Parametric Design Velocity Computation for CAD-Based Design Optimization using Adjoint Methods

[1]Dheeraj Agarwal, [1]Trevor T. Robinson, [1]Cecil G. Armstrong, [1]Simão Marques, [2,3]Ilias Vasilopoulos and [2]Marcus Meyer

[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

Abstract This paper presents an efficient and automated 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 a large parameter 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 the 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 CFD 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 a CAD geometry and has to eventually deliver the optimised geometry in CAD. However, currently there is no practical way of either optimising directly on CAD geometries or generating a CAD model from the optimization results. 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 by using either a *dumb* geometry, which is a non-parametric CAD model from which the construction history has been removed, or using a CAD model with all of 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. The main advantage of the parametric approach is that the optimized model produced can be directly used for downstream applications including manufacturing and process planning. Of course, the main disadvantage is that the final design will just be a parametric variation of the initial one rather than a radically different shape.

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 very few iterations to reach an optimum is a desirable option, but this will require the gradient of the objective function with respect to the design variables. The typical way to get the derivatives for each design variables is to employ a finite difference technique [2, 3] where the effect of a parameter change is computed by analysing 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 will be 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 very 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. The main drawback for all mesh based approaches is that the mesh topology must remain constant as the model updates, and that it is the mesh that reaches the optimum shape. Using this approach, it can be difficult to achieve an optimized geometry for manufacture as mesh nodes are typically positioned independently and quite often the resulting model shape is not smooth. For these reasons many researchers have attempted to use CAD geometry within the optimization process, which will result in a better quality 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 Torterelli [12] used an approach based on the parametric position of points/nodes on the boundary of the un-optimized CAD model. After a parameter perturbation the new point positions are computed based on the parametric values recorded on 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 a finite difference method with the 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 work on the condition that the topology of the geometry model remains constant after the model is perturbed i.e. the same arrangement of surfaces, edges and vertices over the model boundary. Such a constraint is hard to enforce in practice. An example of a topology change is shown in Fig. 1, where Fig. 1(a) shows the unperturbed CAD model and Fig. 1(b) 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. Also, for such approaches the mesh used for the analysis cannot be varied.

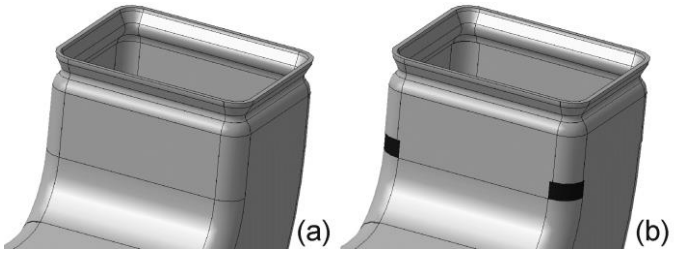


Fig. 1: Topology change after a parameter perturbation where two new faces are created [15]

Kripac [16] 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 ID's 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 Ref. [17]. It is unlikely either approach will be robust where the model changes significantly, or where the features or the order of construction change, hence Chen *et al.* [18] concluded that the persistent naming problem remains unsolved. Nemec and Aftosmis [15] 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 are used to determine the boundary movement. This approach is again restricted to problems where the topology remains constant.

Other authors [19, 20] have attempted to achieve the CAD link through 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 do not occur. Authors have also attempted to use Free-form deformation (FFD) techniques to parameterise deformation instead of geometry [21, 22]. It originated from computer graphics [23] where FFD boxes are used to deform solid geometrical body. 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 when the model was created is lost.

It is possible to calculate the design velocities from the CAD system analytically [24, 25]. 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 specialist access to the underlying 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.* [26] 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 facetted 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 parameterisation allows a face to curve at a much greater resolution than the faceting of the model can recreate. For example in Fig. 2(a) the top face of the block was initially flat and modelled by two facets with nodes at the corners. These facets are unable to capture the subsequent curvature of the face, Fig. 2(b).

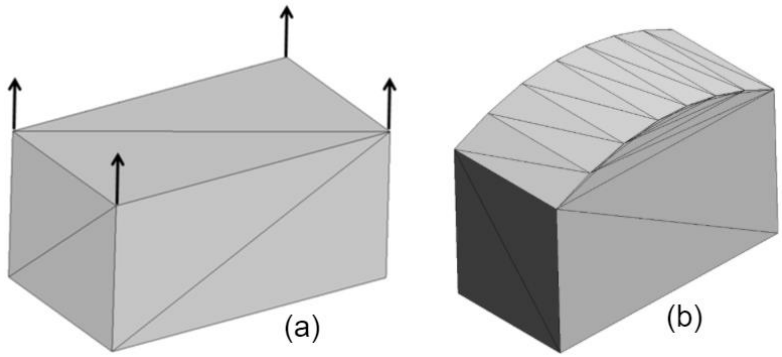


Fig. 2: Top surface represented by two facets with all nodes at surface corners (b) Modified shape of the top surface not captured by the faceting (design velocities is zero at all nodes)

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 dashed to solid).

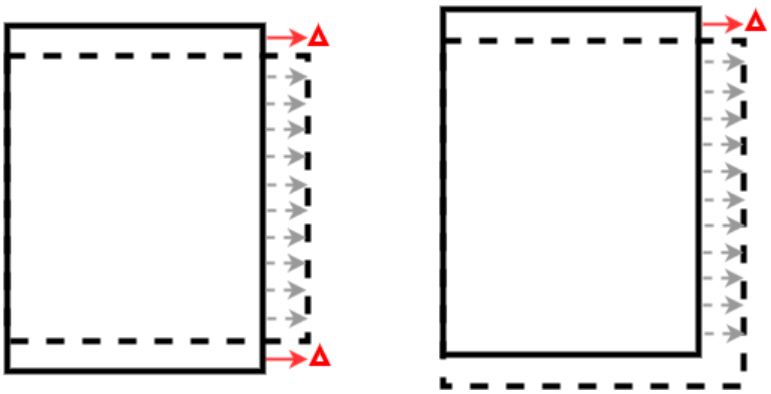


Fig. 3: Geometrical movement when the design velocity fails: original (solid line) & perturbed model (dashed line)

This paper builds on the work of Robinson *et al.* [26], and aims to overcome its aforementioned limitations. The remainder of the paper will first summarize the integration of the adjoint based sensitivity calculation with the design velocity. Section 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 section 4; the proposed methodology and results are discussed in section 5; finally the paper terminates with the main conclusions.

2. Theory

2.1. Adjoint methods

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

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

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

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

$$J = J(\boldsymbol{U}, \boldsymbol{X}(\theta)). \tag{2}$$

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 $\boldsymbol{X}_s$

$$\frac{dJ}{d\theta} = \frac{dJ}{d\boldsymbol{X}} \frac{d\boldsymbol{X}}{d\boldsymbol{X}_s} \frac{d\boldsymbol{X}_s}{d\theta}. \tag{3}$$

The volumetric sensitivity term dJ/dX_s can be obtained by differentiating Eqn.2 with respect to $\boldsymbol{X}$ as

$$\frac{dJ}{d\boldsymbol{X}} = \frac{\partial J}{\partial \boldsymbol{X}} + \frac{\partial J}{\partial U} \frac{dU}{d\boldsymbol{X}}. \tag{4}$$

The solution of Eqn.4 using finite differences requires the solution of Eqn.1 for each design variable. Alternatively, it can be shown that by selecting an arbitrary vector ψ, Eqn.4 can be re-written as [27]

$$\frac{dJ}{d\boldsymbol{X}} = \frac{\partial J}{\partial \boldsymbol{X}} + \psi^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}. \tag{5}$$

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

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

In recent years several adjoint solvers have been developed including adjointFoam [3], SU2 [28], HELYX [29], DLR-TAU [30], HYDRA [31]. 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 Eqn.3 and Eqn.6 as

$$\frac{dJ}{d\theta} = \phi \frac{dX_s}{d\theta}. \quad (7)$$

2.2. Design Velocity

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

$$\boldsymbol{V}_n = \delta \boldsymbol{X}_s . \hat{\boldsymbol{n}}, \quad (8)$$

where $\delta \boldsymbol{X}_s$ is the boundary movement and $\hat{\boldsymbol{n}}$ is the surface normal direction. For each location on the domain boundary the design velocity is represented by a single 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 ($\boldsymbol{V}_n$) are computed, the total change in objective function can be calculated as the summation over the boundary as

$$\Delta \mathrm{J} = -\int_A \phi \boldsymbol{V}_n dA. \quad (9)$$

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

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

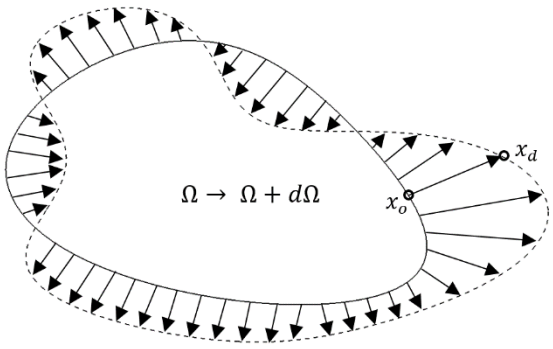


Fig. 4: A two-dimensional design velocity field

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 critical aspect treated here is the calculation of the design velocity due to the perturbation of the CAD feature parameters. This can then be combined with the adjoint 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 [32], CATIA V5 [33], can cater for any boundary topology changes due to parametric change and avoids the need to access the

CAD kernel. Hence, this makes it suitable for implementation within an industrial setting, where closed-source, 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,.

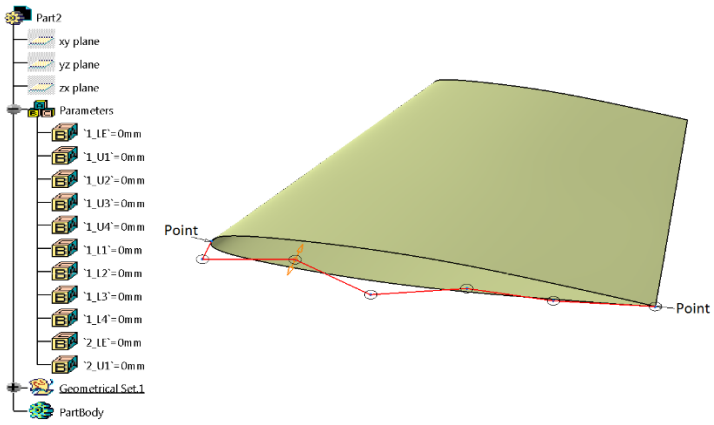


Fig. 5: Feature tree as in CATIA-V5

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 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 [34]. The displacement of the model due to a parameter perturbation is approximated by calculating how much a point at the centre of each facet in the unperturbed model has to move to reside on the boundary of the perturbed model. The required facet density to replicate the CAD model depends on the complexity of the model, its parameterisation and on the curvature discretization by the facets. Fig. 6(b, c) shows the coarse and fine surface facet representation of ONERA-M6 CAD model shown in Fig. 6(a).

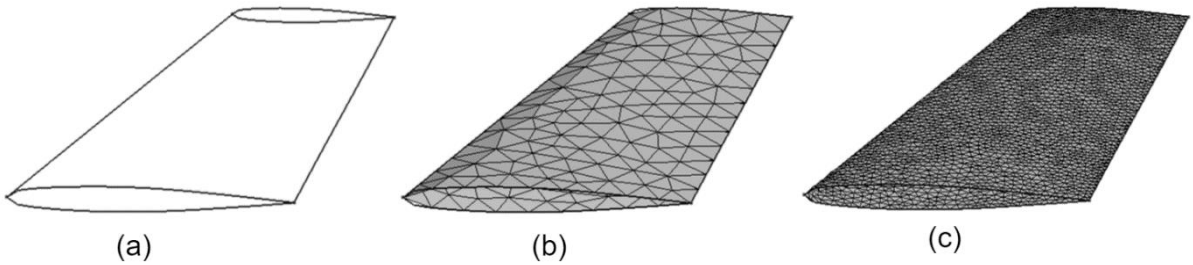


Fig. 6: ONERA M6 (a) CAD model, (b) coarse surface facets, and (c) fine surface facets

To incorporate various CAD design software tools, a generic representation of CAD model (STEP) format is used as the CAD format 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, is automated using the CAD system API. The algorithm for calculating the design velocity from the resulting STEP files is described in Algorithm 1 and is implemented in Python [35].

Algorithm 1: Design velocity Computation

Input: Parametric CAD model
Output: Design velocity for each CAD parameter

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

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.

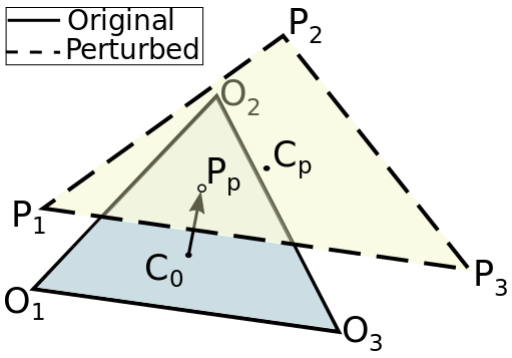


Fig. 7: Projection from unperturbed facet centroid C_0 to perturbed facet with centroid C_p to get the projection point P_p

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:

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

To determine if a given point P_p lies inside the perturbed facet, consider the barycentric coordinates shown in Fig. 8. P_p lies inside the perturbed model facet if $\{\zeta, \eta > 0 \wedge \zeta + \eta < 1\}$. The Barycentric coordinates ζ, η *and* ξ are computed using the procedure described by Ericson [36]. 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.

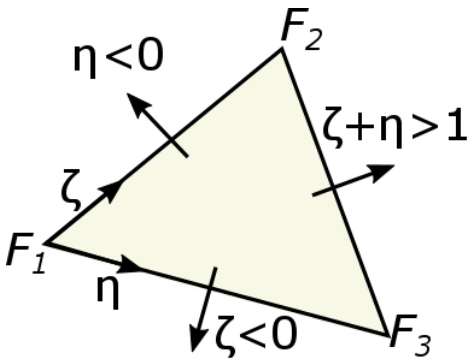


Fig. 8: Using Barycentric coordinates to determine which facet to test next

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 topology to remain constant 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. This is achieved by setting up a multidimensional binary search tree (KD-tree). For each perturbed model, a KD-tree of its centroid point coordinates is created [37]. 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 coordinates of the projection point are recorded.

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 face normal. In order to limit the number of facets to test, only a subset of elements 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 normal at a point on original model lies outside the perturbed model, as depicted in Fig. 3. In these cases, the nearby facets in the perturbed model are tested using the surface normal and if found to lie within a prescribed tolerance (approximately 5°), the unperturbed facet centroid is projected onto the centroid of that element and the design velocities in the normal direction are calculated accordingly.

A final condition is introduced in the event that 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 neighbouring facets in the unperturbed model which share the common vertex with the original facet.

Computing design velocity

Once the unperturbed facet centroid (C_0) is successfully projected in the normal direction ($\hat{\boldsymbol{n}}_{C_0}$) to obtain the projection point (P_p), the design velocity at C_0 is calculated using Eqn.8, which results in

$$\boldsymbol{V}_{n,0} = (P_p - C_0).\hat{\boldsymbol{n}}_{C_0} \quad (12)$$

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 are necessary to compute the design velocity. Hence, making this strategy more attractive and robust than the current alternatives discussed in the Introduction section.

4. Results

4.1. Validation of Design Velocity

To validate the accuracy of the calculation of the design velocities, the methodology described in the preceding section is compared against analytical results for the parameterisation of a plate test case. The plate was created by extruding a profile created using three straight lines (bottom and sides) and a B-Spline along the top. The spline was controlled using five control points, evenly distributed along the top edge as shown in Fig. 9.

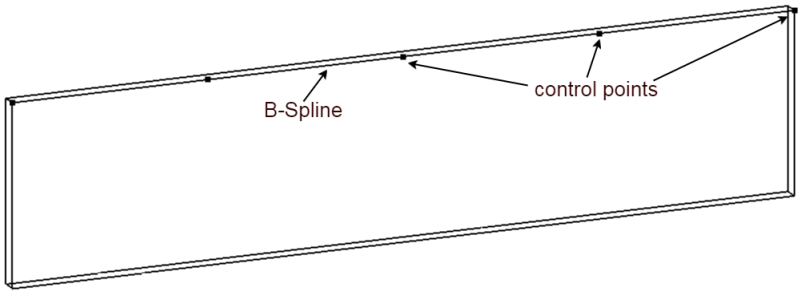


Fig. 9: Plate model with B-Spline control points

The purpose of this simple parameterisation is that it allows the analytical calculation of design velocities based on the mathematical description of a Bezier curve

$$X(\varsigma) = \sum_{i=0}^{n} \beta_i B_{in}(\varsigma), \quad (13)$$

where X is the point on the curve, β_i is the i^{th} control point, and $B_{in}(\varsigma)$ is the Bernstein polynomial of degree n and $\varsigma \in [0,1]$. Using Eqn.13, the z-displacement can be calculated at each surface mesh node for a perturbation of the control points. A series of design velocity fields were calculated for the CAD model by perturbing each control point in z-direction. A vector plot of the design velocity field corresponding to the movement of a control point at the centre of B-Spline is shown in Fig. 10.

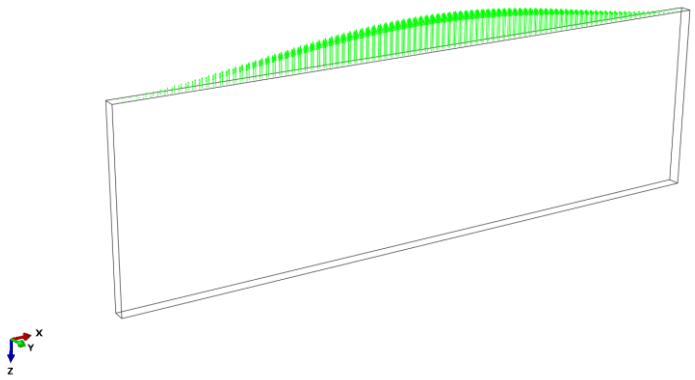


Fig. 10: Design velocity vectors for parameter perturbation of +1mm

The design velocity field computed using the CAD-based approach is compared to the analytical solution for the movement of two control points and shown in Fig. 11. It can be seen that the two approaches give identical results, with the maximum error of the order of $10^{-7}m$ for a perturbation of $10^{-3}m$.

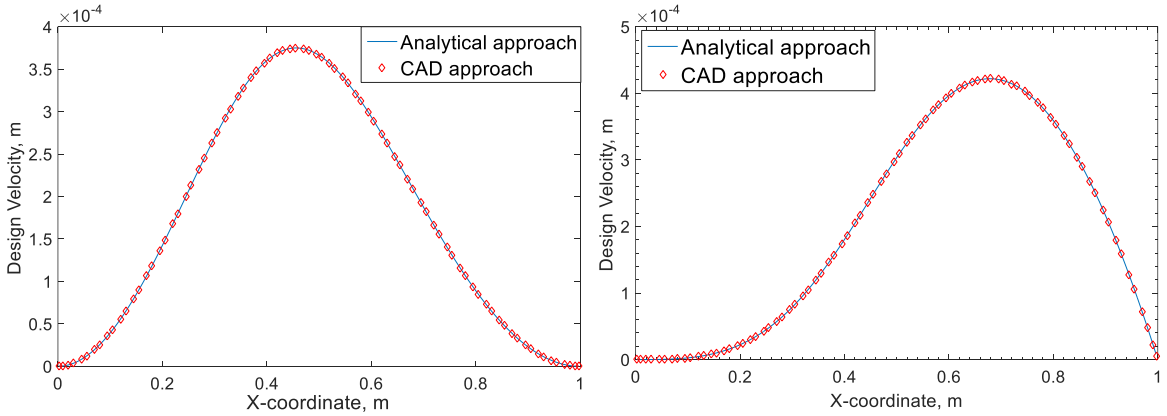


Fig. 11: Comparison between analytical and CAD based design velocity (CAD results plotted for every other point): (a) X=0.5; (b) X=0.75;

4.2. Validation of performance gradients

In order 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 parametric CAD model of NGV geometry was built using Siemens NX. The NGV has fillets at both ends and a cooling slot feature at the trailing edge (TE). This geometry is also investigated in [38], which illustrates different methods in order to quantify the impact of geometric variations on the NGV performance with focus on the capacity. Typically, the engine mass flow (and by association the turbine capacity, Q) is governed by the NGV design, which defines the narrowest cross section of the turbine. As a result, in this test case the capacity is considered as the objective function. After convergence, the capacity at the inlet is calculated as

$$\mathrm{Q} = \dot{m}\frac{\sqrt{T_t}}{p_t}\,, \tag{14}$$

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. The resulting capacity for the baseline geometry was found to match the expected capacity value and is taken as the reference value for all subsequent comparisons with perturbed geometries [38].

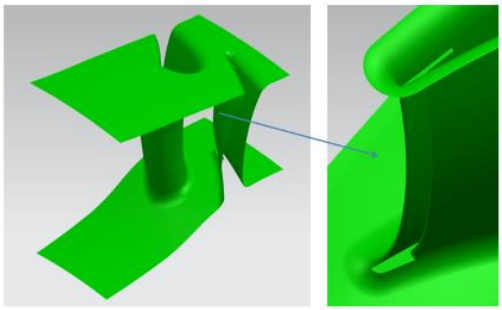


Fig. 12: 3D CAD model of NGV geometry in Siemens NX

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. Further details on the test case and results are given by Vasilopoulos [27].

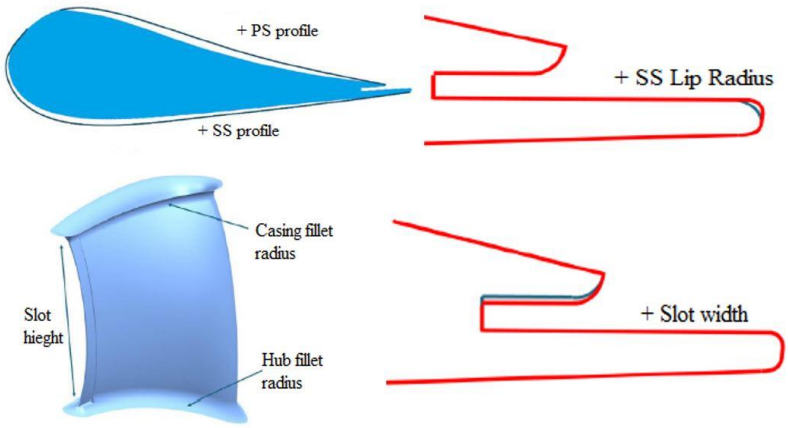
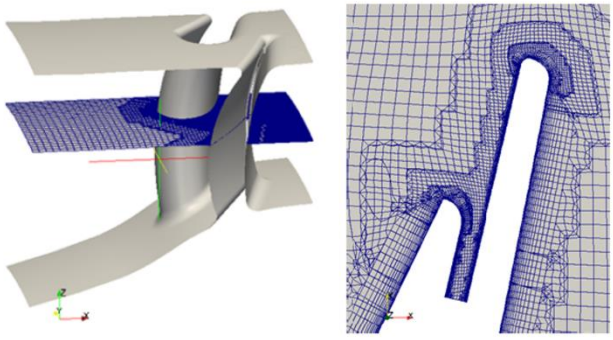


Fig. 13: CAD feature parameters considered as design variables (not to scale)

A total of twelve CAD parameters were used in this problem. The parameters were perturbed and associated geometries created using an iSIGHT workflow. Fig. 13 shows some of the CAD parameters considered in this test case, where *SS* represents the suction side and *PS* represent 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 [31]. The nonlinear flow solver uses a node-based finite-volume discretisation 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.

The mesh for each new geometry was created automatically using the BOXER meshing software [39] and contained approximately 9 million nodes and 13 million cells. A typical example of the mesh is given in Fig. 14, including a detailed view of the mesh around the trailing edge.



(a) (b)

Fig. 14: (a) NGV CFD domain, and (b) mesh around trailing edge

The adjoint sensitivity map is illustrated in Fig. 15 where 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 grey) show that the objective function is insensitive to changes to the boundary in those areas.

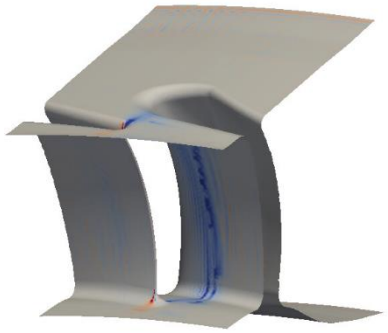


Fig. 15: NGV Adjoint sensitivity map

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 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 Eqn. 9.

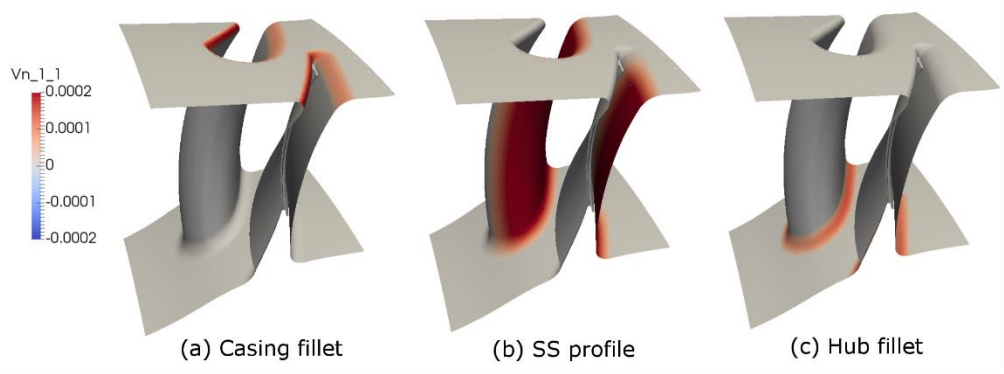


Fig. 16: Design Velocity contours for NGV

Fig. 17 compares the gradients obtained using the adjoint and finite-differences approaches. All gradients have the same direction and for most parameters both methods are in close agreement in terms of magnitude. For parameters controlling the trailing edge shape, the magnitude of the finite-difference gradients are consistently lower than the adjoint predictions. These differences may be related to the step size used in the finite-differences, or in accuracies in the adjoint solution near the TE discontinuity. Further investigation is required to obtain a definitive answer.

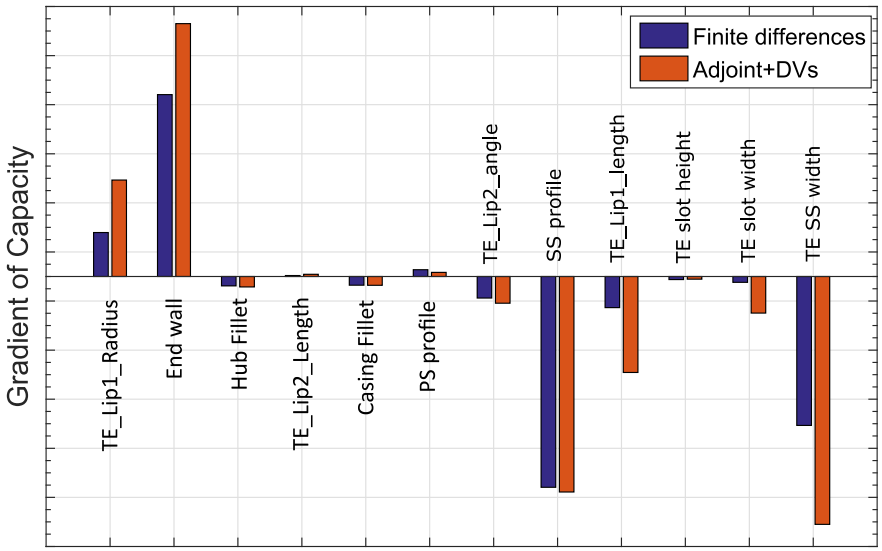


Fig. 17: Validation of gradient of capacity predicted by adjoint results

To perform one CFD analysis takes approximately 24 hours using 20 cores, consequently the finite difference approach using the 12 design variables requires approximately two weeks of effort. The design velocity approach took two days, solving one primal and one adjoint solution. The computation of design velocities is carried out in parallel to the flow analysis and takes 45 minutes for all parameters on a 3.60GHz workstation with 16GB 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 the 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. 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:

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

where $f(\theta)$, is the objective function to be minimised (maximised), $g(\theta)$ 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 [40].

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

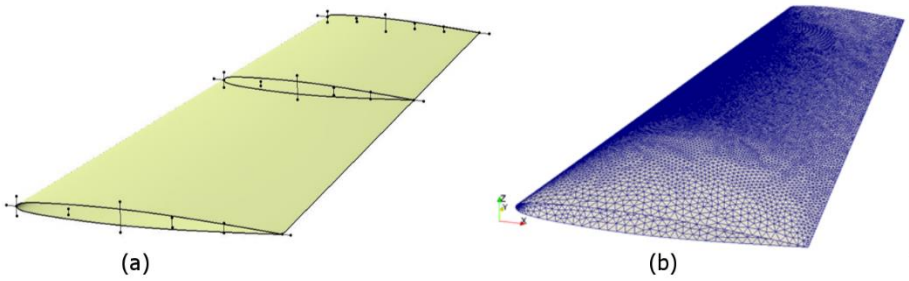


Fig. 18: (a) ONERA M6 CAD model showing Bezier control points for section profiles. (b) CFD mesh

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. 18(b) and used for both flow and adjoint analysis. For this problem the SU2 analysis framework [28] 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 to march the equations forward in time and the Jameson-Schmidt-Turkel scheme with scalar artificial dissipation is used for the spatial discretization.

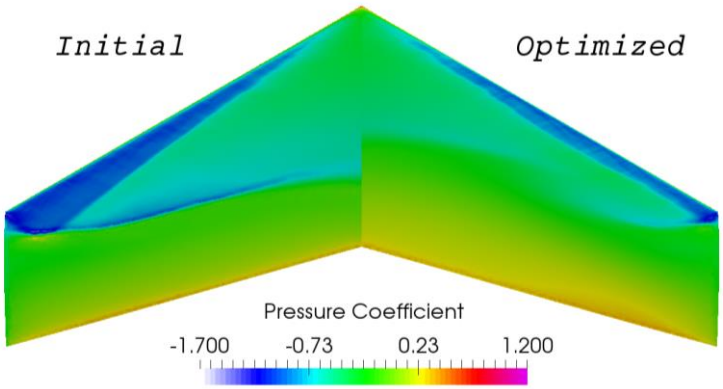


Fig. 19: Pressure contours for initial and optimized ONERA M6

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 [41-43]. The convergence of the residuals of the density and correspondent adjoint variable equations for the initial wing is shown in Fig. 20.

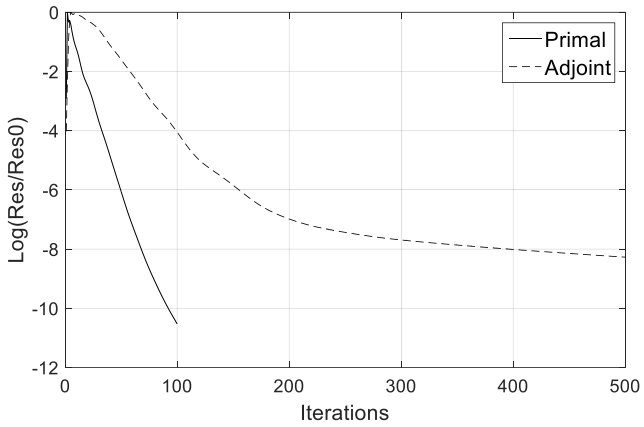


Fig. 20: Residual convergence for initial ONERA M6

The perturbed geometries required for the calculation of the design velocity was created using a CAD system API developed in this work. The design velocity contours for six of the parameters controlling the upper surface of the wing is shown in Fig. 21.

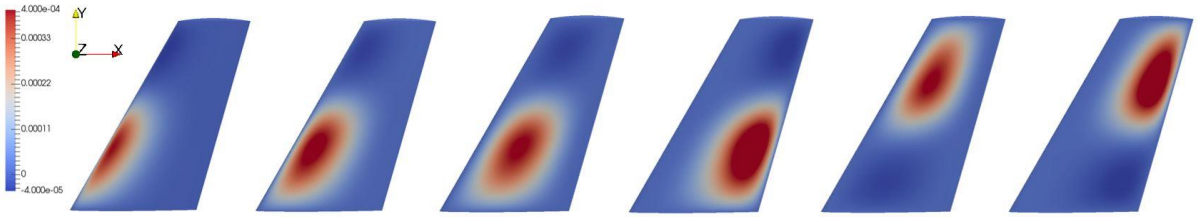


Fig. 21: Design Velocity contours for ONERA M6

Thereafter, the performance gradient with respect to CAD parameters is calculated using Eqn. 9 and Eqn. 10, and are used in the SLSQP optimization algorithm. 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. 22.

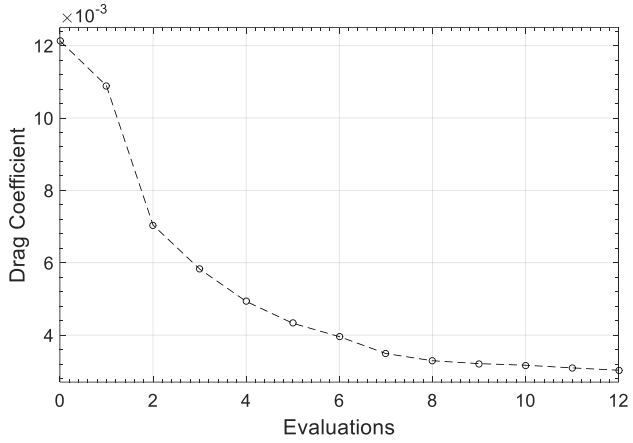


Fig. 22: Optimization history for drag minimization on ONERA M6

A comparison of the pressure coefficient between the initial and optimized geometry at two different cross-sections is shown in Fig. 23. 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 as well that the lift is significantly reduced by increasing the aft camber. Both reductions in shock strength and lift contribute to limit the total drag produced.

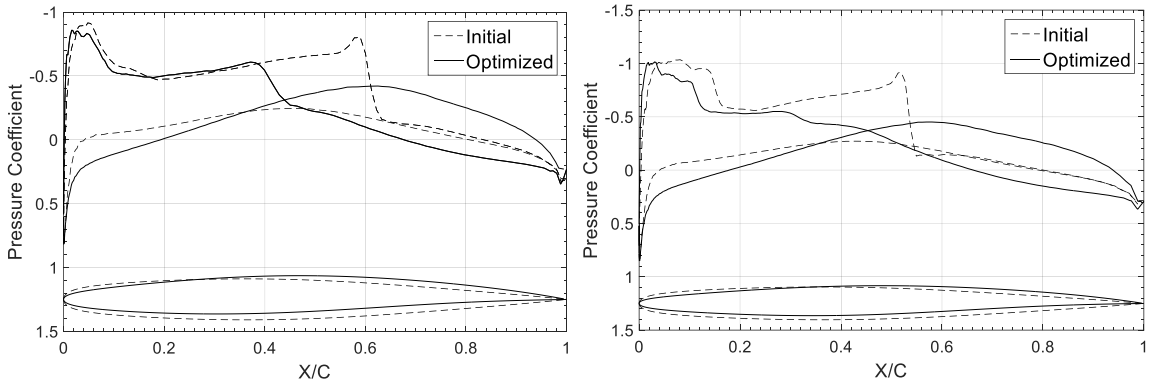


Fig. 23: C_p distribution along the wing span: (a) Y=0.30, (b) Y=0.60

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 analysed in this work were created in two different CAD modelling packages, (SIEMENS NX and CATIA V5), and resolved with different CFD solvers (HYDRA and SU2). This substantiates its applicability to industrial Computer Aided Engineering (CAE) systems where models which 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 parameter spaces. Furthermore, the proposed approach to compute the design velocity is unaffected by changes in topology which hamper alternative 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 lead to the appearance of new sliver faces in the trailing edge slot region, 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.

In order 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 using STEP files by 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 [44] (which does not support STEP files). The applicability of design velocity for prediction of gradients is given by a combination of Eqn. 8 and Eqn. 9, which requires design velocity to be computed accurately in regions of high surface sensitivity. This was achieved by using a dense geometrical faceting in these regions which was controlled by surface mesh generators.

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.

Acknowledgement

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.

References

1. Shahpar S (2011) Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. The Aeronautical Journal 115:615-625.

2. Mader CA, RA Martins J, 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, Turbulence and Combustion 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. Computers and Fluids 28:675-700.

9. Brezillon J, Gauger NR (2004) 2D and 3D aerodynamic shape optimization using the adjoint approach. Aerospace Science and Technology 8:715-727.

10. Anderson WK, Venkatakrishnan V (1999) Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. Computers & Fluids 28:443-480.

11. Roth R, Ulbrich S (2013) A discrete adjoint approach for the optimization of unsteady turbulent flows. Flow, Turbulence and Combustion 90:763-783.

12. Chen S, Torterelli DA (1997) Three dimensional shape optimization with variational geometry. Struct. Opt. 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, Changb K, Tua J, Choia KK, Grindeanua I, Yu X (1999) A CAD-based design parameterization for shape optimization of elastic solids. Advances in Engineering Softwares 30:185-199.

15. Nemec M, Aftosmis MJ (2008) Adjoint sensitivity computations for an embedded boundary cartesian mesh methods. Journal of Computational Physics 227:2724-2742.

16. Kripac J (1995) A mechanism for persistently naming topological entities in history based parametric solid models. In: 3rd ACM symposium on Solid modelling and application

17. Raghothama S, Shapiro V (1998) Boundary representation deformation in parametric solid modelling. In: ACM transactions on Graphics

18. Chen J, Freytag M, Shapiro V (2008) Shape Sensitivity of Constructively Represented Geometric Models. Comput. Aided Geom. Des. 25:470-488.

19. Yu G, Müller JD, Jones D (2011) CAD based shape optimization using adjoint sensitivities. Computers and Fluids 46:512-516.

20. Xu S, Jahn W, Muller JD (2014) CAD based shape optimization with CFD using a discrete adjoint. International Journal of Numerical methods in Fluids 74:153-168.

21. Palacios F, Economon TD, Wendorf AD, Alonso JJ (2015) Large-scale aircraft design using SU2. In: 53rd AIAA Aerospace Sciences Meeting

22. Hoogervorst JEK, Elham A (2016) Wing aerostructural optimization using the Individual discipline feasible architecture. In: 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

23. Sederberg TW, Parry SR (1986) Free-Form Deformation of Solid Geometric Models. In: 13th annual conference on Computer graphics and interactive techniques20: 151-160.

24. Alonso J, Martins J, Reuther J, Haimes R, Crawford C (2003) High-Fidelity Aero-Structural Design Using a Parametric CAD-Based model. In: 16th AIAA Computational Fluid Dynamics Conference

25. Lazzara DS, Drela M, Haimes R (2009) Model Sensitivity of Edges to a Parameter. In: 18th International Meshing Roundtable Research Notes

26. Robinson TT, Armstrong CG, Chua HS, Othmer C, Grahs T (2012) Optimizing parametrised CAD geometries using sensitivities based on adjoint functions. Computer Aided Design and Application 9:253-268.

27. 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 Engineering2: 3812-3827.

28. Economon 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.

29. HELYX. http://engys.com/products/helyx. Accessed 01/27 2017

30. 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, Egmond aan Zee, The Netherlands, September 5-8, 2006

31. HYDRA. https://www.mpls.ox.ac.uk/research/the-hydra-code-rolls-royces-standard-aerodynamic-design-tool. Accessed 01/27

32. Siemens NX. https://www.plm.automation.siemens.com/en_gb/products/nx/about-nx-software.shtml. Accessed 01/27 2017

33. CATIA V5. http://www.3ds.com/products-services/catia/. Accessed 01/27 2017

34. Geuzaine C, Remacle JF (2009) GMSH: a three dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal of Numerical methods in Engineering 79:1309-1331.

35. Python. https://www.python.org/. Accessed 01/27 2017

36. C. E (2005) Real-time collision detection. Morgan Kaufmann Publishers,

37. Friedmann JH,Bentley J.L., Finkel A.R. (1977) An algorithm for finding best matches in. ACM Transactions on Mathematical Software (TOMS) 3:209.

38. 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

39. BOXER. http://www.cambridgeflowsolutions.com/en/products/boxer-mesh/. Accessed 01/27 2017

40. Scipy Optimize. https://docs.scipy.org/doc/scipy/reference/optimize.html. Accessed 01/27 2017

41. Lyu Z, Kenway GKW, 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

42. 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

43. Nielsen EJ, Anderson WK (1999) Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. AIAA J. 37:1411-1419.

44. SnappyHex Mesh. https://openfoamwiki.net/index.php/SnappyHexMesh. Accessed 01/27 2017