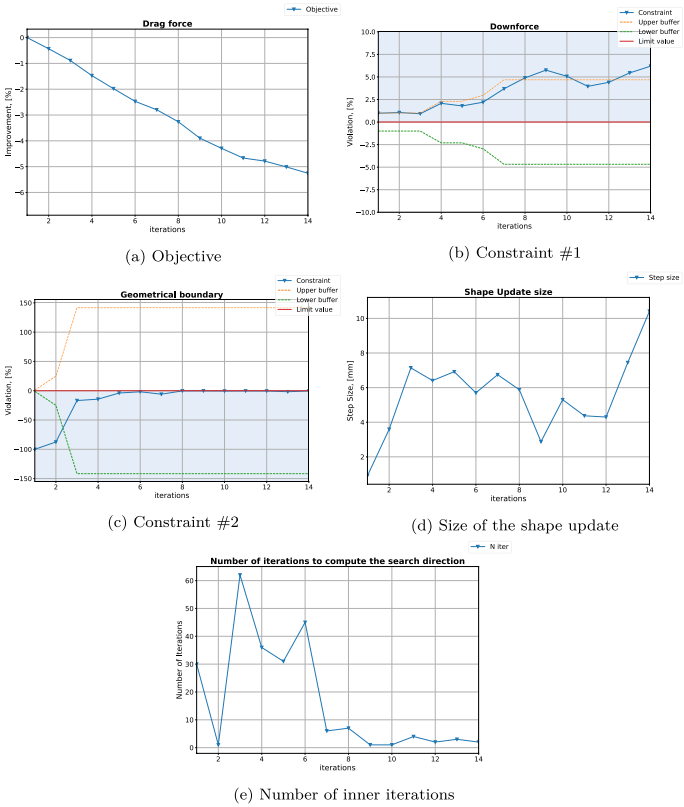


Fig. 17 Optimization results: response function evaluations & size of shape updates

(a) Objective

(b) Constraint #1

(c) Constraint #2

(d) Size of the shape update

(e) Number of inner iterations

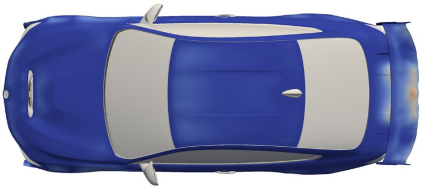
Table 4 Computation time [hour]

Computation time [hour]	
Aver. primal analysis	10
Aver adjoint analysis per response	2.5
Aver. time per optimization iteration	14.95
Aver. time to find Δx	0.85
Other operations	1.6
Overall optimization time	210
CPU hours	120960

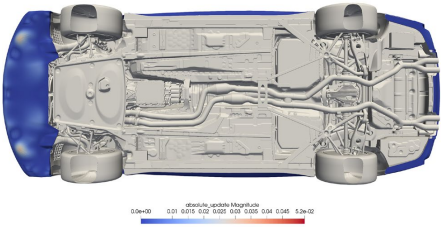
around 0.2 s, the shape update – 4.3 s, and mapping the shape update takes the rest. Figure 17e shows the number of iterations required to converge the inner loop. The most needed number of iterations is at optimization iterations 3 when the geometric constraint gets active.

Figure 17b shows that the lift constraint is not active at the final design. However, if the downforce constraint is not included, the optimizer chooses another local optimum and dramatically reduces the downforce. Figure 17c shows the performance of the optimization algorithm to hold the geometric constraint. As described in Section 4.3 and 4.4, the geometric constraints are aggregated and accurately handled during optimization. The design stays just in 0.04 mm distance from the limit value (limit value: 5 mm, max design value: 4.96 mm), while the design update in the whole model is up to 10 mm (see Fig. 19d).

The optimization results are shown in Figs. 18 and 19. One can see that the most shape changes happened on the rear wing, trunk lid, front flaps, and front splitter. The middle section of the rear wing, trunk lid, and frontal flaps has been deformed to generate less drag. In contrast, side sections of the rear wing and front splitter generate more lift,



(a) Upper view



(b) Lower view

Fig. 18 Optimization results: shape change on top & bottom view

this explains why our final design has improved both aerodynamic characteristics. Suppose the design surface is not so large and includes only the rear wing and front splitter, in that case, the final results will not be so impressive, just 4% drag reduction (the results have been shown at ISSMO-14th World Congress of Structural and Multidisciplinary Optimization, "Quasi-Newton Relaxed Gradient Projection method in large constrained node-based shape optimization problems").

In Figure 19d, one can see the difference on the final shape update with different radii. The rear wing has a radius of 5 mm, while the trunk lid is – 20 mm. The final shape of the trunk lid is very smooth, and the wavelength of shape change is large. In contrast, the shape change of the rear wing is locally detailed, but it is smooth. Due to manufacturing limitations or unaesthetic design, one can change the radii to find a new design in the next optimization process.

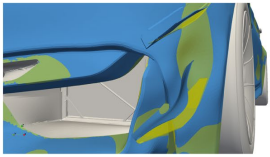
7 Conclusions

Adaptive Vertex Morphing extends Vertex Morphing parameterization to solve large-scale optimization problems with local control on the final shape. It allows reasonable control of the final form and finds various solutions. The QN–BB–RGP method shows good potential as an acceleration and stabilization technique for gradient descent methods with many design variables and expensive response functions. The proposed QN–BB–RGP method, in combination with the maximum value aggregation technique, solves large optimization problems with a huge amount of geometric constraints. Linear approximation is not accurate for highly non-linear constraints, but still, the method can handle such constraints and find feasible solutions. In future work, modifications of the Barzilai–Borwein method can be studied to improve the QN–BB method, and adaptive Vertex Morphing can be extended to solve multi-disciplinary optimization problems with non-matching meshes.

Acknowledgements The authors wish to thank the ShapeModule project (BMW Group, München) for the support. Thanks are also due to Mr. Vignesh Manickavasagam Manian (BMW Group) for technical support. The authors acknowledge the support of colleagues Dr.-Ing. Reza Najian Asl and Mr. Armin Geiser for their valuable insights.

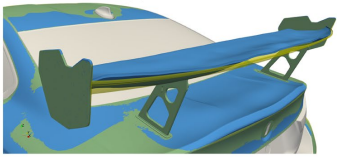
Fig. 19 Optimization results: optimized design—blue, initial design—transparent yellow, non-design parts—light gray

(a) Front view



(b) Front flaps

(c) Rear view



(d) Rear wing

Funding Open Access funding enabled and organized by Projekt DEAL. This paper is partly based on research sponsored by the BMW group.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Replication of the results The proposed methods are implemented in the optimization framework "ShapeModule" (BMW Group, shapemodule@bmw.de) (no public access) and in the open-source package Kratos Multiphysics (Dadvand et al. 2010). An interested reader can try the Vertex Morphing implementation in the shape optimization application of Kratos Multiphysics software.

References

Agarwal D, Robinson T, Armstrong C, Kapellos C (2018) Enhancing cad-based shape optimisation by automatically updating the cad model's parameterisation. Struct Multidisc Optim 59:11. https://doi.org/10.1007/s00158-018-2152-7

Ahookhosh M, Ghaderi S (2017) On efficiency of nonmonotone armijo-type line searches. Appl Math Model 43:170–190. https://doi.org/10.1016/j.apm.2016.10.055

Antonau I, Hojjat M, Bletzinger K-U (2021) Relaxed gradient projection algorithm for constrained node-based shape optimization. Struct Multidisc Optim 64(4):1633–1651, 04 . https://doi.org/10.1007/s00158-020-02821-y

Barzilai J, Borwein J. M (1988) Two-point step size gradient methods. IMA J Numer Anal 8(1):141–148, 01. ISSN 0272-4979. https://doi.org/10.1093/imanum/8.1.141

Baumgärtner D (2020) On the grid-based shape optimization of structures with internal flow and the feedback of shape changes into a CAD model. Dissertation, Technische Universität München, München

Baumgärtner D, Viti A, Dumont A, Carrier G, Bletzinger K-U (2016) Comparison and combination of experience-based parametrization with vertex morphing in aerodynamic shape optimization of a forward-swept wing aircraft 06. https://doi.org/10.2514/6.2016-3368

Bletzinger K-U (2014) A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. Struct Multidisc Optim 49(6):873–895. https://doi.org/10.1007/s00158-013-1031-5

Bletzinger K.-U (2017) Shape optimization, pp 1–42. ISBN 9781119176817. https://doi.org/10.1002/9781119176817.ecm2109

Chen L (2021) Gradient descent akin method. Dissertation, Technische Universität München, München,

Chen L, Bletzinger K-U, Bletzinger A, Wüchner R (2019) A modified search direction method for inequality constrained optimization problems using the singular-value decomposition of normalized response gradients. Struct Multidisc Optim. https://doi.org/10.1007/s00158-019-02320-9

Dadvand P, Rossi R, Oñate E (2010) An object-oriented environment for developing finite element codes for multi-disciplinary applications. Arch Comput Methods Eng 17(3):253–297. https://doi.org/10.1007/s11831-010-9045-2

Dai Y-H, Fletcher R (2005) Projected barzilai-borwein methods for large-scale box-constrained quadratic programming. Numer Math 100(1):21–47. https://doi.org/10.1007/s00211-004-0569-y

Ertl F-J (2020) Vertex Morphing for constrained shape optimization of three-dimensional solid structures. Dissertation, Technische Universität München, München,

Firl M, Bletzinger K-U (2012) Shape optimization of thin walled structures governed by geometrically nonlinear mechanics. Comput Methods Appl Mech Eng 107–117(09):237–240. https://doi.org/10.1016/j.cma.2012.05.016

Firl M, Wüchner R, Bletzinger K-U (2012) Regularization of shape optimization problems using FE-based parametrization. Struct Multidisc Optim 47(4):507–521. https://doi.org/10.1007/s00158-012-0843-z

L. A. G, Guillaume P (2018) Soft handle triggering: A cad-free parameterization tool for adjoint-based optimization methods. https://doi.org/10.5281/ZENODO.1887986

Geiser A, Antonau I, Bletzinger K.-U (2021) AGGREGATED FORMULATION OF GEOMETRIC CONSTRAINTS FOR NODE-BASED SHAPE OPTIMIZATION WITH VERTEX MORPHING. In: 14th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control. Institute of Structural Analysis and Antiseismic Research National Technical University of Athens. https://doi.org/10.7712/140121.7952.18383

Geiser A B. K.-U, Wüchner R (2017) Variable filter radii for vertex morphing based design of adaptive structures. In: Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia and Industry

Ghantasala A, Asl RN, Geiser A, Brodie A, Papoutsis E, Bletzinger K-U (2021) Realization of a framework for simulation-based large-scale shape optimization using vertex morphing. J Optim Theory Appl 189(1):164–189. https://doi.org/10.1007/s10957-021-01826-x

Hojjat M, Stavropoulou E, Bletzinger K.-U (2014) The vertex morphing method for node-based shape optimization. Comput Methods Appl Mech Eng 268:494–513, 01 https://doi.org/10.1016/j.cma.2013.10.015

Huang Y, Dai Y-H, Liu X-W, Zhang H (2022) On the acceleration of the barzilai-borwein method. Comput Optim Appl 81(3):717–740. https://doi.org/10.1007/s10589-022-00349-z

Jameson A (1988) Aerodynamic design via control theory. J Sci Comput 3(3):233–260. https://doi.org/10.1007/bf01061285

Jameson A (1995) Optimum aerodynamic design using CFD and control theory. In: 12th Computational Fluid Dynamics Conference. American Institute of Aeronautics and Astronautics . https://doi.org/10.2514/6.1995-1729

Kröger J, Rung T (2015) CAD-free hydrodynamic optimisation using consistent kernel-based sensitivity filtering. Ship Technol Res 62(3):111–130. https://doi.org/10.1080/09377255.2015.1109872

Launder B. E, Spalding D. B (1983) The numerical computation of turbulent flows, pp 96–116, https://doi.org/10.1016/B978-0-08-030937-8.50016-7

Müller PM, Kühl N, Siebenborn M, Deckelnick K, Hinze M, Rung T (2021) A novel p-harmonic descent approach applied to fluid

dynamic shape optimization. Struct Multidisc Optim 64(6):3489–3503. https://doi.org/10.1007/s00158-021-03030-x

Najian Asl R (2019) Shape optimization and sensitivity analysis of fluids, structures, and their interaction using Vertex Morphing parametrization. Dissertation, Technische Universität München, München,

Najian Asl R, Shayegan S, Geiser A, Hojjat M, Bletzinger K.-U (2017) A consistent formulation for imposing packaging constraints in shape optimization using vertex morphing parametrization. *Structural and Multidisciplinary Optimization*, 56: 1–13, 10 . https://doi.org/10.1007/s00158-017-1819-9

Oleg Burdakov N. H, Dai Yu-Hong (2019) Stabilized barzilai-borwein method. J Comput Math 37(6): 916–936. https://doi.org/10.4208/jcm.1911-m2019-0171

Papoutsis-Kiachagias EM, Giannakoglou KC (2014) Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. Arch Comput Methods Eng 23(2):255–299. https://doi.org/10.1007/s11831-014-9141-9

Pironneau O (1974) On optimum design in fluid mechanics. J Fluid Mech 64(1):97–110. https://doi.org/10.1017/s0022112074002023

Rosen J (1960) The gradient projection method for nonlinear programming. part i. linear constraints. J Soci Ind Appl Math 8:03. https://doi.org/10.1137/0108011

Rosen J (1961) The gradient projection method for nonlinear programming: Part ii. Siam J Appl Math 9:01

Stavropoulou E, Hojjat M, Bletzinger K-U (2014) In-plane mesh regularization for node-based shape optimization problems. Comput Methods Appl Mech Eng 275:39–54. https://doi.org/10.1016/j.cma.2014.02.013

Stück A, Rung T (2011) Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation. Comput Fluids 47(1):22–32. https://doi.org/10.1016/j.compfluid.2011.01.041

Stück A, Rung T (2013) Adjoint complement to viscous finite-volume pressure-correction methods. J Comput Phys 248:402–419. https://doi.org/10.1016/j.jcp.2013.01.002

Warnakulasuriya S (2021) Development of Finite Element Based Sensitivity Analysis and Goal Oriented Mesh Refinement Using Adjoint Approach for Steady and Transient Flows. Dissertation, Technische Universität München, München,

Xu S, Jahn W, Mueller J-D (2014) Cad-based shape optimisation with cfd using a discrete adjoint. Int J Numer Methods Fluids 74:01. https://doi.org/10.1002/fld.3844

Zhou B, Gao L, Dai Y-H (2006) Gradient methods with adaptive step-sizes. Comput Optim Appl 35(1):69–86. https://doi.org/10.1007/s10589-006-6446-0

List of Figures

List of Tables

Bibliography

[1] M. Ahookhosh and S. Ghaderi. "On efficiency of nonmonotone Armijo-type line searches". In: *Applied Mathematical Modelling* 43 (Mar. 2017), pp. 170–190. DOI: 10.1016/j.apm.2016.10.055.

[2] G. Allaire, F. Jouve, and G. Michailidis. "Thickness control in structural optimization via a level set method". In: *Structural and Multidisciplinary Optimization* 53.6 (Apr. 2016), pp. 1349–1382. DOI: 10.1007/s00158-016-1453-y.

[3] I. Antonau, M. Hojjat, and K.-U. Bletzinger. "Relaxed gradient projection algorithm for constrained node-based shape optimization". In: *Structural and Multidisciplinary Optimization* 64.4 (Apr. 2021), pp. 1633–1651. DOI: 10.1007/s00158-020-02821-y.

[4] R. Bartz, T. Franke, S. Fiebig, and T. Vietor. "Density-based shape optimization of 3D structures with mean curvature constraints". In: *Structural and Multidisciplinary Optimization* 65.1 (Dec. 2021). DOI: 10.1007/s00158-021-03089-6.

[5] J. Barzilai and J. M. Borwein. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (Jan. 1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141. eprint: http://oup.prod.sis.lan/imajna/article-pdf/8/1/141/2402762/8-1-141.pdf.

[6] D. Baumgärtner. "On the grid-based shape optimization of structures with internal flow and the feedback of shape changes into a CAD model". Dissertation. München: Technische Universität München, 2020.

[7] D. Baumgärtner, A. Viti, A. Dumont, G. Carrier, and K.-U. Bletzinger. "Comparison and combination of experience-based parametrization with Vertex Morphing in aerodynamic shape optimization of a forward-swept wing aircraft". In: June 2016. DOI: 10.2514/6.2016-3368.

[8] K.-U. Bletzinger. "A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape". In: *Structural and Multidisciplinary Optimization* 49.6 (Jan. 2014), pp. 873–895. DOI: 10.1007/s00158-013-1031-5.

[9] K.-U. Bletzinger. "Shape Optimization". In: *Encyclopedia of Computational Mechanics Second Edition*. 2017, pp. 1–42. ISBN: 9781119176817. DOI: 10.1002/9781119176817.ecm2109.

[10] B. J. Brelje, J. L. Anibal, A. Yildirim, C. A. Mader, and J. R. R. A. Martins. "Flexible Formulation of Spatial Integration Constraints in Aerodynamic Shape Optimization". In: *AIAA Journal* 58.6 (June 2020), pp. 2571–2580. DOI: 10.2514/1.j058366.

[11] C. G. Broyden. "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations". In: *IMA Journal of Applied Mathematics* 6.1 (1970), pp. 76–90. DOI: 10.1093/imamat/6.1.76.

[12] A. Cauchy. "Méthode générale pour la résolution des systéms d'équations simultanées". In: *Comp. Rend. Sci. Paris* 25 (1847), pp. 536–538.

[13] L. Chen. "Gradient descent akin method". Dissertation. München: Technische Universität München, 2021.

[14] P. Dadvand, R. Rossi, and E. Oñate. "An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications". In: *Archives of Computational Methods in Engineering* 17.3 (July 2010), pp. 253–297. DOI: 10.1007/s11831-010-9045-2.

[15] Y.-H. Dai and R. Fletcher. "Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming". In: *Numerische Mathematik* 100.1 (Feb. 2005), pp. 21–47. DOI: 10.1007/s00211-004-0569-y.

[16] M. G. Damigos, E. M. Papoutsis-Kiachagias, and K. C. Giannakoglou. "Adjoint variable-based shape optimization with bounding surface constraints". In: *International Journal for Numerical Methods in Fluids* 93.3 (Sept. 2020), pp. 590–609. DOI: 10.1002/fld.4900.

[17] F.-J. Ertl. "Vertex Morphing for constrained shape optimization of three-dimensional solid structures". Dissertation. München: Technische Universität München, 2020.

[18] V. M. Ferrándiz et al. *KratosMultiphysics/Kratos: Release 9.2*. en. 2022. DOI: 10.5281/ZENODO.3234644.

[19] M. Firl, R. Wüchner, and K.-U. Bletzinger. "Regularization of shape optimization problems using FE-based parametrization". In: *Structural and Multidisciplinary Optimization* 47.4 (Sept. 2012), pp. 507–521. DOI: 10.1007/s00158-012-0843-z.

[20] R. Fletcher. "General Linearly Constrained Optimization". In: *Practical Methods of Optimization*. John Wiley & Sons, Ltd, 2013. Chap. 11, pp. 259–276. ISBN: 9781118723203. DOI: 10.1002/9781118723203.ch11.

[21] R. Fletcher. "A new approach to variable metric algorithms". In: *The Computer Journal* 13.3 (Mar. 1970), pp. 317–322. DOI: 10.1093/comjnl/13.3.317.

[22] R. Fletcher and C. M. Reeves. “Function minimization by conjugate gradients”. In: *Comput. J.* 7 (1964), pp. 149–154.

[23] R. Fletcher. “On the Barzilai-Borwein Method”. In: *Optimization and Control with Applications*. Ed. by L. Qi, K. Teo, and X. Yang. Boston, MA: Springer US, 2005, pp. 235–256. ISBN: 978-0-387-24255-2.

[24] A. Garaigordobil, R. Ansola, J. Santamarıéa, and I. F. de Bustos. “A new overhang constraint for topology optimization of self-supporting structures in additive manufacturing”. In: *Structural and Multidisciplinary Optimization* 58.5 (May 2018), pp. 2003–2017. DOI: 10.1007/s00158-018-2010-7.

[25] A. Geiser, I. Antonau, and K.-U. Bletzinger. “AGGREGATED FORMULATION OF GEOMETRIC CONSTRAINTS FOR NODE-BASED SHAPE OPTIMIZATION WITH VERTEX MORPHING”. In: *14th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control*. Institute of Structural Analysis and Antiseismic Research National Technical University of Athens, 2021. DOI: 10.7712/140121.7952.18383.

[26] B. K.-U. Geiser A. Wüchner R. “Variable filter radii for Vertex Morphing based design of adaptive structures”. In: *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia and Industry*. 2017.

[27] A. Ghantasala, R. N. Asl, A. Geiser, A. Brodie, E. Papoutsis, and K.-U. Bletzinger. “Realization of a Framework for Simulation-Based Large-Scale Shape Optimization Using Vertex Morphing”. In: *Journal of Optimization Theory and Applications* 189.1 (Mar. 2021), pp. 164–189. DOI: 10.1007/s10957-021-01826-x.

[28] A. Ghantasala, J. Diller, A. Geiser, D. Wenzler, D. Siebert, C. Radlbeck, R. Wüchner, M. Mensinger, and K.-U. Bletzinger. “Node-Based Shape Optimization and Mechanical Test Validation of Complex Metal Components and Support Structures, Manufactured by Laser Powder Bed Fusion”. In: July 2021, pp. 10–17. ISBN: 978-3-030-80461-9. DOI: 10.1007/978-3-030-80462-6{_}2.

[29] D. Goldfarb. “A family of variable-metric methods derived by variational means”. In: *Mathematics of Computation* 24.109 (1970), pp. 23–26. DOI: 10.1090/s0025-5718-1970-0258249-6.

[30] G. H. Golub and C. F. V. Loan. *Matrix Computations*. 3rd Edition. The Johns Hopkins Press, Baltimore, 1996.

[31] L. Grippo, F. Lampariello, and S. Lucidi. “A Nonmonotone Line Search Technique for Newton’s Method”. In: *SIAM Journal on Numerical Analysis* 23.4 (2022/11/10/ 1986), pp. 707–716.

[32] M. Hojjat. “Node-based parametrization for shape optimal design”. Dissertation. Technische Universität München, 2015.

[33] M. Hojjat, E. Stavropoulou, and K.-U. Bletzinger. "The Vertex Morphing method for node-based shape optimization". In: *Computer Methods in Applied Mechanics and Engineering* 268 (Jan. 2014), pp. 494–513. DOI: 10.1016/j.cma.2013.10.015.

[34] Y. Huang, Y.-H. Dai, X.-W. Liu, and H. Zhang. "On the acceleration of the Barzilai–Borwein method". In: *Computational Optimization and Applications* 81.3 (Jan. 2022), pp. 717–740. DOI: 10.1007/s10589-022-00349-z.

[35] K. A. James, G. J. Kennedy, and J. R. Martins. "Concurrent aerostructural topology optimization of a wing box". In: *Computers and Structures* 134 (Apr. 2014), pp. 1–17. DOI: 10.1016/j.compstruc.2013.12.007.

[36] M. Langelaar. "Topology optimization of 3D self-supporting structures for additive manufacturing". In: *Additive Manufacturing* 12 (Oct. 2016), pp. 60–70. DOI: 10.1016/j.addma.2016.06.010.

[37] H. Li, I. Alhashim, H. Zhang, A. Shamir, and D. Cohen-Or. "Stackabilization". In: *ACM Transactions on Graphics* 31.6 (Nov. 2012), pp. 1–9. DOI: 10.1145/2366145.2366177.

[38] A. K. Lianos, H. Bikas, and P. Stavropoulos. "A Shape Optimization Method for Part Design Derived from the Buildability Restrictions of the Directed Energy Deposition Additive Manufacturing Process". In: *Designs* 4.3 (July 2020), p. 19. DOI: 10.3390/designs4030019.

[39] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.1 (1989), pp. 503–528. DOI: 10.1007/BF01589116.

[40] J. Liu et al. "Current and future trends in topology optimization for additive manufacturing". In: *Structural and Multidisciplinary Optimization* 57.6 (2018), pp. 2457–2483. DOI: 10.1007/s00158-018-1994-3.

[41] J. R. R. A. Martins and A. Ning. *Engineering Design Optimization*. Cambridge University Press, 2021. DOI: 10.1017/9781108980647.

[42] Z. Michalewicz, R. Hinterding, and M. Michalewicz. "Evolutionary Algorithms". In: *Fuzzy Evolutionary Computation*. Ed. by W. Pedrycz. Boston, MA: Springer US, 1997, pp. 3–31. ISBN: 978-1-4615-6135-4. DOI: 10.1007/978-1-4615-6135-4_1.

[43] P. M. Müller, N. Kühl, M. Siebenborn, K. Deckelnick, M. Hinze, and T. Rung. "A novel p-harmonic descent approach applied to fluid dynamic shape optimization". In: *Structural and Multidisciplinary Optimization* 64.6 (Aug. 2021), pp. 3489–3503. DOI: 10.1007/s00158-021-03030-x.

[44] R. Najian Asl. "Shape optimization and sensitivity analysis of fluids, structures, and their interaction using Vertex Morphing parametrization". Dissertation. München: Technische Universität München, 2019.

[45] R. Najian Asl, S. Shayegan, A. Geiser, M. Hojjat, and K.-U. Bletzinger. "A consistent formulation for imposing packaging constraints in shape optimization using Vertex Morphing parametrization". In: *Structural and Multidisciplinary Optimization* 56 (Oct. 2017), pp. 1–13. DOI: 10.1007/s00158-017-1819-9.

[46] N. H. Oleg Burdakov Yu-Hong Dai. "Stabilized Barzilai-Borwein Method". In: *Journal of Computational Mathematics* 37.6 (June 2019), pp. 916–936. DOI: 10.4208/jcm.1911-m2019-0171.

[47] C. Othmer. "Adjoint methods for car aerodynamics". In: *Journal of Mathematics in Industry* 4.1 (2014), p. 6. DOI: 10.1186/2190-5983-4-6.

[48] M. Raydan. "The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem". In: *SIAM Journal on Optimization* 7 (Feb. 1997). DOI: 10.1137/S1052623494266365.

[49] J. Rosen. "The gradient projection method for nonlinear programming: Part II". In: *Siam Journal on Applied Mathematics - SIAMAM* 9 (Jan. 1961).

[50] J. Rosen. "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints". In: *Journal of The Society for Industrial and Applied Mathematics* 8 (Mar. 1960). DOI: 10.1137/0108011.

[51] V. Savchuk, D. Medvedev, and O. Vjarvilskaja. *Theoretical Mechanics*. Belarusian State University, 2016. ISBN: 978-985-566-356-1.

[52] M. Schramm, B. Stoevesandt, and J. Peinke. "Optimization of Airfoils Using the Adjoint Approach and the Influence of Adjoint Turbulent Viscosity". In: *Computation* 6.1 (Jan. 2018), p. 5. DOI: 10.3390/computation6010005.

[53] D. F. Shanno. "Conditioning of quasi-Newton methods for function minimization". In: *Mathematics of Computation* 24.111 (1970), pp. 647–656. DOI: 10.1090/s0025-5718-1970-0274029-x.

[54] A. Stück and T. Rung. "Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation". In: *Computers & Fluids* 47.1 (Aug. 2011), pp. 22–32. DOI: 10.1016/j.compfluid.2011.01.041.

[55] G. Vanderplaats. *Multidiscipline Design Optimization*. Vanderplaats Research & Development, Inc, 2007.

[56] G. Vanderplaats. "Very Large Scale Continuous and Discrete Variable Optimization". In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, Aug. 2004. DOI: 10.2514/6.2004-4458.

[57] B. Zhou, L. Gao, and Y.-H. Dai. "Gradient Methods with Adaptive Step-Sizes". In: *Computational Optimization and Applications* 35.1 (Mar. 2006), pp. 69–86. DOI: 10.1007/s10589-006-6446-0.

Bisherige Titel der Schriftenreihe

Band	Titel
1	Frank Koschnick, *Geometrische Lockingeffekte bei Finiten Elementen und ein allgemeines Konzept zu ihrer Vermeidung*, 2004.
2	Natalia Camprubi, *Design and Analysis in Shape Optimization of Shells*, 2004.
3	Bernhard Thomee, *Physikalisch nichtlineare Berechnung von Stahlfaserbetonkonstruktionen*, 2005.
4	Fernaß Daoud, *Formoptimierung von Freiformschalen - Mathematische Algorithmen und Filtertechniken*, 2005.
5	Manfred Bischoff, *Models and Finite Elements for Thin-walled Structures*, 2005.
6	Alexander Hörmann, *Ermittlung optimierter Stabwerkmodelle auf Basis des Kraftflusses als Anwendung plattformunabhängiger Prozesskopplung*, 2006.
7	Roland Wüchner, *Mechanik und Numerik der Formfindung und Fluid-Struktur-Interaktion von Membrantragwerken*, 2006.
8	Florian Jurecka, *Robust Design Optimization Based on Metamodeling Techniques*, 2007.
9	Johannes Linhard, *Numerisch-mechanische Betrachtung des Entwurfsprozesses von Membrantragwerken*, 2009.
10	Alexander Kupzok, *Modeling the Interaction of Wind and Membrane Structures by Numerical Simulation*, 2009.

Band Titel

11 Bin Yang, *Modified Particle Swarm Optimizers and their Application to Robust Design and Structural Optimization*, 2009.

12 Michael Fleischer, *Absicherung der virtuellen Prozesskette für Folgeoperationen in der Umformtechnik*, 2009.

13 Amphon Jrusjrungkiat, *Nonlinear Analysis of Pneumatic Membranes - From Subgrid to Interface*, 2009.

14 Alexander Michalski, *Simulation leichter Flächentragwerke in einer numerisch generierten atmosphärischen Grenzschicht*, 2010.

15 Matthias Firl, *Optimal Shape Design of Shell Structures*, 2010.

16 Thomas Gallinger, *Effiziente Algorithmen zur partitionierten Lösung stark gekoppelter Probleme der Fluid-Struktur-Wechselwirkung*, 2011.

17 Josef Kiendl, *Isogeometric Analysis and Shape Optimal Design of Shell Structures*, 2011.

18 Joseph Jordan, *Effiziente Simulation großer Mauerwerksstrukturen mit diskreten Rissmodellen*, 2011.

19 Albrecht von Boetticher, *Flexible Hangmurenbarrieren: Eine numerische Modellierung des Tragwerks, der Hangmure und der Fluid-Struktur-Interaktion*, 2012.

20 Robert Schmidt, *Trimming, Mapping, and Optimization in Isogeometric Analysis of Shell Structures*, 2013.

21 Michael Fischer, *Finite Element Based Simulation, Design and Control of Piezoelectric and Lightweight Smart Structures*, 2013.

22 Falko Hartmut Dieringer, *Numerical Methods for the Design and Analysis for Tensile Structures*, 2014.

Band Titel

23 Rupert Fisch, *Code Verification of Partitioned FSI Environments for Lightweight Structures*, 2014.

24 Stefan Sicklinger, *Stabilized Co-Simulation of Coupled Problems Including Fields and Signals*, 2014.

25 Madjid Hojjat, *Node-based parametrization for shape optimal design*, 2015.

26 Ute Israel, *Optimierung in der Fluid-Struktur-Interaktion - Sensitivitätsanalyse für die Formoptimierung auf Grundlage des partitionierten Verfahrens*, 2015.

27 Electra Stavropoulou, *Sensitivity analysis and regularization for shape optimization of coupled problems*, 2015.

28 Daniel Markus, *Numerical and Experimental Modeling for Shape Optimization of Offshore Structures*, 2015.

29 Pablo Suárez, *Design Process for the Shape Optimization of Pressurized Bulkheads as Components of Aircraft Structures*, 2015.

30 Armin Widhammer, *Variation of Reference Strategy - Generation of Optimized Cutting Patterns for Textile Fabrics*, 2015.

31 Helmut Masching, *Parameter Free Optimization of Shape Adaptive Shell Structures*, 2016.

32 Hao Zhang, *A General Approach for Solving Inverse Problems in Geophysical Systems by Applying Finite Element Method and Metamodel Techniques*, 2016.

33 Tianyang Wang, *Development of Co-Simulation Environment and Mapping Algorithms*, 2016.

34 Michael Breitenberger, *CAD-integrated Design and Analysis of Shell Structures*, 2016.

Band Titel

35 Önay Can, *Functional Adaptation with Hyperkinematics using Natural Element Method: Application for Articular Cartilage*, 2016.

36 Benedikt Philipp, *Methodological Treatment of Non-linear Structural Behavior in the Design, Analysis and Verification of Lightweight Structures*, 2017.

37 Michael Andre, *Aeroelastic Modeling and Simulation for the Assessment of Wind Effects on a Parabolic Trough Solar Collector*, 2018.

38 Andreas Apostolatos, *Isogeometric Analysis of Thin-Walled Structures on Multipatch Surfaces in Fluid-Structure Interaction*, 2018.

39 Altuğ Emiroğlu, *Multiphysics Simulation and CAD-Integrated Shape Optimization in Fluid-Structure Interaction*, 2019.

40 Mehran Saeedi, *Multi-Fidelity Aeroelastic Analysis of Flexible Membrane Wind Turbine Blades*, 2017.

41 Reza Najian Asl, *Shape Optimization and Sensitivity Analysis of Fluids, Structures, and their Interaction Using Vertex Morphing Parametrization*, 2019.

42 Ahmed Abodonya, *Verification Methodology for Computational Wind Engineering Prediction of Wind Loads on Structures*, 2020.

43 Anna Maria Bauer, *CAD-Integrated Isogeometric Analysis and Design of Lightweight Structures*, 2020.

44 Andreas Winterstein, *Modeling and Simulation of Wind- Structure Interaction of Slender Civil Engineering Structures Including Vibration Mitigation Systems*, 2020.

45 Franz-Josef Ertl, *Vertex Morphing for constrained shape optimization of three-dimensional solid structures*, 2020.

Band Titel

46 Daniel Baumgärtner, *On the grid-based shape optimization of structures with internal flow and the feedback of shape changes into a CAD model*, 2020.

47 Mohamed Magdi Mohamed Mohamed Khalil, *Combining Physics-Based Models and Machine Learning for an Enhanced Structural Health Monitoring*, 2021.

48 Long Chen, *Gradient descent akin method*, 2021.

49 Aditya Ghantasala, *Coupling Procedures for Fluid-Fluid and Fluid-Structure Interaction Problems Based on Domain Decomposition Methods*, 2021.

50 Ann-Kathrin Goldbach, *The CAD-integrated Design Cycle for Structural Membranes*, 2021.

51 Iñigo Pablo Lòpez Canalejo, *A Finite-Element Transonic Potential Flow Solver with an Embedded Wake Approach for Aircraft Conceptual Design*, 2022

52 Mayu Sakuma, *An Application of Multi-Fidelity Uncertainty Quantification for Computational Wind Engineering*, 2022.

53 Suneth Warnakulasuriya, *Development of Methods for Finite Element-Based Sensitivity Analysis and Goal-Directed Mesh Refinement Using the Adjoint Approach for Steady and Transient Flows*, 2022.

54 Klaus Bernd Sautter, *Modeling and Simulation of Flexible Protective Structures by Coupling Particle and Finite Element Methods*, 2022.

55 Efthymios Papoutsis, *On the incorporation of industrial constraints in node-based optimization for car body design*, 2023

56 Thomas Josef Oberbichler, *A modular and efficient implementation of isogeometric analysis for the interactive CAD-integrated design of lightweight structures. Dissertation*, 2023

57 Tobias Christoph Teschemacher, *CAD-integrated constitutive modelling, analysis, and design of masonry structures. Dissertation*, 2023

58 Shahrokh Shayegan, *Enhanced Algorithms for Fluid-Structure Interaction Simulations – Accurate Temporal Discretization and Robust Convergence Acceleration*, 2023