



**QUEEN'S
UNIVERSITY
BELFAST**

Parametric design velocity computation for CAD-based design optimization using adjoint methods

Agarwal, D., Robinson, T., Armstrong, C., Marques, S., Vasilopoulos, I., & Meyer, M. (2017). Parametric design velocity computation for CAD-based design optimization using adjoint methods. *Engineering With Computers*, 1-15. DOI: 10.1007/s00366-017-0534-x

Published in:
Engineering With Computers

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

Queen's University Belfast - Research Portal:
[Link to publication record in Queen's University Belfast Research Portal](#)

Publisher rights

Copyright 2017 the authors.

This is an open access article published under a Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution and reproduction in any medium, provided the author and source are cited.




General rights

Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Parametric design velocity computation for CAD-based design optimization using adjoint methods

Dheeraj Agarwal¹  · Trevor T. Robinson¹  · Cecil G. Armstrong¹  ·
Simão Marques¹ · Ilias Vasilopoulos^{2,3} · Marcus Meyer²

Received: 28 February 2017 / Accepted: 18 July 2017
© The Author(s) 2017. This article is an open access publication

Abstract This paper presents an efficient optimization process, where the parameters defining the features in a feature-based CAD model are used as design variables. The process exploits adjoint methods for the computation of gradients, and as such the computational cost is essentially independent of the number of design variables, making it ideal for optimization in large design spaces. The novelty of this paper lies in linking the adjoint surface sensitivity information with geometric sensitivity values, referred to as design velocities, computed for CAD models created in commercial CAD systems (e.g. CATIA V5 or Siemens NX). This process computes gradients based on the CAD feature parameters, which are used by the optimization algorithm, which in turn updates the values of the same parameters in the CAD model. In this paper, the design velocity and resulting gradient calculations are validated against analytical and finite-difference results. The proposed approach is demonstrated to be compatible with different commercial CAD packages and computational fluid dynamics solvers.

Keywords CAD · Design velocity · Adjoint method · Gradient · Optimization

1 Introduction

In the context of the industrial design process, a product design typically starts with CAD geometry and eventually delivers optimized geometry in CAD. However, currently there is no efficient way of either optimizing directly on feature-based CAD geometries or generating a feature-based CAD model from optimization performed on computer-aided engineering (CAE) meshes. The current techniques to capture the geometry are inefficient and lose important geometric details. Hence, there is a need to use CAD models within the optimization framework to strengthen the industrial workflow. Current research in this area aims to enable shape optimization using either a *dumb* geometry, which is a non-parametric CAD model from which the construction history has been removed, or a CAD model with its construction history, features, and parameters included. For the latter, the shape of a model can be updated by changing values of the parameters that define it. One of the main advantages of this approach is that as the optimized model is a feature-based CAD model, it can be used directly for downstream applications including manufacturing and process planning. The constraints imposed by the features in the CAD model feature tree will mean that the optimized part can be manufactured, provided the features were well chosen. To a large extent this will depend on the skill and experience of the CAD model creator, and their ability to visualize and parameterize the design space. The drawbacks of this approach are that, as the choice of CAD features constrains how the model can change shape, it may stifle the creation of innovative solutions. It may therefore be necessary to insert additional features into the CAD model feature tree if a radical change in shape or performance is desired.

✉ Trevor T. Robinson
t.robinson@qub.ac.uk

¹ School of Mechanical and Aerospace Engineering, Queen's University Belfast, Belfast BT9 5AH, Northern Ireland, UK

² Rolls-Royce Deutschland, Dahlewitz, Blankenfelde-Mahlow, Germany

³ School of Mechanical Engineering, National Technical University of Athens, Athens, Greece

The successful integration of a CAD model in an optimization loop requires an efficient CAD-based optimization methodology. In the field of computational fluid dynamics (CFD), a typical runtime for an industrial component ranges from hours to days on high performance clusters [1]. In this regard, the use of a gradient-based optimization approach which requires a very few iterations to reach an optimum is desirable. However, this requires the gradient of the objective function with respect to the design variables. The typical way to get the derivatives for each design variable is to employ a finite-difference technique [2, 3], where the effect of a parameter change is computed by analyzing both the baseline and perturbed geometries, and comparing the results. For each parameter, a perturbed geometry is created in the CAD system and then used for analysis (including the need for geometry healing and mesh generation processes), where the resulting difference in performance enables the derivative calculation. It is clear this strategy is subject to the so-called “curse of dimensionality”, and quickly becomes impractical for large, high-fidelity models [4].

To address the issue of gradient calculation, adjoint based techniques have shown promising results, and have been an area of extensive research over the last two decades [2, 5–11]. The primary attraction of adjoint methods is their ability to compute gradient information at a computational cost which is essentially independent of the number of design parameters. This, in turn, opens up the possibility to explore significantly larger design spaces than those with traditional approaches, in time-scales which are acceptable for industrial design. Adjoint methods provide the sensitivities of a function of interest with respect to mesh nodal movements. The straightforward route to optimization would be to directly use the mesh nodal displacements as design variables. One drawback for all mesh based approaches is that the mesh topology (in terms of the number of elements present and their connectivity) must remain constant as the model updates. Also, for such approaches the mesh used for the analysis cannot be varied. As it is the mesh that reaches the optimum shape, it must be translated into a CAD model before it can be manufactured. This can be difficult to achieve as mesh nodes are typically positioned independently, and quite often the resulting model shape is not smooth. Using CAD geometry within the optimization process should result in a better quality CAD model shape and aligns with the industrial ambition of having a more integrated design process.

The integration of a parametric CAD model in an adjoint-based optimization framework requires the calculation of parametric design velocity, i.e., the boundary shape movement resulting from a change in a CAD parameter. A number of approaches have been proposed in

the literature for the computation of the design velocity. Chen and Tortorelli [12] used an approach based on the parametric position of points/nodes on the boundary of the unoptimized CAD model. After a parameter perturbation, the new point positions are computed based on the parametric values recorded on the original model. Alternatively, Truong et al. [13] presented an approach for the movement of surface mesh by comparing the parametric definition of surface mesh nodes with respect to tessellation of faces in the original and perturbed CAD model. Hardee et al. [14] applied a hybrid of finite-difference method with boundary displacement method for the computation of design velocity directly from the CAD model. This involved comparing the parametric description of the faces in the original CAD model with the parametric description after the perturbation of one of the design parameters. All of these approaches require that the topology of the geometric model remains constant after the model is perturbed. In the context of this work, the term “constant topology” refers to the number and arrangement of surfaces, edges, and vertices over the model boundary (or B-Rep) remaining the same. Such a constraint is hard to enforce in practice. An example of a boundary topology change is shown in Fig. 1, where Fig. 1a shows the unperturbed CAD model and Fig. 1b shows the same model after a parameter has been perturbed. In this example, the topology change is demonstrated by the introduction of the new shaded faces.

Kripac [17] computed the design velocity from the CAD geometry by assigning certain identities (IDs) to the topological entities in the unperturbed model, and computing velocities from the entities with the same IDs when the model is re-evaluated after a parameter perturbation. This approach is hampered by the persistent naming problem, where the IDs applied to the entities in the CAD model change after it is regenerated after a parameter perturbation. In the cases where IDs are not persistent, techniques are described to rebuild/remap entities based on the construction order of the features in the model, and using adjacency information. An alternative approach to deal with the persistent naming problem is found in [18]. It is unlikely either approach will be robust where the model

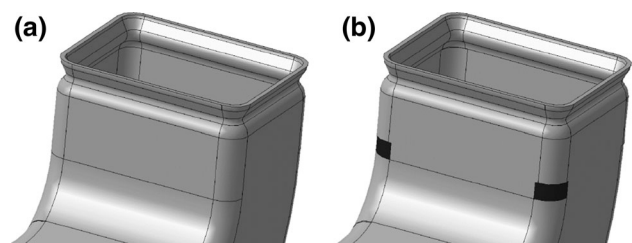


Fig. 1 Topology change after a parameter perturbation where two new faces are created [15]

changes significantly, or where the features or the order of construction change; hence Chen et al. [19] concluded that the persistent naming problem remains unsolved. Nemeč and Aftosmis [16] present an approach for computing the displacement of the boundary based on an embedded boundary Cartesian mesh method, where the movement of the intersections between a Cartesian mesh and a mesh of the component geometry is used to determine the boundary movement. This approach is again restricted to problems where the boundary topology remains constant.

Other authors [20, 21] have attempted to achieve the CAD link through the development of processes based on non-uniform rational B-splines (NURBS), where the NURBS control point locations are the design parameters. Key benefits of these approaches are that NURBS are capable of representing a wide variety of surfaces, and topology changes cannot occur. Authors have also attempted to use free-form deformation (FFD) techniques to parameterize deformation instead of geometry [22, 23]. These techniques originated from computer graphics [24] where FFD boxes are used to deform a solid geometry. The downside of these approaches, compared to a model created in a feature-based CAD system, is that the design intent and parametric associativity captured in the choice of features and parameters used to build the model is lost.

It is possible to calculate the design velocities from the CAD system analytically [25, 26]. Analytical approaches to calculate shape sensitivity have significant advantages, in that they do not require a geometry or mesh to be recomputed, which is both efficient and robust against topology changes. Analytical sensitivities also avoid difficulties with numerical accuracy associated with finite-difference approaches. That said, these approaches are still in the early stages of development and are currently unavailable for general application. Furthermore, they require access to the underlying source code of the CAD system kernel, which is unlikely to become an industrial reality for the major CAD packages in the near future.

To overcome the restrictions associated with topology changes and with the persistent naming problem, Robinson et al. [15] used geometric faceting in the Virtual Reality Modelling Language (VRML) format exported directly from the CAD package. The design velocity was then calculated at the nodes of the faceted model and associated back to the CAD geometry. The key limitation of this approach is that it was not able to calculate design velocity for shape changes which the initial faceting was unable to represent. This can occur when the parameterization allows a face to deform at a much greater resolution than the faceting of the model can recreate. For example, in Fig. 2a, the top face of the block was initially flat and modeled by two facets with nodes at the corners. These facets are

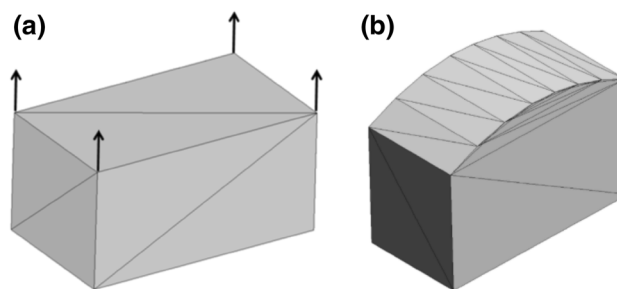


Fig. 2 **a** Top surface represented by two facets with all nodes at surface corners. **b** Modified shape of the top surface not captured by the faceting (design velocities is zero at all nodes)

unable to capture the subsequent curvature of the face, Fig. 2b.

Another example is when the perturbation causes the model to update such that the normal at a point on original model lies outside the perturbed model. In Fig. 3, the symbol “ Δ ” represents facets for which there is no obvious projection after the perturbation shown (from solid to dashed).

This paper builds on the work of Robinson et al. [15], overcomes its aforementioned limitations and applies the approach on a range of models. The remainder of the paper will first summarize the integration of the adjoint-based sensitivity calculation with the design velocity. Sect. 3 presents the methodology proposed to compute the design velocity; the new method is exercised using a turbomachinery static component and a transonic wing, and the corresponding results are shown in Sect. 4; the proposed methodology and results are discussed in Sect. 5; finally the paper terminates with the main conclusions.

2 Theory

2.1 Adjoint methods

The underlying theory and implementation of adjoint methods are well documented in the literature [2, 5–11, 27]. An overview of the approach follows.

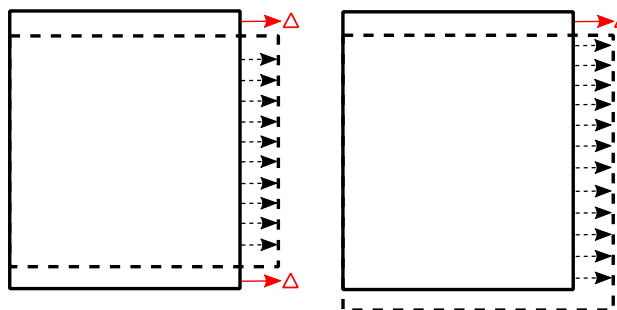


Fig. 3 Geometrical movement when the design velocity fails: original (solid line) and perturbed model (dashed line)

Consider a semi-discrete system of fluid conservation laws described as

$$\frac{d\mathbf{U}}{dt} = \mathbf{R}(\mathbf{U}, \mathbf{X}). \quad (1)$$

Eq. 1 is referred to as the *primal* solution, where \mathbf{X} represents the mesh coordinates and \mathbf{U} is the vector of the fluid system variables. During the convergence of the primal solution, the non-linear residual \mathbf{R} for each equation is driven to zero. The objective function J depends on the system variables.

$$J = J(\mathbf{U}, \mathbf{X}(\theta)). \quad (2)$$

The change in performance dJ due to a change in the value of the design parameter, $d\theta$ can be defined in terms of the surface mesh coordinates \mathbf{X}_s

$$\frac{dJ}{d\theta} = \frac{dJ}{d\mathbf{X}} \frac{d\mathbf{X}}{d\mathbf{X}_s} \frac{d\mathbf{X}_s}{d\theta}. \quad (3)$$

The volumetric sensitivity term $dJ/d\mathbf{X}$ can be obtained by differentiating Eq. 2 with respect to \mathbf{X} as

$$\frac{dJ}{d\mathbf{X}} = \frac{\partial J}{\partial \mathbf{X}} + \frac{\partial J}{\partial \mathbf{U}} \frac{d\mathbf{U}}{d\mathbf{X}}. \quad (4)$$

The solution of Eq. 4 using finite-differences requires the solution of Eq. 1 for each design variable. Alternatively, it can be shown that by selecting an arbitrary vector ψ , Eq. 4 can be re-written as [28]

$$\frac{dJ}{d\mathbf{X}} = \frac{\partial J}{\partial \mathbf{X}} + \psi^T \frac{\partial \mathbf{R}}{\partial \mathbf{X}}. \quad (5)$$

Using this method, only one set of additional equations needs to be solved for each objective function (known as the *adjoint* solution), regardless of the number of design parameters. The adjoint volumetric sensitivities are then combined with the inverse operation of a mesh moving algorithm to yield the adjoint surface sensitivities ϕ as

$$\phi = \frac{dJ}{d\mathbf{X}_s} = \frac{dJ}{d\mathbf{X}} \frac{d\mathbf{X}}{d\mathbf{X}_s}. \quad (6)$$

In recent years, several adjoint solvers have been developed including adjointFoam [3], SU2 [29], HELYX [30], DLR-TAU [31], and HYDRA [32]. These adjoint solvers are capable of producing the sensitivity of a measure of performance with respect to a shape change. The parametric sensitivities $dJ/d\theta$ are then calculated using Eqs. 3 and 6 as

$$\frac{dJ}{d\theta} = \phi \frac{d\mathbf{X}_s}{d\theta}. \quad (7)$$

2.2 Design velocity

Design velocity quantifies the boundary movement with respect to a change in a parameter value $d\theta$, i.e., $d\mathbf{X}_s/d\theta$.

This measure was first developed in the context of adjoint shape sensitivity and optimization in a structural analysis context [12]. In Fig. 4, the arrows represent the design velocity as the boundary changes from solid line to the dashed line. In particular, this paper is concerned with computing the normal component of the design velocity on the boundary of the model as

$$V_n = \delta\mathbf{X}_s \cdot \hat{n}, \quad ((8))$$

where $\delta\mathbf{X}_s$ is the boundary movement and \hat{n} is the surface normal direction. For each location on the domain boundary, the design velocity is represented by a scalar value. The convention adopted throughout this paper is that a positive design velocity represents an outward movement of the boundary, and negative is inward. Once the adjoint sensitivity (ϕ) and design velocity (V_n) are computed, the total change in objective function can be calculated as the summation over the boundary as

$$\Delta J = - \int_A \phi V_n dA. \quad (9)$$

Knowing the change in objective function due to the parametric perturbation in question, the gradient ∇J can then be calculated by normalizing this value with respect to the size of the parameter perturbation used to perturb the boundary as

$$\nabla J = \frac{\Delta J}{\Delta\theta}. \quad (10)$$

3 Methodology

Throughout this work, the feature parameters represent the input, the change in shape of the CAD model is the output, and the CAD modelling system is treated as a black box. The key contribution is the calculation of the design velocity due to the perturbation of the CAD feature parameters. This can then be combined with the adjoint

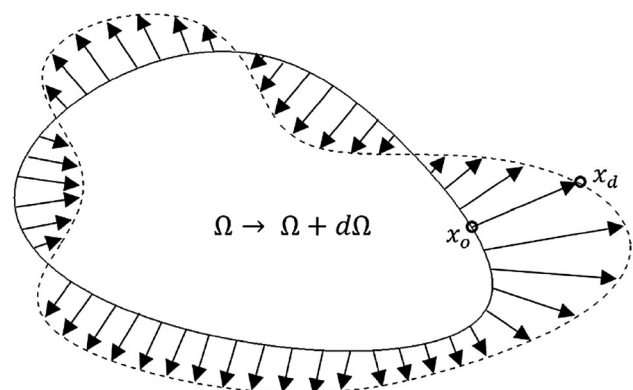


Fig. 4 A two-dimensional design velocity field

sensitivities over the model boundary to calculate the gradient value. Furthermore, this methodology is designed to be applicable to any feature-based CAD modelling package, e.g., SIEMENS NX [33] and CATIA V5 [34], that can cater for any boundary topology changes due to parametric change and avoid the need to access the CAD kernel. This makes it suitable for implementation within an industrial setting, where closed-source and commercial CAD packages are widely used.

The optimization work follows a CAD-centric approach in which the CAD parameters that define the geometric features in the CAD modelling system are considered as design variables. A typical CAD model is shown in Fig. 5, comprising individual features which are combined to represent an overall shape. These features are classified as sketch-based features, which are created by defining a 2D sketch profile and sweeping it to generate a 3D model, and dress-up features, such as fillets and chamfers, which are created directly on the solid model.

As it is common for industrial CAD models to be defined by hundreds (or even thousands) of parameters, using a computationally expensive approach is infeasible for optimization. Herein, the design velocity is calculated using a finite-difference based on the CAD model before and after a parameter perturbation. Such operations performed directly on the CAD geometry are computationally expensive, even for simple CAD models. In this work, CAD geometries are represented using a surface

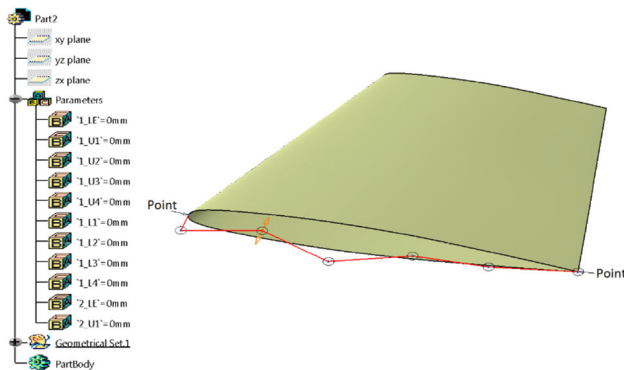


Fig. 5 Feature tree as in CATIA-V5

tessellation of linear triangular elements. This is referred to as faceting to differentiate it from the analysis mesh used to compute the CFD and adjoint solutions. A faceted representation of CAD geometries is created by employing the open-source meshing tool GMSH [35]. The displacement of the model due to a parameter perturbation is approximated by calculating how much a point at the center of each facet in the unperturbed model must move to reside on the boundary of the perturbed model. The key requirement is that the faceting of the unperturbed model is of sufficient resolution to capture the curvatures in the original model, and in each of the perturbations of the model. The first is important as the normal direction over the boundary of the unperturbed model in Eq. (8) is calculated from the facets. The inability to capture the curvatures in the perturbed models hampered the approach in [15], as such sizing information is difficult to gauge a priori. It can be obtained by applying curvature-sensitive meshing to each perturbed model, and ensuring that the maximum mesh density required for a face in any of the perturbed models is reflected in the faceting of the original model. In future work, it will be explored how the curvature in each of the perturbed models can be used to drive a metric-based mesh refinement process for the mesh of the original model. Figure 6b, c shows the coarse and fine surface facet representation of ONERA-M6 CAD model shown in Fig. 6a.

To incorporate various CAD design software tools, a generic representation of the CAD model (the STEP format) is used as the input to the calculation of design velocities. Using the CAD design software, STEP files are created for each parametric perturbation to be used for the calculation of design velocities. The perturbation of a model feature parameter and the export of the corresponding STEP file are automated using the CAD system API. The success of the approach is sensitive to the parameter perturbation size. In this work, in each case the perturbation size used was selected to be small (0.1% – 1%) relative to the size of the features in the model, and different step sizes were experimented with to find the one which gave consistent and accurate results. The algorithm for calculating the design velocity from the resulting STEP files is described in Algorithm 1 and is implemented in Python [36].

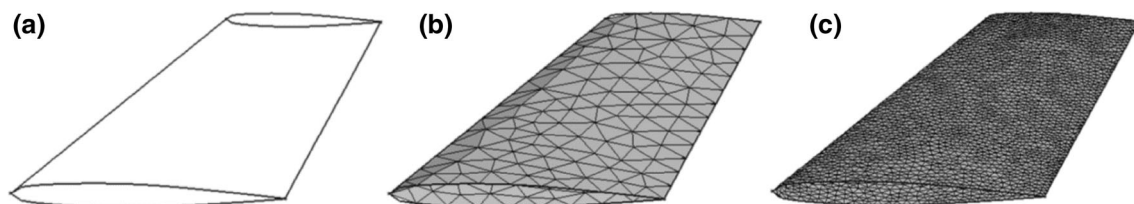


Fig. 6 ONERA M6 a CAD model, b coarse surface facets, and c fine surface facets

Algorithm 1: Design velocity Computation

Input: Parametric CAD model

Output: Design velocity for each CAD parameter

- 1 Python CAD API interacts with CAD modeller to extract parametric information of the model (n parameters).
- 2 Generate STEP file for original model.
- 3 Perturb each parameter by a small value ($\pm\epsilon$) and generate STEP files (n).
- 4 Generate surface facets for each STEP files ($n + 1$) using GMSH.
- 5 Read surface facets for original geometry.
- 6 Compute centroid and normal vector for each surface facet (N facets)
- 7 for *parameter* $i \leftarrow 1$ to n do
 - 8 Read surface facet for i^{th} geometry
 - 9 Compute centroid and normal vector for each surface facet
 - 10 generate KD Tree for each centroid
 - 11 for *facet* $j \leftarrow 1$ to N in *original model* do
 - 12 while *projection unsuccessful* do
 - 13 project j^{th} facet centroid in the normal direction onto facet of i^{th} geometry
 - 14 if *projection is successful* then
 - 15 Design velocity calculated
 - 16 else
 - 17 select next facet candidate

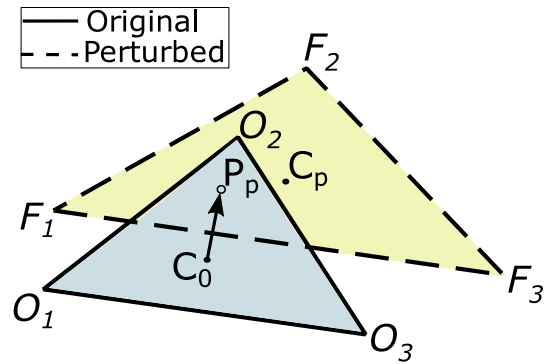


Fig. 7 Projection from unperturbed facet centroid C_0 to perturbed facet with centroid C_p to get the projection point P_p

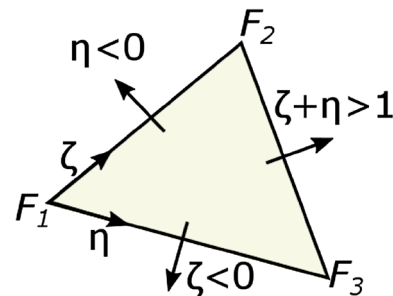


Fig. 8 Using Barycentric coordinates to determine which facet to test next

3.1 The projection test

The displacement of the model due to a parametric perturbation is calculated by projecting a point at the centroid of each facet in the unperturbed model onto the facets in the perturbed model in the normal direction.

In Fig. 7, a facet on an unperturbed model defined as $\Delta O_1O_2O_3$ has its centroid C_0 projected onto the facet $\Delta F_1F_2F_3$ in the perturbed model to find a projection point P_p . The coordinates of P_p are computed using

$$P_p = C_0 + \left\{ \frac{(C_p - C_0) \cdot \hat{n}_{C_p}}{\hat{n}_{C_p} \cdot \hat{n}_{C_0}} \right\} \hat{n}_{C_0}. \tag{11}$$

To determine if a given point P_p lies inside the perturbed facet, consider the Barycentric coordinates [37] shown in Fig. 8. P_p lies inside the perturbed model facet if $\{\zeta > 0; \eta > 0; \zeta + \eta < 1\}$. The Barycentric coordinates ζ, η and ξ are computed using the procedure described by Ericson [37]. If a projection is found to be contained within the boundaries of a triangular facet, then the normal vectors of the facet in the unperturbed and perturbed models are compared to check that they lie within an angular threshold. If both the conditions are satisfied (i.e., the projected point lies within the triangular facet and the surface normal is within the specified tolerance), the projection is deemed

to be successful. If not, then a search is conducted on an adjacent perturbed facet using the same criteria.

3.2 Determining which facet in the perturbed model to test first

One of the goals of this approach is to overcome two limitations: (1) the need for the model’s boundary topology to remain the same before and after a parameter perturbation; (2) the need for the facet labels and their correspondence to the model geometry to guide the projection. This is achieved by projecting the unperturbed facet centroid onto the perturbed facets after each parameter change. To determine which facet on the perturbed model to use in the projection requires a search operation over the facet discretization (as opposed to relying on face labels or boundary topology). This is achieved by setting up a multidimensional binary search tree (KD-tree [38]). For each perturbed model, a KD-tree of its centroid point coordinates is created. A KD-tree query returns for each centroid in the unperturbed model, the closest facet centroid in the perturbed model. The first projection test is performed using this facet. If the projection test is successful, the facet label and the coordinates of the projection point are recorded.

3.3 Determining which facet in the perturbed model to test next

If the projection is unsuccessful for the selected perturbed facet, the Barycentric coordinates are used to determine which facet to use for the next projection test. With reference to Fig. 8, the next facet to test is selected according to

- $\zeta < 0$ the adjacent facet which shares the points F_1 and F_3 should be tested next,
- $\eta < 0$ the adjacent facet which shares the points F_1 and F_2 should be tested next,
- $\zeta + \eta > 1$ the adjacent facet which shares the points F_2 and F_3 should be tested next.

This sequence of projection and facet identification tests should continue until the projection test is successful. If the number of unsuccessful projections reaches a threshold value, a brute force approach is employed. This involves the centroid being projected onto all the facets in the perturbed model and selecting the closest facet which it projects onto with a similar facet normal. To limit the number of facets to test, only a subset of facets within a defined radius from the unperturbed facet centroid is tested. This threshold value depends on the complexity of geometry and also on the surface facet density. Numerical experiments have shown that a radius of twice the length of the maximum parametric perturbation gives adequate results.

It is possible for the brute force approach not to yield a successful projection. This is a possibility in cases where the perturbation causes the model to update such that the direction of the normal vector at a point on original model points outside the perturbed model, as depicted in Fig. 3. In these cases, the nearby facets in the perturbed model are tested. If their surface normal are found to lie within a prescribed tolerance (approximately 5°), the unperturbed facet centroid is projected onto the centroid of that facet and the design velocities in the normal direction are calculated accordingly.

A final condition is introduced if facets of the unperturbed model are still unable to be projected onto the perturbed model. In such cases, the design velocity for these facets is interpolated from those of the neighboring facets in the unperturbed model, which share the common vertex with the original facet.

3.4 Computing design velocity

Once the unperturbed facet centroid (C_0) has been successfully projected in the normal direction (\hat{n}_{C_0}) to obtain the projection point (P_p) on the perturbed model, the design velocity at C_0 is calculated using Eq. (8), which results in

$$V_{n,0} = (P_p - C_0) \cdot \hat{n}_{C_0} \tag{12}$$

It should be noted that neither the face or facet labels, nor the correspondence of the CAD model topology between the perturbed and unperturbed model is necessary to compute the design velocity. Hence, making this strategy more attractive and robust than the current alternatives is discussed in Sect. 1.

4 Results

4.1 Validation of design velocity

To validate the design velocities computed, the methodology described in the preceding section is compared to analytical results calculated for a swept constant section aerofoil with twist as the design parameters. A CAD model of a 3D wing is constructed in CATIA V5 by extruding an aerofoil section defined by Bézier curves as shown in Fig. 9.

The boundary of a 2D section along the wing span is described as

$$P(\zeta) = \sum_{i=0}^n \beta_i B_{i,n}(\zeta). \tag{13}$$

where P is the point on the curve, β_i is the i th control point, and $B_{i,n}(\zeta)$ is the Bernstein polynomial of degree n and $\zeta \in [0, 1]$. The tangential direction at a point on the curve can be calculated by differentiating Eq. 13, giving

$$dP(\zeta) = n \sum_{i=0}^{n-1} B_{i,n-1}(\zeta) (\beta_{i+1} - \beta_i). \tag{14}$$

The normal vector at the point can then be obtained by rotating the normalized tangential vectors as

$$\hat{N}_x = d\hat{P}_x \cdot \cos \frac{\pi}{2} - d\hat{P}_z \cdot \sin \frac{\pi}{2} = -d\hat{P}_z(\zeta), \tag{15}$$

$$\hat{N}_z = d\hat{P}_x \cdot \sin \frac{\pi}{2} + d\hat{P}_z \cdot \cos \frac{\pi}{2} = d\hat{P}_x(\zeta). \tag{16}$$

Considering twist (φ_0) of the wing tip about the Y -axis, which is located along the leading edge, as the design parameter, for a point (x_i, y_i, z_i) on the wing, the coordinate

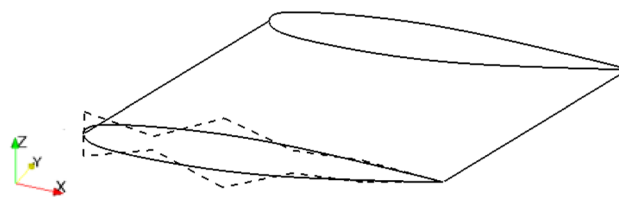


Fig. 9 CAD model of wing with Bézier control points

transformation matrix can be used to obtain the new position (X_r, Y_r, Z_r) as

$$\begin{Bmatrix} X_r \\ Y_r \\ Z_r \end{Bmatrix} = \begin{Bmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{Bmatrix} \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}. \tag{17}$$

φ varies linearly along the wing span according to

$$\varphi_i = \frac{y_i - y_{root}}{y_{tip} - y_{root}} \varphi_0. \tag{18}$$

The derivative of position on the wing body relative to φ is

$$\begin{aligned} dX &= (-x_i * \sin \varphi_i + z_i * \cos \varphi_i) * d\varphi \\ dY &= 0 \\ dZ &= (-x_i * \cos \varphi_i - z_i * \sin \varphi_i) * d\varphi \end{aligned} \tag{19}$$

The design velocity can be calculated using Eqs. (15), (16), and (19) as

$$V_n = dX \cdot \hat{N}_x + dZ \cdot \hat{N}_z. \tag{20}$$

The CAD geometry movement caused by wing twist is shown in Fig. 10, where solid and broken lines represent original and perturbed geometry, respectively. For the design velocity calculation, a perturbation value of 0.5° is used to twist the wing. The surface tessellations are created in GMSH with a total of 126,289 facets using background mesh with refinement boxes. The difference between the design velocities computed using the approach described in this paper and those computed analytically for the wing twist are shown in Fig. 11. It can be seen that the two approaches give results with a maximum difference in the order of 10^{-5} m. The maximum displacement of the boundary caused by this parametric perturbation is in the order of 10^{-3} m.

4.2 Validation of performance gradients

To evaluate the developed approach for industrial geometry, gradients were calculated for a state-of-the-art nozzle guide vane (NGV) of a high pressure turbine (HPT) developed by Rolls-Royce. A 3D CAD model of NGV geometry was built using Siemens NX. The NGV design defines the engine mass flow (and by association the turbine capacity, Q) and is characterised by filleted ends with

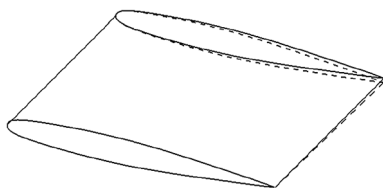


Fig. 10 Comparison of CAD model before and after twist

a cooling slot feature at the trailing edge (TE). In this test case the capacity is considered as the objective function, calculated as

$$Q = \dot{m} \frac{\sqrt{T_t}}{p_t}, \tag{21}$$

where \dot{m} denotes the inlet mass flow, T_t the total temperature, and p_t the total pressure at the inlet using mass averaged values. This geometry is also investigated in [28, 39], where more details about the test case can be found.

Due to the symmetries in the model, the CFD simulation was conducted on one periodic section of the engine’s annulus. Consequently, the sector domain shown in Fig. 12 was used for both CFD and design velocity computation.

The CAD model was created as part of an automated iSIGHT [40] workflow and 12 CAD parameters were analyzed. The parameters were perturbed using the workflow. Figure 13 shows some of the CAD parameters considered in this test case, where SS represents the suction side and PS represents the pressure side of the blade profile.

The primal and adjoint CFD results were obtained using the Rolls-Royce in-house CFD solver HYDRA, solving the steady state RANS equations with the Spalart–Allmaras turbulence model and wall functions, and its corresponding discrete adjoint solver [32]. The nonlinear flow solver uses a node-based finite-volume discretization method, and the pseudo-time-marching to steady state is accelerated by a block-Jacobi pre-conditioner and a geometric multigrid technique. The convergence criteria used required the residual to reduce by nine and five orders of magnitude for the primal and adjoint solutions, respectively.

For each design variable, a new geometry was created with a perturbation step size between 0.1 and 1% of the model size. The mesh for CFD analysis was then automatically created using the BOXER meshing software [41] and contained approximately 9 million nodes and 13 million cells. A typical example of the mesh is given in

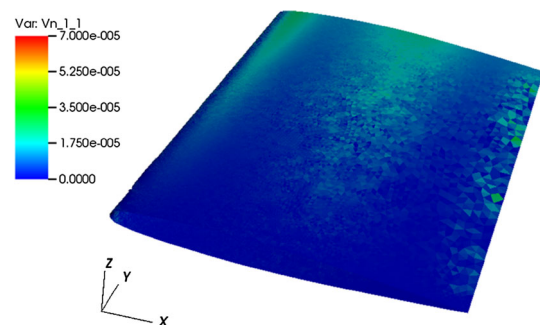


Fig. 11 Comparison of difference between analytical and CAD-based design velocity

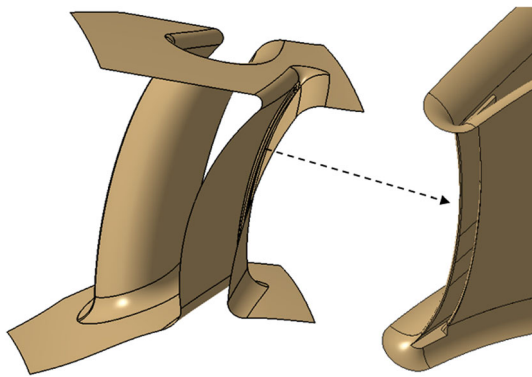


Fig. 12 3D CAD model of NGV geometry in Siemens NX

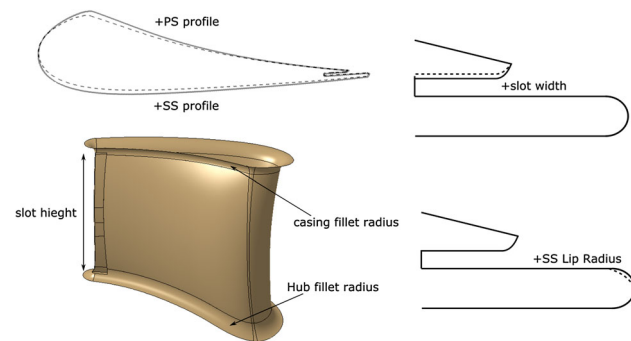


Fig. 13 CAD feature parameters considered as design variables (not to scale)

Fig. 14, including a detailed view of the mesh around the TE.

The adjoint sensitivity map is illustrated in Fig. 15, in which areas of extreme sensitivity are shown in red and blue representing areas where the boundary of the model should be displaced outwards or inwards, respectively, to achieve an increase in objective function. Areas of low sensitivity (shaded in gray) show that the objective function is not sensitive to changes to the boundary in those areas.

For each design variable, a design velocity field was calculated using the approach developed in this work and linked with the adjoint sensitivity maps. The required surface facets are created in GMSH using Delaunay triangulation algorithm with the target element size to be 0.5% of the global model size. The design velocity contours for the casing fillet, SS profile, and hub fillet are shown in Fig. 16. Finally, the change in performance caused by each parametric perturbation is predicted by taking the inner product of the sensitivity map with the corresponding design velocity field, using Eq. 9.

Figure 17 compares the gradients obtained using the adjoint approach and those calculated using finite-differences. All gradients have the same direction and for most parameters the magnitude of the gradients calculated by both

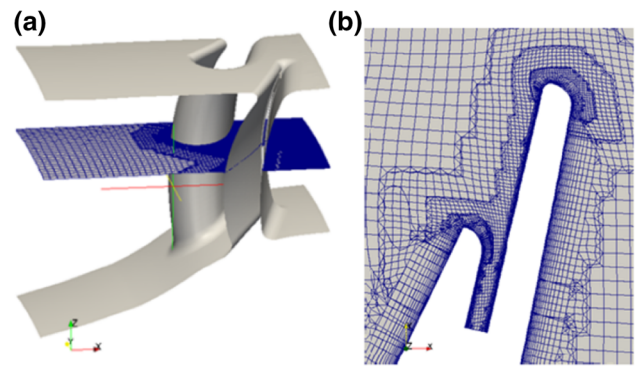


Fig. 14 a NGV CFD domain and b mesh around trailing edge

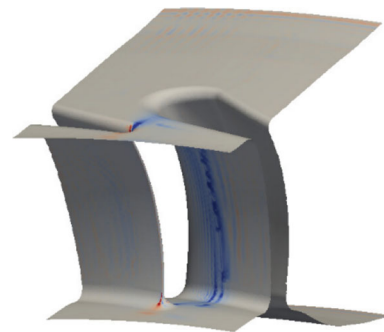


Fig. 15 NGV adjoint sensitivity map

methods are in close agreement. For parameters controlling the TE shape, the magnitude of the finite-difference gradients are consistently lower than the adjoint predictions.

The reason for the differences exhibited for some parameters in Fig. 17 is attributed to:

1. The fact that the parameters were perturbed as part of an industrial workflow and it was not possible to determine if the perturbation size was well chosen for calculating a gradient.
2. The numerical noise in the sensitivity map shown in Fig. 15, especially for parameters which perturb only a small part of the surface in the region of the noise and its effects are magnified (as the ones in the TE slot).

These issues are also investigated using the next test case. It should be noted that even with the differences in the magnitudes of the gradients calculated for some parameters, this is the only approach that can currently calculate sensitivities with respect to CAD parameters that have the same direction for all parameters. The fact that it calculates gradients that are in the correct direction means that it can be used to drive the optimization in industry.

To perform one CFD analysis, it takes approximately 24 h using 20 cores; consequently, the finite-difference approach using the 12 design variables requires approximately 2 weeks of effort. The adjoint approach took

Fig. 16 Design velocity contours for NGV

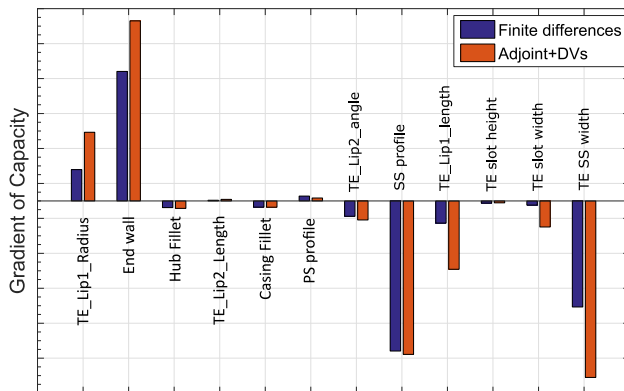
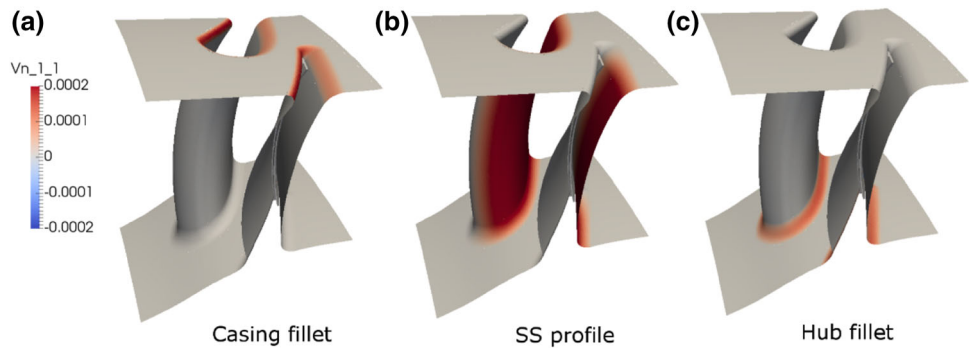


Fig. 17 Validation of gradient of capacity predicted by adjoint results for NGV

2 days, solving 1 primal and 1 adjoint solution. The computation of design velocities is carried out in parallel to the flow analysis and required 45 min for all parameters on a 3.60 GHz workstation with 16 GB RAM.

4.3 Optimization test case

To further test the robustness of the methodology, the application to the shape optimization of a transonic wing is exploited. The aim is to minimize the drag of the ONERA M6 and illustrate the efficacy of the gradient calculation throughout several iterations of the optimization process.

Numerical optimization involves the minimization of a chosen objective function through the manipulation of a set of design variables θ . Within gradient-based optimization methods, the gradient is used to guide the design towards a local optimum over multiple optimization steps. With each new step, a new set of design variables is produced, causing a change in the objective function. A general optimization can be defined as:

$$\begin{aligned} &\text{Minimize: } f(\theta), \\ &\text{Subject to: } g(\theta) \geq 0, \end{aligned}$$

$$h(\theta) = 0$$

where $f(\theta)$, is the objective function to be minimized (maximized), $g(\theta)$ is the inequality constraint, and $h(\theta)$

represents equality constraints. The optimization algorithm used in this work is the “Sequential Least Squares Programming” (SLSQP) implementation in Scipy [42].

For the purpose of optimization, a parametric CAD model for the ONERA M6 was constructed in CATIA V5, using three different cross-sections along the wing span. Each cross-section is defined using two Bézier curves each defined by five points, one defining the upper surface and the other defining the lower surface. The design variables are the z -coordinates of each control point of the Bézier curves, with the following constraints: the leading edge and TE points are fixed, and the first control point on each surface after the leading edge is constrained to move in equal and opposite directions, vertically offset from the leading edge point. This is to preserve $C2$ continuity at the leading edge and gives a total of 27 parameters (3 sections) to be used for optimization. The wing is then constructed by sweeping a surface through the section curves as shown in Fig. 18a.

An unconstrained optimization problem for the following flow conditions is defined:

- Freestream temperature = 288.15 K
- Freestream mach number = 0.8395
- Angle of attack (AoA) = 3.06°
- Objective function = $\min(C_D)$
- No. of design variables = 27

An unstructured mesh was created in GMSH with 154,617 nodes and 707,115 tetrahedral elements; the respective surface mesh is shown in Fig. 18b and used for both flow and adjoint analysis. The mesh density near the leading and TE of the wing are controlled by implementing a background mesh field with refinement boxes. For this problem, the SU2 analysis framework [29] was used to solve the compressible Euler fluid equations and the respective adjoint equations using its continuous adjoint formulation. This required no modification to the gradient calculation method, only the ability to process the output from SU2. SU2 is a finite-volume, node-based solver and has several options to discretize the equations in time and space. For this test case, an implicit Euler method was used

Fig. 18 **a** ONERA M6 CAD model showing Bézier control points for section profiles. **b** CFD mesh

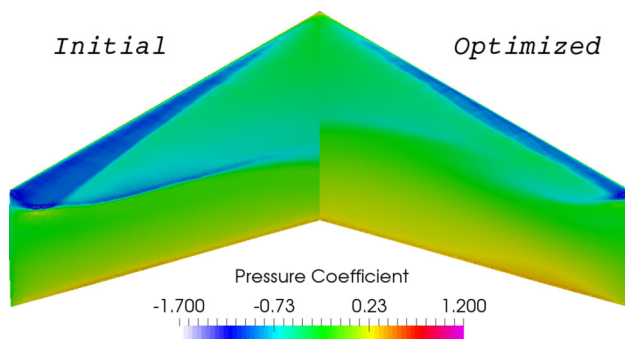
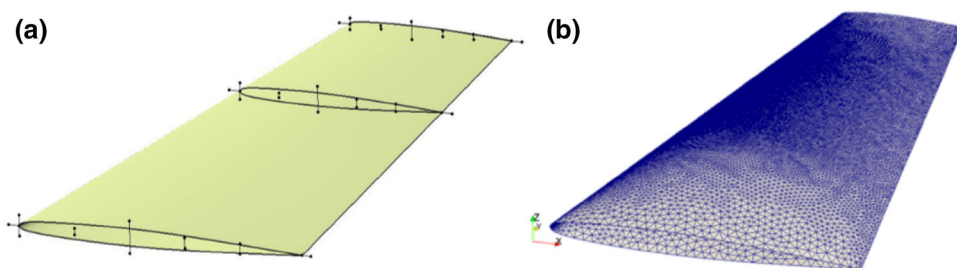


Fig. 19 Pressure contours for initial and optimized ONERA M6

to march the equations forward in time, and the Jameson–Schmidt–Turkel scheme with scalar artificial dissipation is used for the spatial discretization.

The pressure flow field in Fig. 19 shows the formation of shock on the upper surface of the wing, which is similar to results available from the literature [42–45]. The convergence of the residuals of the density and correspondent adjoint-variable equations for the initial wing is shown in Fig. 20.

The perturbed geometries required for the calculation of the design velocity were created using a CAD system API developed in this work. All parameters were perturbed by 1% of the wing chord length, and the surface facets for computation of design velocities are created following the same strategy used for creating the CFD mesh. The generation of 28 STEP files took 21 s, the creation of surface facets in GMSH took 19 min in total, and the design velocity computation was completed in 3.5 min, leading to the total process time of 23 min. The creation of facets was later parallelized (4 cores) to reduce the total process time to 7 min. The contours for six of the parameters controlling the upper surface of the wing are shown in Fig. 21.

Thereafter, the performance gradient with respect to CAD parameters is calculated using Eqs. 9 and 10, and is used in the SLSQP optimization algorithm. The gradients are compared to those calculated using finite-differences in Fig. 22. For each parameter perturbation, a new mesh was generated with sufficient and similar density to maintain the grid independence of the results, hence limiting the distortion of the finite-differences calculation. Overall,

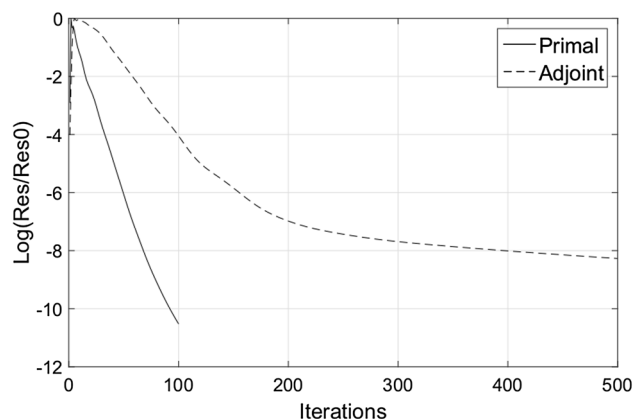


Fig. 20 Residual convergence for initial ONERA M6

both methods are in close agreement for all parameters. The drag coefficient for the wing was reduced from 0.012135 to 0.00303 in 12 optimization steps as illustrated in the optimization history plot in Fig. 23.

A comparison of the pressure coefficient between the initial and optimized geometry at two different cross-sections is shown in Fig. 24. During the optimization process, a reduction of thickness at the leading edge is observed and the point of maximum camber moves slightly aft, resulting in a weakened system of shocks or the elimination of the rear shock, as observed at 60% span. Note that lift is significantly reduced by increasing the aft camber. Both reductions in shock strength and lift contribute to limit the total drag produced.

5 Discussion

The main objective of this work is to present an automated workflow to efficiently calculate the design velocity with respect to the parameters which define the shape of a feature-based CAD model. The design velocity is then linked with adjoint surface sensitivities to output gradients of performance with respect to CAD parameters by the chain rule. The robustness of the developed approach is demonstrated through the application to a turbomachinery component and a 3D transonic wing model. The two models analyzed in this work were created in two different CAD modelling packages

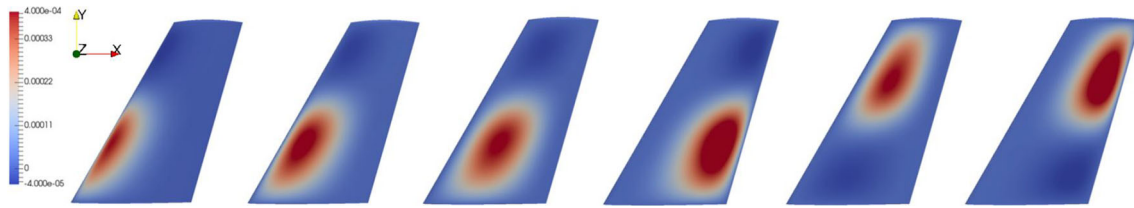
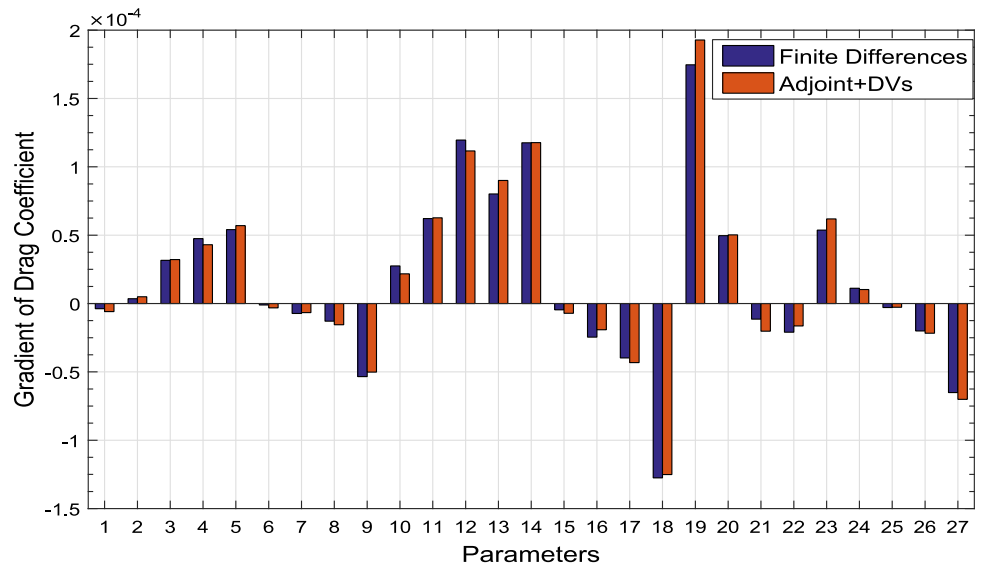


Fig. 21 Design velocity contours for ONERA M6

Fig. 22 Validation of gradient of drag to CAD parameters predicted by adjoint results for ONERA M6 wing



(SIEMENS NX and CATIA V5), and resolved with different CFD solvers (HYDRA and SU2). This substantiates its applicability to industrial CAE systems where models are generally built using commercial CAD packages or other in-house tools, and have different CFD solvers.

The primary benefit of this work is the efficient and robust computation of design velocity for a parametric CAD model built with any CAD modelling package. In terms of computational efficiency, calculating design

velocities is computationally inexpensive and enhances the ability of adjoint methods to reduce the optimization time for industrial size test cases using large design spaces. For each of the examples shown in the paper, the cost of computing the design velocities is small compared to the cost of computing the adjoint sensitivities, and as the proposal is to do this in parallel with the primal/adjoint computation, the cost of including an additional parameter adds no additional time to the optimization loop. That said, for models defined by large numbers of parameters, there will be a point where the cost of computing the design velocities will become greater than the cost of the primal/adjoint computation. At this point, to reduce the computational cost, it is possible to distribute the calculation of design velocities for different parameter sets across different machines. Doing so will require an additional CAD license for each additional machine used; however, one additional license will half the time to compute design velocities for all parameters (and 3 will reduce it to a third, etc.). It is difficult to imagine a scenario where more than a very small number of CAD licenses will be required. There is unlikely to be a scenario where a company is working with a model of such complexity to require parallelization across many machines, but does not have sufficient licenses to allow the parallelization to take place.

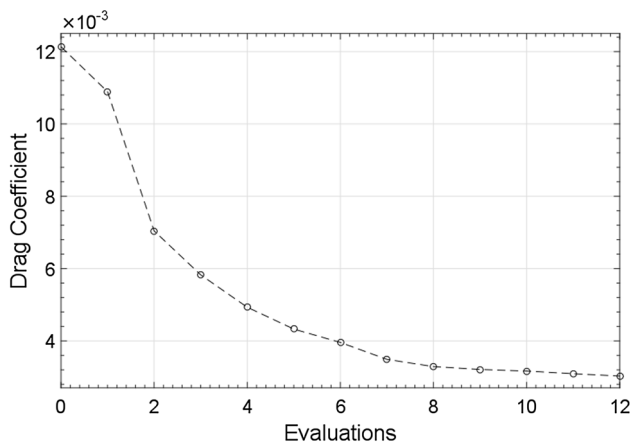


Fig. 23 Optimization history for drag minimization on ONERA M6

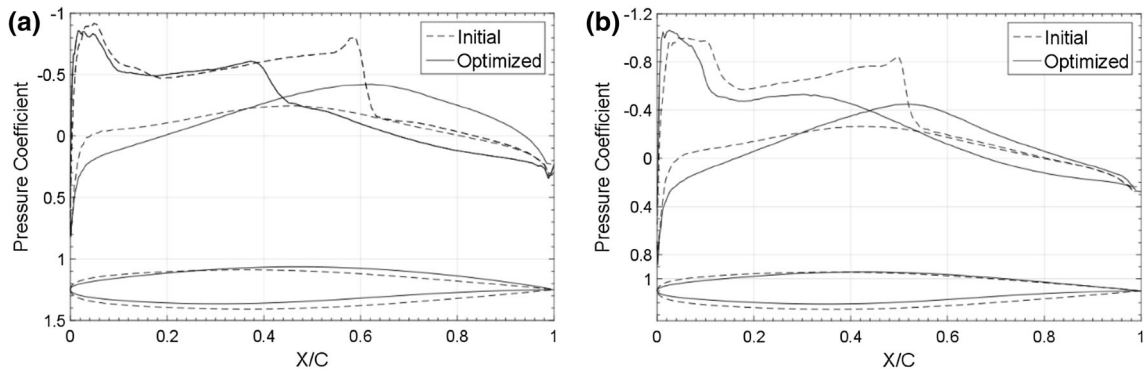


Fig. 24 Pressure coefficient distribution along the wing span: **a** $Y = 0.30$, **b** $Y = 0.60$

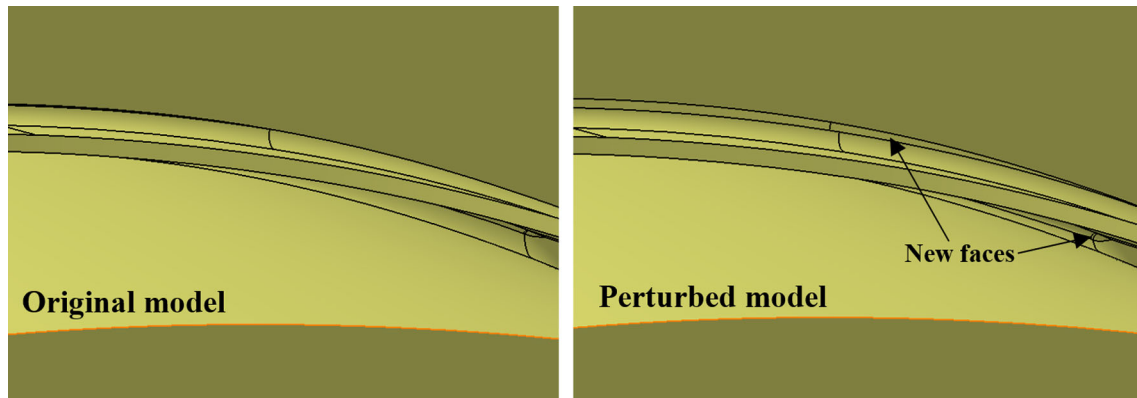
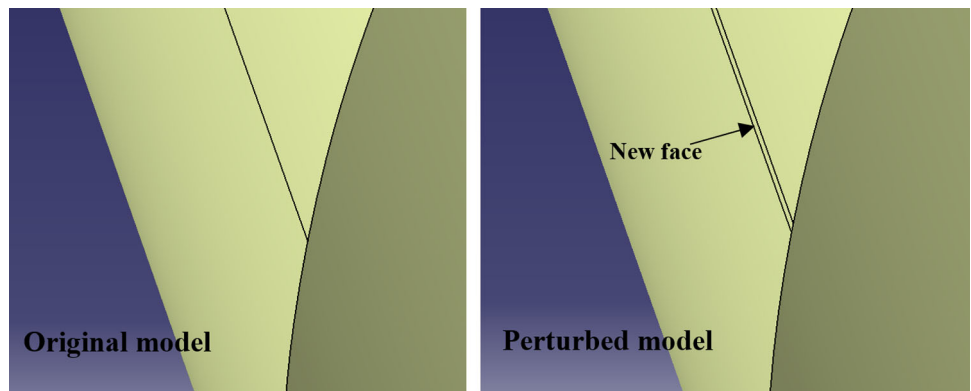


Fig. 25 Parametric perturbation causing appearance of sliver face on NGV model

The proposed approach to compute the design velocity is unaffected by changes in topology which hamper existing approaches, but which are likely to occur during shape optimization of complex models. This was observed for the NGV test case where the parametric perturbations led to the appearance of new sliver faces in the TE slot region (Fig. 25), while during the optimization of the ONERA M6, wing sliver faces appeared near to the leading edge of the wing where the surface curvature is high (Fig. 26).

To facilitate the linkage with different CAD packages, the STEP format of the CAD model is used for each perturbed model. The facets required for the computation of design velocity can be directly generated for the STEP files using a suitable mesh generator. Alternatively, the user can generate a STL (Stereolithography) file and generate the surface facets using a compatible mesh generator, e.g. SnappyHex [46]. The applicability of design velocity for prediction of gradients is given by a combination of Eqs. 8 and 9, which requires design

Fig. 26 Parametric perturbation causing appearance of sliver face on ONERA wing



velocity to be computed accurately in regions of high surface sensitivity. This was achieved using a dense geometrical faceting in these regions which was controlled by surface mesh generators. Future work will be to explore the use of metrics, based on the changes in curvatures caused by changes in the parameters, to adapt the mesh of the original model.

An optimization problem using a transonic wing with 27 design variables was investigated. In this case, a CAD system API was developed to link CATIA V5 with the optimization framework, which automatically updates the CAD parameter values with new values from the optimizer, and exports a new CAD model for the CFD and adjoint calculations. The advantage of using this approach lies in the fact that the optimization is performed directly on the CAD model, and consequently the optimized model is available in the CAD package and can be directly used for other design applications.

6 Conclusion

The following conclusions have been drawn from this work:

- An efficient procedure to calculate performance gradients with respect to CAD parameters, using adjoint methods, was presented.
- The gradients obtained using this approach can be used in an optimization framework to produce an optimized CAD model geometry in a feature-based CAD system.
- The projection methodology using a surface tessellation of CAD geometries overcomes several limitations of alternative approaches, such as the persistent naming problem or changes in the model's topology.

Acknowledgements Authors D. Agarwal and I. Vasilopoulos are PhD researchers within the IODA project (<http://ioda.sems.qmul.ac.uk>), funded by the European Union HORIZON 2020 Framework Programme for Research and Innovation under Grant Agreement No. 642959. The authors would like to thank Rolls-Royce Deutschland for the permission to publish the work. The authors would also like to thank one of the reviewers for some valuable suggestions concerning future work on metric-based faceting.

Compliance with ethical standards

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

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Shahpar S (2011) Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. *Aeronaut J* 115:615–625
2. Mader CA, Martins JRA, Alonso JJ, Der Weide EV (2008) Adjoint: an approach for the rapid development of discrete adjoint solvers. *AIAA J* 46:863–873
3. Othmer C (2008) A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *Int J Numer Methods Fluids* 58:861–877
4. Othmer C, Grahs T (2006) CFD topology and shape optimization with adjoint methods. *VDI BERICHTE* 1967:61
5. Giles MB, Pierce NA (2000) An introduction to the adjoint approach to design. *Flow Turbul Combust* 65:393–415
6. Giles MB, Duta MC, Muller J, Pierce NA (2003) Algorithm developments for discrete adjoint methods. *AIAA J* 41:198–205
7. Jameson A (2003) Aerodynamic shape optimization using the adjoint method. Lectures at the Von Karman Institute, Brussels
8. Reuther J, Alonso JJ, Rimlinger MJ, Jameson A (1999) Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on distributed memory parallel computers. *Comput Fluids* 28:675–700
9. Brezillon J, Gauger NR (2004) 2D and 3D aerodynamic shape optimization using the adjoint approach. *Aerosp Sci Technol* 8:715–727
10. Anderson WK, Venkatakrishnan V (1999) Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Comput Fluids* 28:443–480
11. Roth R, Ulbrich S (2013) A discrete adjoint approach for the optimization of unsteady turbulent flows. *Flow Turbul Combust* 90:763–783
12. Chen S, Tortorelli DA (1997) Three dimensional shape optimization with variational geometry. *Struct Optim* 13:81–94
13. Truong AH, Zingg DW, Haimes R (2016) Surface mesh movement algorithm for computer-aided-design-based aerodynamic shape optimization. *AIAA J* 54:542–556
14. Hardee E, Chang K, Tu J, Choi KK, Grindeanu I, Yu X (1999) A CAD-based design parameterization for shape optimization of elastic solids. *Adv Eng Softw* 30:185–199
15. Robinson TT, Armstrong CG, Chua HS, Othmer C, Grahs T (2012) Optimizing parametrised CAD geometries using sensitivities based on adjoint functions. *Comput Aided Des Appl* 9:253–268
16. Nemeč M, Aftosmis MJ (2008) Adjoint sensitivity computations for an embedded boundary cartesian mesh methods. *J Comput Phys* 227:2724–2742
17. Kripac J (1997) A mechanism for persistently naming topological entities in history based parametric solid models. *Comput-Aided Des* 29:113–122
18. Raghouthama S, Shapiro V (1998) Boundary representation deformation in parametric solid modelling. *ACM Trans Gr* 17:259–286
19. Chen J, Freytag M, Shapiro V (2008) Shape sensitivity of constructively represented geometric models. *Comput Aided Geom Des* 25:470–488
20. Yu G, Müller JD, Jones D (2011) CAD based shape optimization using adjoint sensitivities. *Comput Fluids* 46:512–516
21. Xu S, Jahn W, Muller JD (2014) CAD based shape optimization with CFD using a discrete adjoint. *Int J Numer methods Fluids* 74:153–168
22. Palacios F, Economon TD, Wendorf AD, Alonso JJ (2015) Large-scale aircraft design using SU2. In: 53rd AIAA aerospace sciences meeting, AIAA 2015–1946

23. Hoogervorst JEK, Elham A (2016) Wing aerostructural optimization using the individual discipline feasible architecture. In: 17th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA 2016–3996
24. Sederberg TW, Parry SR (1986) Free-form deformation of solid geometric models. In: 13th annual conference on computer graphics and interactive techniques vol 20, pp 151–160
25. Alonso JJ, Martins J, Reuther JJ, Haimes R, Crawford CA (2003) High-fidelity aero-structural design using a parametric CAD-based model. AIAA 3429:2003
26. Lazzara DS, Drela M, Haimes R (2009) Model sensitivity of edges to a parameter. In: 18th International meshing roundtable research notes, Salt Lake City, UT, USA, 25–28 October. <http://imr.sandia.gov/papers/imr18/Lazzara.pdf>. Accessed 24 July 2017
27. Agarwal D, Marques S, Robinson TT, Armstrong CG, Hewitt P (2017) Aerodynamic Shape Optimization Using Feature based CAD Systems and Adjoint Methods. In: 18th AIAA/ISSMO multidisciplinary analysis and optimization conference: AIAA 2017–3999
28. Vasilopoulos I, Agarwal D, Meyer M, Robinson TT, Armstrong CG (2016) Linking parametric cad with adjoint surface sensitivities. In: ECCOMAS Congress 2016—proceedings of the 7th European congress on computational methods in applied sciences and engineering vol 2, pp 3812–3827
29. Economou TD, Palacios F, Copeland SR, Lukaczyk TW, Alonso JJ (2016) SU2: an open-source suite for multiphysics simulation and design. AIAA J 54:828–846
30. HELYX. <http://engys.com/products/helyx>. Accessed 01/27/2017
31. Schwamborn D, Gerhold T, Heinrich R (2006) The DLR TAU-code: recent applications in research and industry. In: ECCOMAS CFD 2006: proceedings of the European conference on computational fluid dynamics, The Netherlands, 5–8 September 2006
32. HYDRA. <https://www.mpls.ox.ac.uk/research/the-hydra-code-rolls-royces-standard-aerodynamic-design-tool>. Accessed 01/27/2017
33. Siemens NX. https://www.plm.automation.siemens.com/en_gb/products/nx/about-nx-software.shtml. Accessed 01/27/2017
34. CATIA V5. <http://www.3ds.com/products-services/catia/>. Accessed 01/27/2017
35. Geuzaine C, Remacle JF (2009) GMSH: a three dimensional finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer methods Eng 79:1309–1331
36. Python. <https://www.python.org/>. Accessed 01/27/2017
37. Ericson C (2005) Real-time collision detection. Morgan Kaufmann Publishers, Burlington
38. Friedmann JH, Bentley JL, Finkel AR (1977) An algorithm for finding best matches in logarithmic expected time. ACM Trans Math Softw 3:209–226
39. Hogner L, Meyer M, Nasuf A, Voigt P, Voigt M, Vogeler K, Berridge C, Goenaga F (2016) Analysis of high pressure turbine nozzle guide vanes considering geometric variations. In: ASME Turbo Expo GT2016-57502
40. iSIGHT. <https://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/>. Accessed 09/07/2017
41. BOXER. <http://www.cambridgeflowsolutions.com/en/products/boxer-mesh/>. Accessed 01/27/2017
42. Scipy Optimize. <https://docs.scipy.org/doc/scipy/reference/optimize.html>. Accessed 01/27/2017
43. Lyu Z, Kenway GK, Paige C, Martins J (2013) Automatic differentiation adjoint of the Reynolds-averaged Navier–Stokes equations with a turbulence model. In: 21st AIAA computational fluid dynamics conference, AIAA 2013–2581
44. Hewitt P, Marques S, Robinson TT, Agarwal D (2016) Aerodynamic optimization using Adjoint methods and parametric CAD models. In: ECCOMAS congress 2016—proceedings of the 7th European congress on computational methods in applied sciences and engineering
45. Nielsen EJ, Anderson WK (1999) Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. AIAA J 37:1411–1419
46. SnappyHex Mesh. <https://openfoamwiki.net/index.php/SnappyHexMesh>. Accessed 01/27/2017