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 / Published online: 6 July 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 S Andreasen, 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

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 $\Gamma$ with surface coordinates $(\xi, \eta, \zeta)$:

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

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 $\boldsymbol{A}$:

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

where $I$ 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.

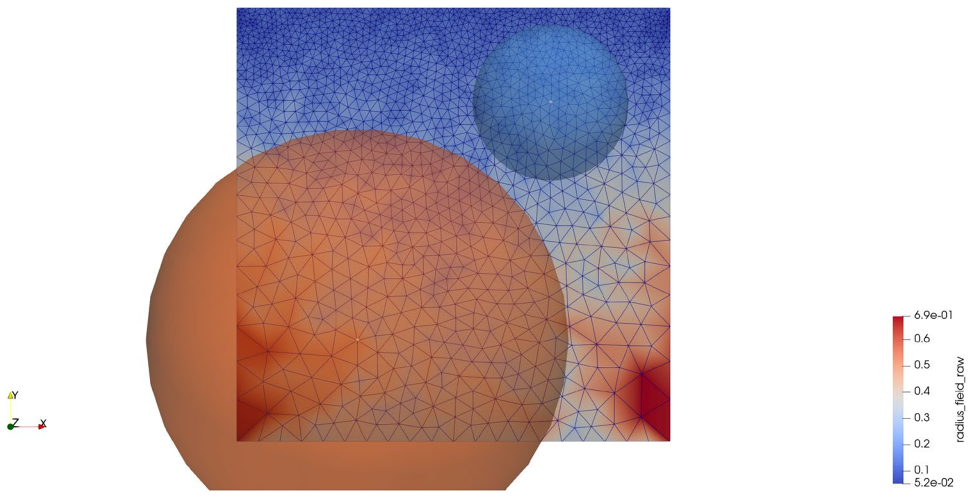


**Fig. 1** Individual radii for each node of the mesh, orange—radius of the big element, blue—radius of the small element

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

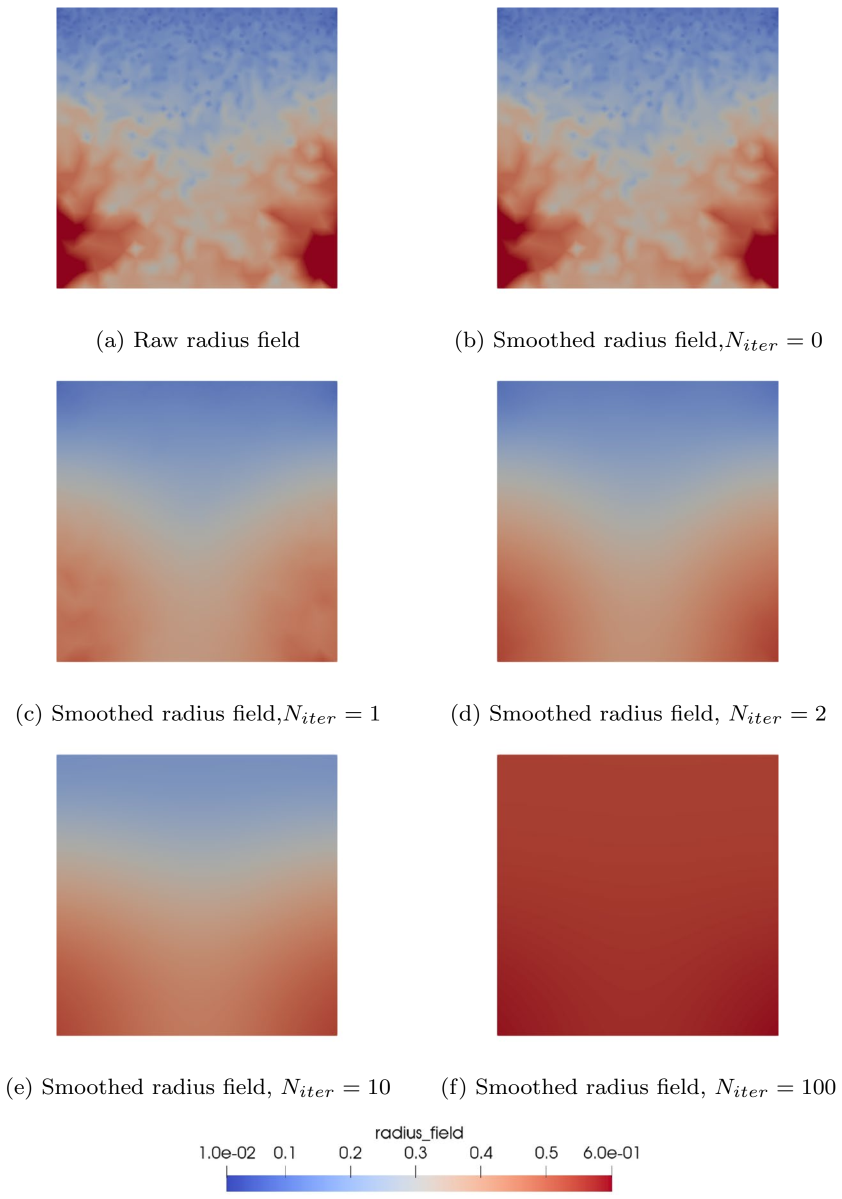


**Fig. 2** Radius field comparison with respect to $N_{iter}$ smoothing iterations

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:

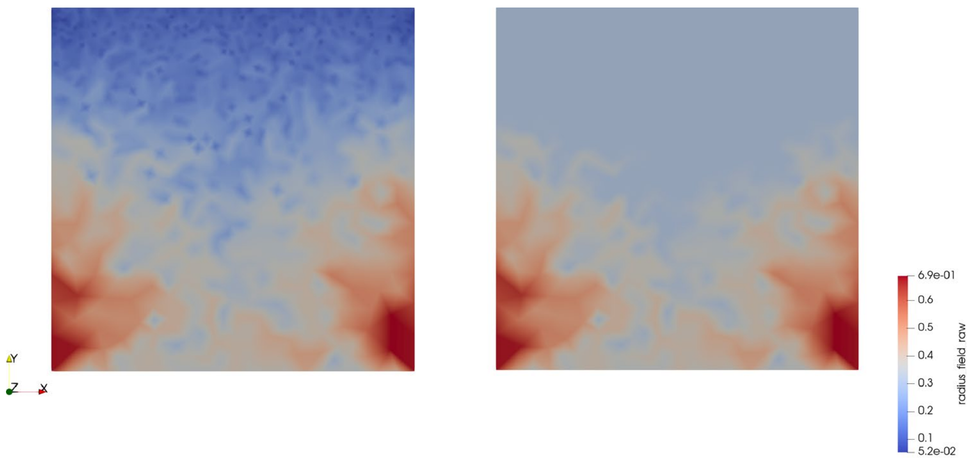
$$\nabla f(x_i) = \boldsymbol{n}_i A_i, \tag{9}$$

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

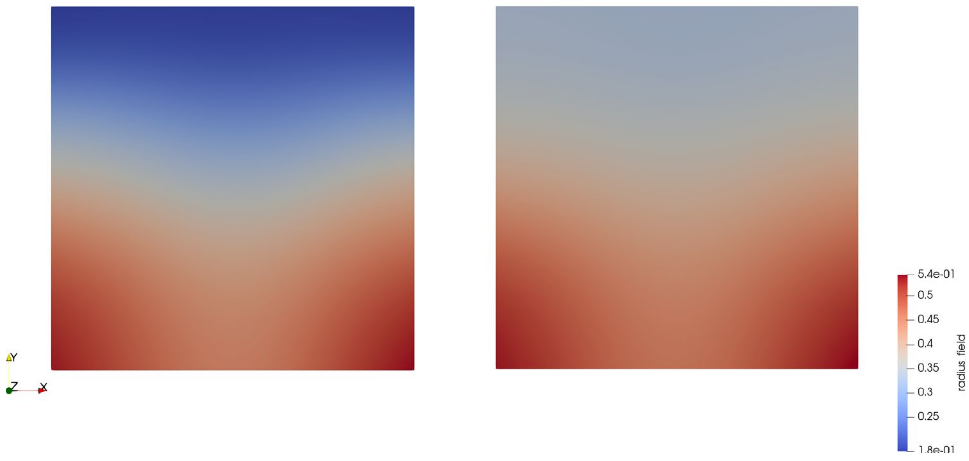
$$\Delta \boldsymbol{x} = \boldsymbol{A}\left(\frac{\alpha}{||s||}s\right), \tag{10}$$

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 $BS_j^{(i)}$:

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

$$LBV_j^{(i)} = CBV_j^{(i)} - BS_j^{(i)}$$

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

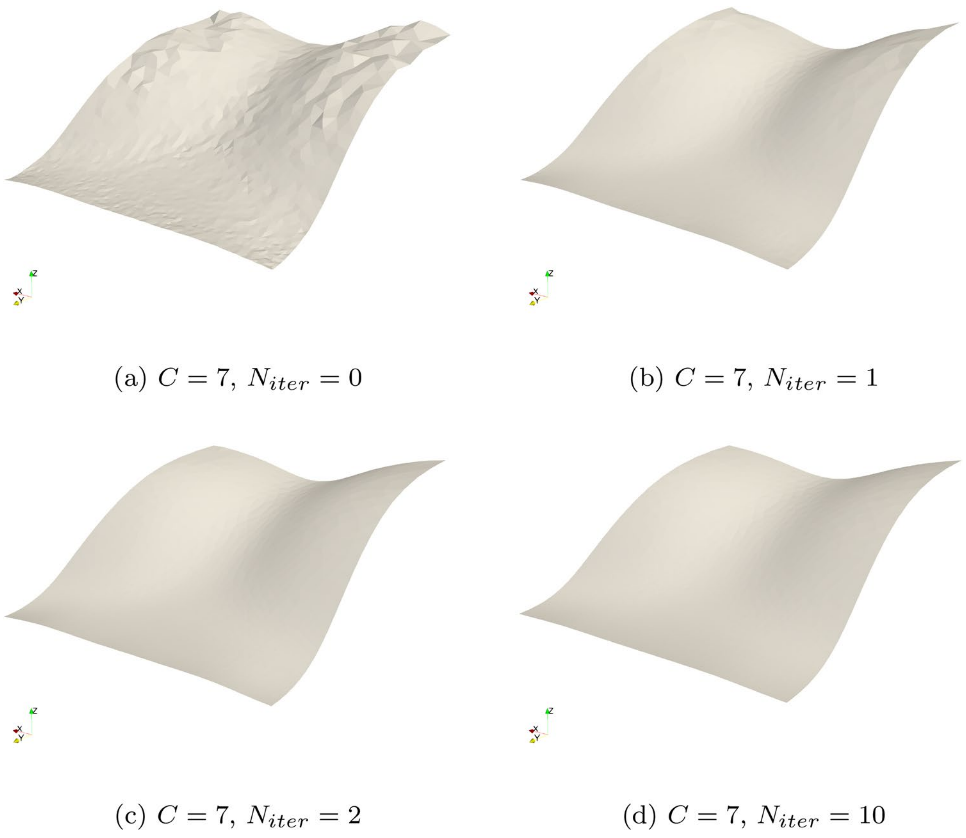


**Fig. 4** Deformed plate with respect to different $N_{iter}$ numbers

(a) $C = 7,\ N_{iter} = 0$
(b) $C = 7,\ N_{iter} = 1$
(c) $C = 7,\ N_{iter} = 2$
(d) $C = 7,\ N_{iter} = 10$

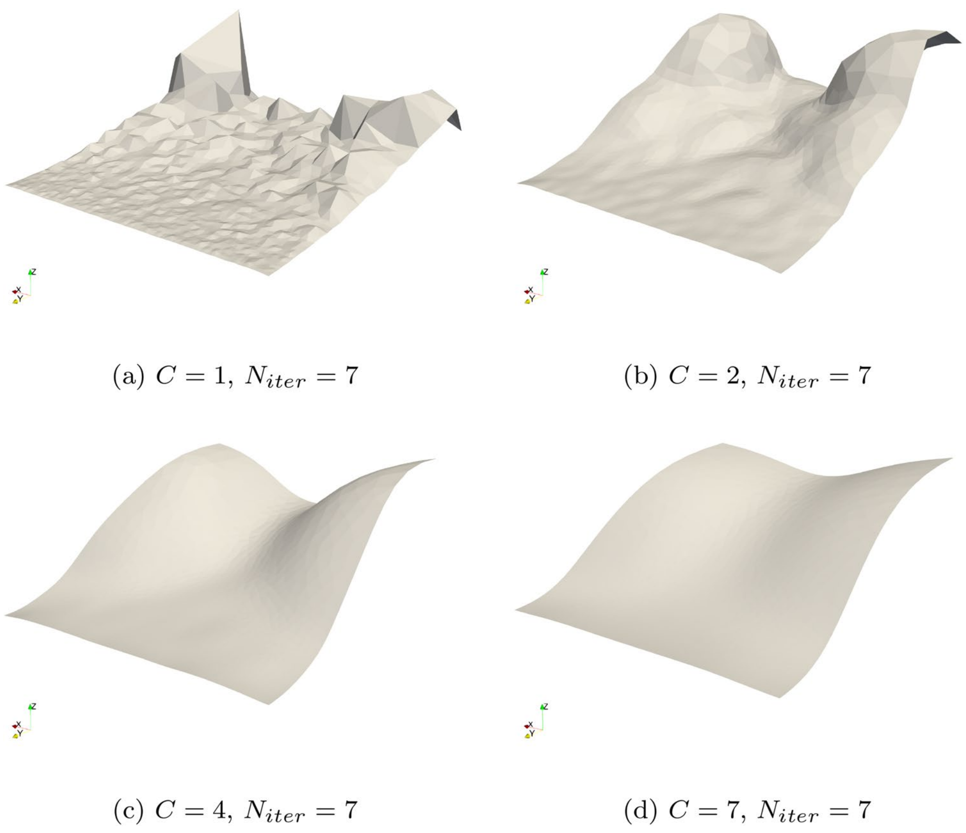


**Fig. 5** Deformed plate with respect to different $C$ constant

(a) $C = 1,\ N_{iter} = 7$
(b) $C = 2,\ N_{iter} = 7$
(c) $C = 4,\ N_{iter} = 7$
(d) $C = 7,\ N_{iter} = 7$

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

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 $\omega^{max}$ 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 $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i)}) - \nabla f(\boldsymbol{x}^{(i-1)})$ 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

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

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:

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

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

$$\boldsymbol{y}_k^{(i)} = \boldsymbol{s}_k^{(i-1)} - \boldsymbol{s}_k^{(i)} \tag{22}$$

$$\Delta\boldsymbol{x}^{(i)} = \boldsymbol{H}^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{23}$$

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:

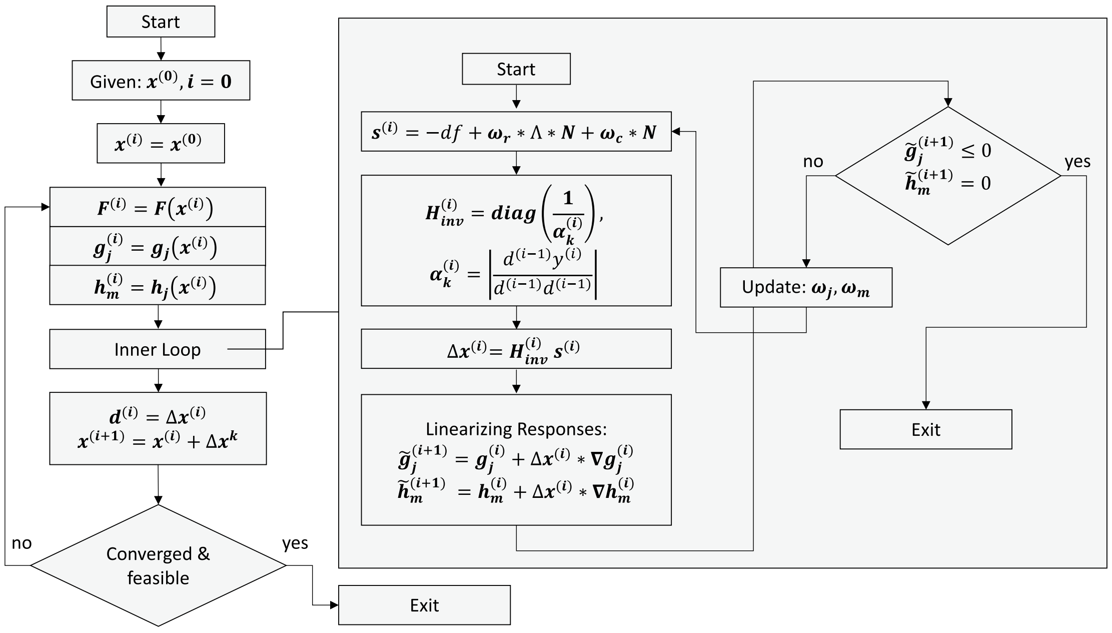

Start
Given: $x^{(0)}, i = 0$
$x^{(i)} = x^{(0)}$
$F^{(i)} = F(x^{(i)})$
$g_j^{(i)} = g_j(x^{(i)})$
$h_m^{(i)} = h_j(x^{(i)})$
Inner Loop
$d^{(i)} = \Delta x^{(i)}$
$x^{(i+1)} = x^{(i)} + \Delta x^k$
no
Converged & feasible
yes
Exit
Start
$s^{(i)} = -df + \omega_r * \Lambda * N + \omega_c * N$
$H_{inv}^{(i)} = diag\left(\frac{1}{\alpha_k^{(i)}}\right)$,
$\alpha_k^{(i)} = \left|\frac{d^{(i-1)}y^{(i)}}{d^{(i-1)}d^{(i-1)}}\right|$
$\Delta x^{(i)} = H_{inv}^{(i)} s^{(i)}$
Linearizing Responses:
$\tilde{g}_j^{(i+1)} = g_j^{(i)} + \Delta x^{(i)} * \nabla g_j^{(i)}$
$\tilde{h}_m^{(i+1)} = h_m^{(i)} + \Delta x^{(i)} * \nabla h_m^{(i)}$
no
$\tilde{g}_j^{(i+1)} \leq 0$
$\tilde{h}_m^{(i+1)} = 0$
yes
Update: $\omega_j, \omega_m$
Exit

**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. (23);
   (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:

$$w_{j,k}^{(i)} = \frac{g_j(x_k^{(i)}) - LBV_j^{(i)}}{BS_j^{(i)}} \tag{24}$$

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:

$$\begin{aligned} g_j(x^{(i)}) &= \max_k(g_j(x_k^{(i)})) \\ \nabla g(x^{(i)}) &= \sum_k w_{j,k}^{(i)} \cdot \nabla g_j(x_k^{(i)}). \\ w_j^{(i)} &= \max_k(w_{j,k}^{(i)}) \end{aligned} \tag{25}$$

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

$$\tilde{g}(x_k^{(i+1)}) = g(x_k^{(i)}) + \nabla g_j(x_k^{(i)} \boldsymbol{\Delta x}^{(i)}). \tag{26}$$

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).

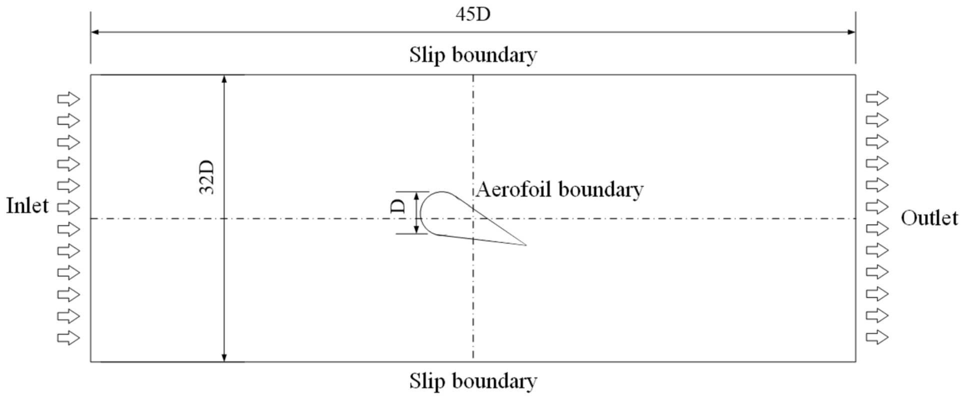


**Fig. 7** Aerofoil problem configuration used in RANS in 2D

## 5 Academic experiment

This numerical investigation is designed to illustrate the applicability of the methods above in a highly non-linear 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., $\Gamma_{inlet}$) 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., $\Gamma_{outlet}$) is applied with a $0\,Pa$ Dirichlet boundary condition for $P$, zero gradient boundary conditions for variables $u, k, \epsilon, \omega$. The slip condition (i.e., $\Gamma_{far}$) 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., $\Gamma_s$) 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

$$\begin{aligned} &\boldsymbol{R}_{\phi} = \boldsymbol{0} \quad \forall \phi \in \{u, v, w, p, k, \omega\} \\ &G_{centroid}(\boldsymbol{s}) = 0 \\ &J_{lift}(\boldsymbol{w}(\boldsymbol{s}), \boldsymbol{s}) - J_{lift}\left(\boldsymbol{w}(\boldsymbol{s}_{initial}), \boldsymbol{s}_{initial}\right) \geq 0.0 \end{aligned} \quad , \qquad (27)$$

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 $\Gamma_s$. It is applied to constrain aerofoil geometry to be present at the center of $\Omega$ for all design iterations. AVM is used to smooth the noisy sensitivity field on the aerofoil boundary, and QN–BB–RGP

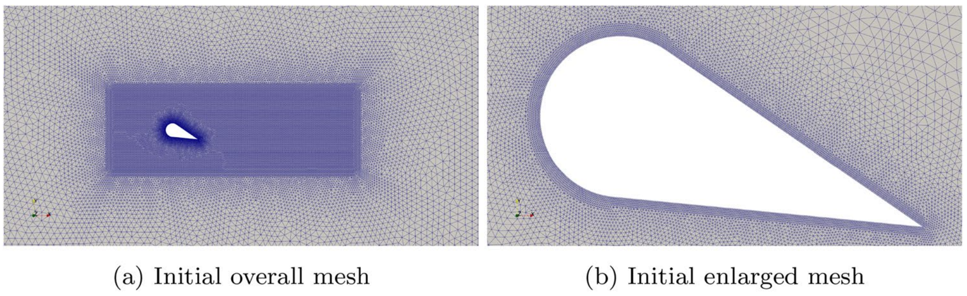


**Fig. 8** Initial mesh for a 2D aerofoil optimization problem

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

$$G_{centroid} = \left\| \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_{current,i} - \boldsymbol{x}_{\Omega_{center}}) \right\|_2^2, \tag{28}$$

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

## 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.

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,

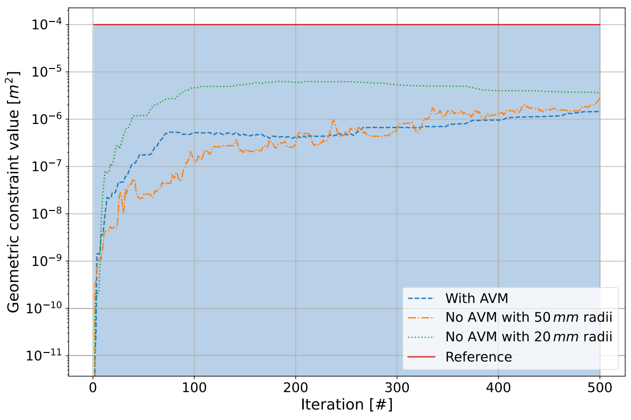


**Fig. 10** Geometric constraint variation with optimization design iterations

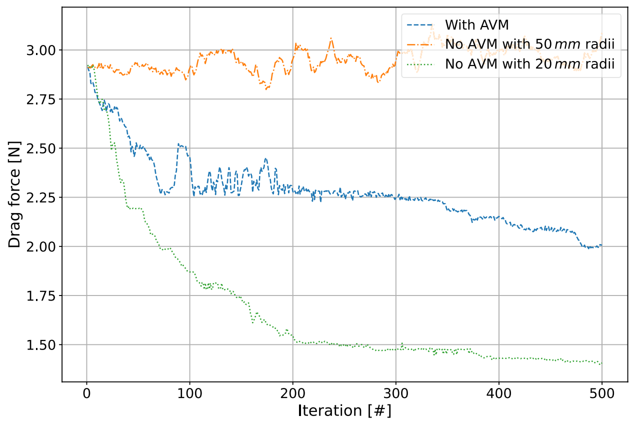


**Fig. 9** Drag force variation with optimization design iterations

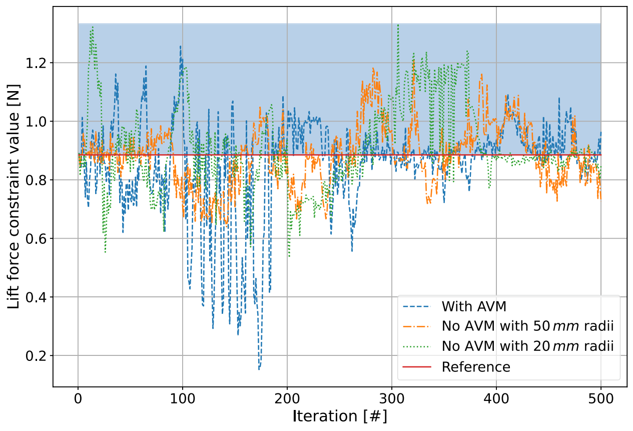


**Fig. 11** Lift constraint variation with optimization design iterations

**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,

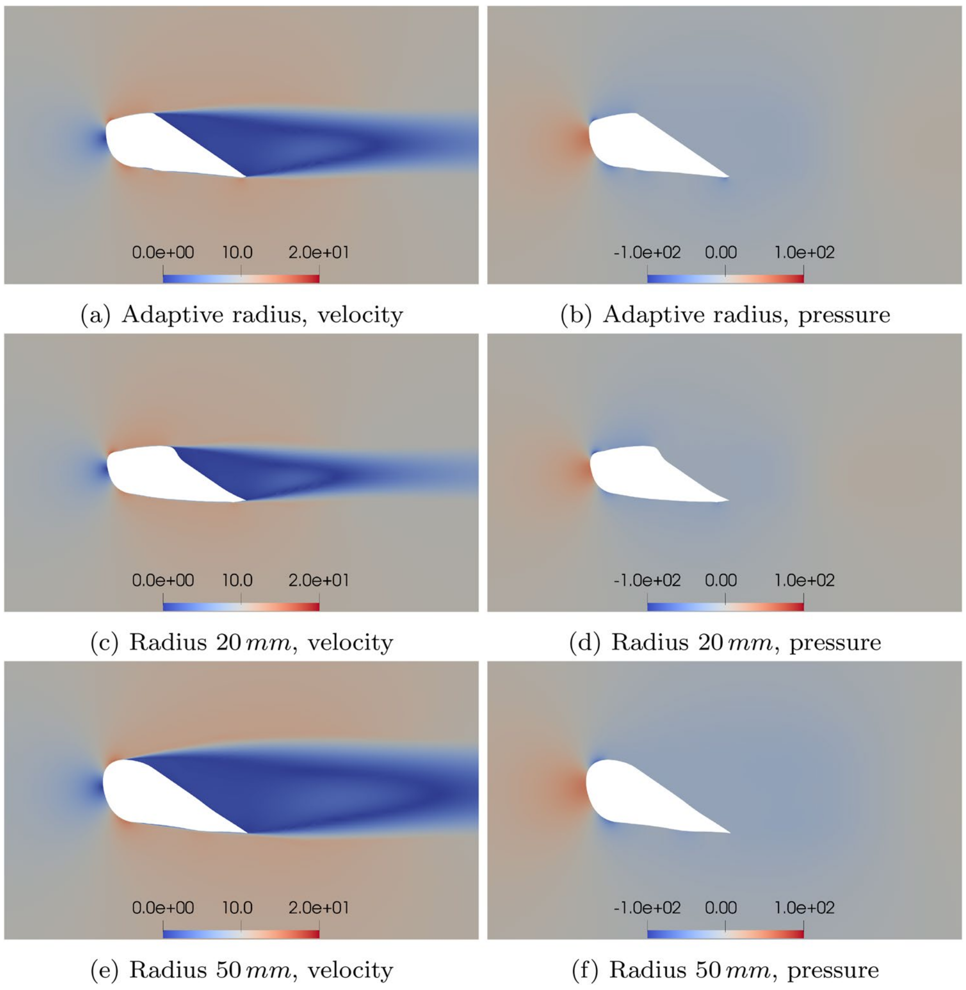
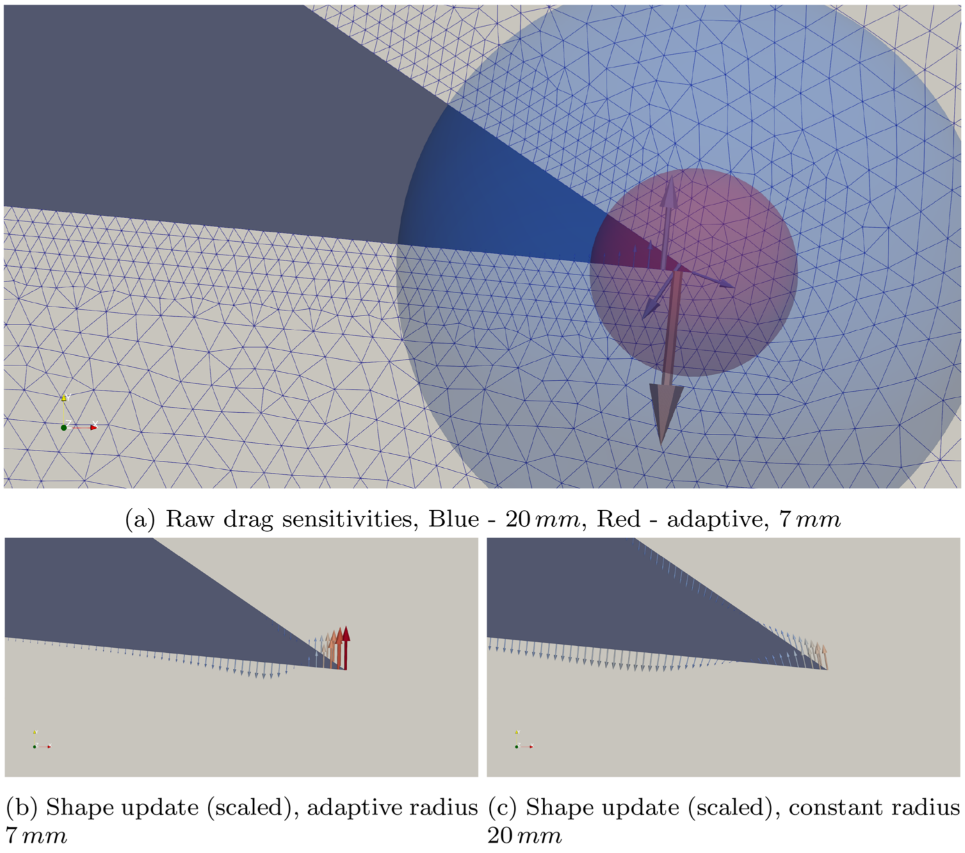


**Fig. 12** Optimized aerofoil design, on left—velocity [$ms^{-1}$] and on right—pressure [$Pa$] distributions


(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+$^{TM}$ 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

**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 |

Section 4.4. The details of the turbulence models, solvers, 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

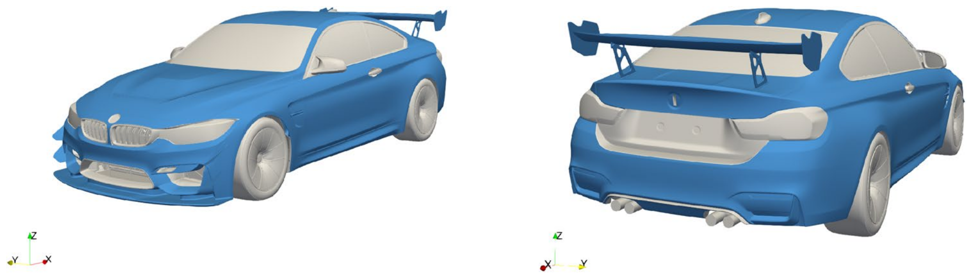


**Fig. 14** Geometry of the CFD model: design surface—blue, non-design parts—light gray

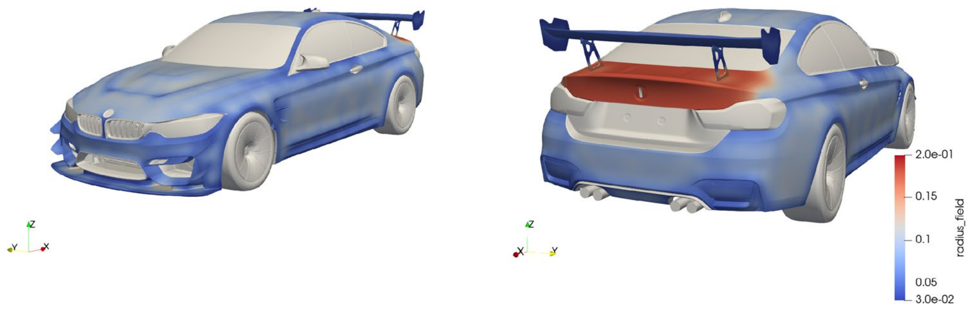


**Fig. 15** Radius field over the design surface

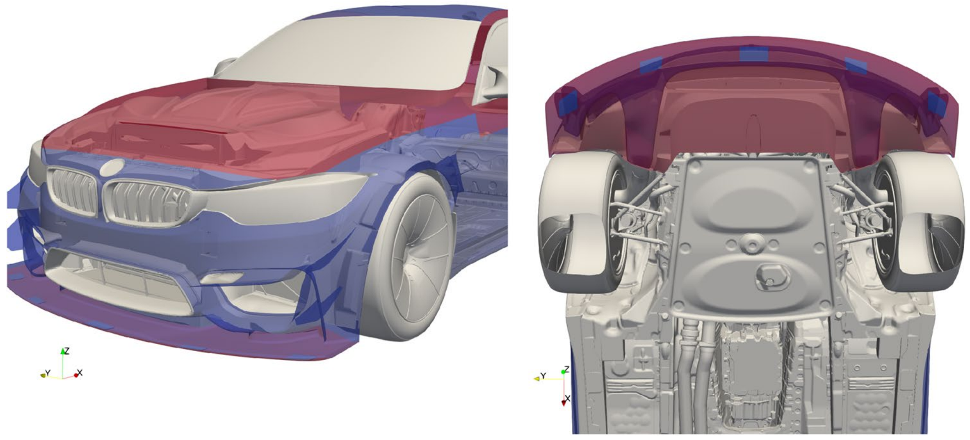
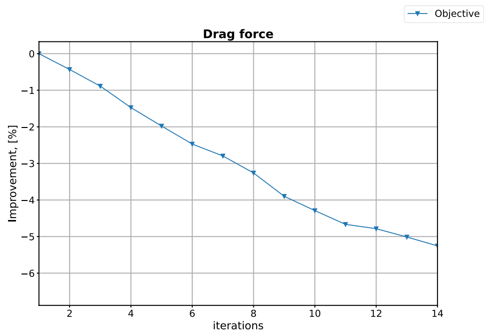
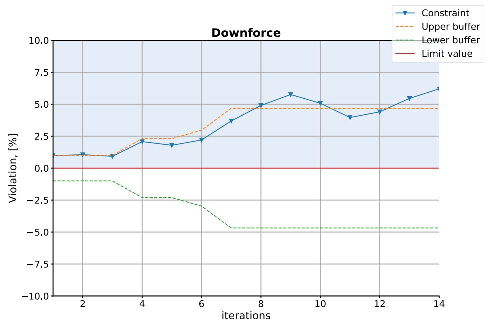


**Fig. 16** Geometrical constraint: constrained zone—red, free design surface—blue, non-design parts—light gray

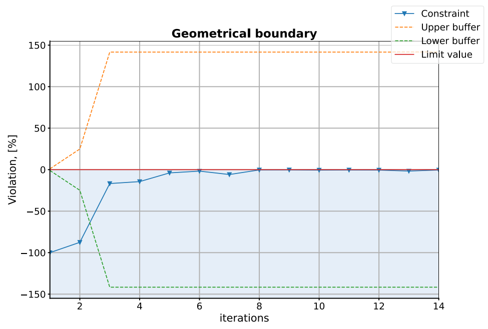
**Fig. 17** Optimization results: response function evaluations & size of shape updates



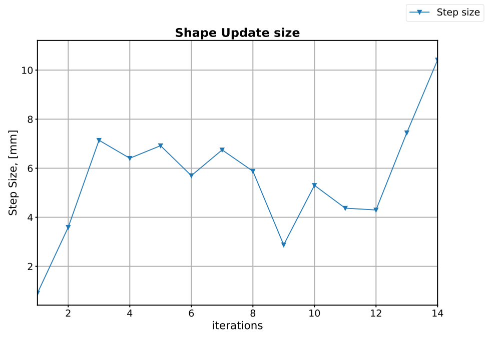
(a) Objective



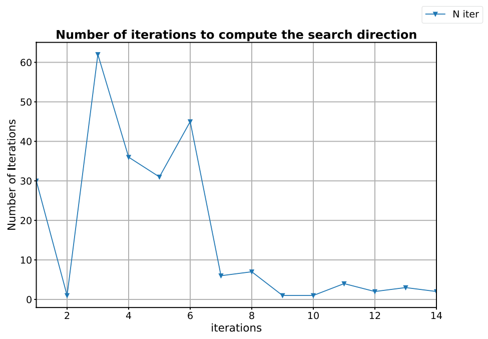
(b) Constraint #1



(c) Constraint #2



(d) Size of the shape update



(e) Number of inner iterations

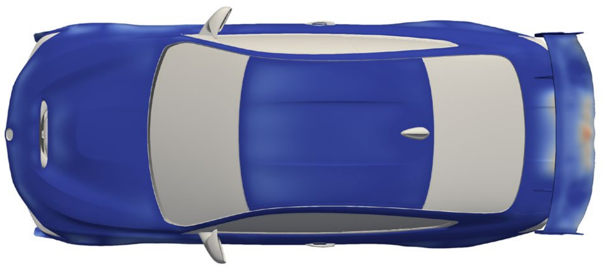
**Table 4** Computation time [hour]

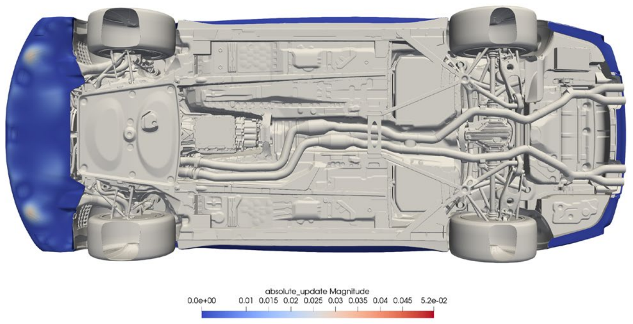
| Computation time [hour] | |
|---|---|
| Aver. primal analysis | 10 |
| Aver adjoint analysis per response | 2.5 |
| Aver. time per optimization iteration | 14.95 |
| Aver. time to find $\Delta x$ | 0.85 |
| Other operations | 1.6 |
| Overall optimization time | 210 |
| CPU hours | 120960 |

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 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


(a) Upper view



(b) Lower view

**Fig. 18** Optimization results: shape change on top & bottom view

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, 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”).

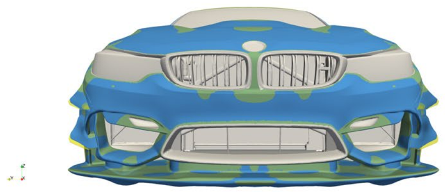
In Figure 19d, one can see the difference on the final shape update with different radii. The rear wing has a radius of 5 mm, while the trunk lid is – 20 mm. The final shape of the trunk lid is very smooth, and the wavelength of shape change is large. In contrast, the shape change of the rear wing is 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.

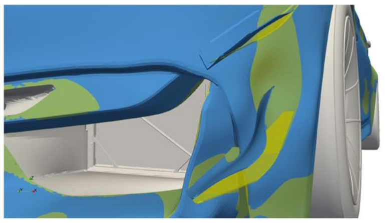
## 7 Conclusions

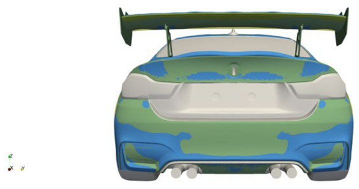
Adaptive Vertex Morphing extends Vertex Morphing parameterization to solve large-scale optimization problems with local control on the final shape. It allows reasonable control of the final form and finds various solutions. The QN–BB–RGP method shows good potential as an acceleration and stabilization 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.

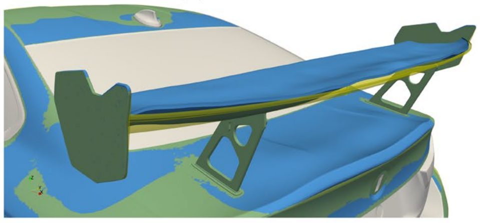
**Acknowledgements** The authors wish to thank the ShapeModule project (BMW Group, München) for the support. Thanks are also due to Mr. Vignesh Manickavasagam Manian (BMW Group) for technical support. The authors acknowledge the support of colleagues Dr.-Ing. Reza Najian Asl and Mr. Armin Geiser for 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, shape-module@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 stepsizes. Comput Optim Appl 35(1):69–86. https://doi.org/10.1007/s10589-006-6446-0