
Article:
Mura, G.L., Hinchliffe, B.L., Qin, N. orcid.org/0000-0002-6437-9027 et al. (1 more author) (2018) Nonconsistent mesh movement and sensitivity calculation on adjoint aerodynamic optimization. AIAA Journal, 56 (4). pp. 1541-1553. ISSN 0001-1452

https://doi.org/10.2514/1.J055904

Non-Consistent Mesh Movement and Sensitivity Calculation on Adjoint Aerodynamic Optimization

Gabriele Luigi Mura[1], Benjamin Lee Hinchliffe[2], and Ning Qin[3]

Department of Mechanical Engineering, University of Sheffield, Sheffield S1 3JD, UK

Joël Brezillon

Airbus Operations SAS, 316 route de Bayonne, 31060 Toulouse Cedex 09, France

This paper presents an investigation of the influence of a non-consistent approach in terms of mesh movement and mesh sensitivity calculation in a discrete adjoint-based optimization. Some mesh movement methods are more robust or of higher quality, while the others can be more efficient for calculating mesh sensitivity. It is found that a non-consistent approach gives comparable results when compared to a consistent approach. Therefore an appropriate combination of non-consistent approaches can be achieved for efficient adjoint optimisation. This paper investigates and compares various consistent and non-consistent combinations by using linear elasticity, Delaunay graph mapping and radial basis function mesh movement methods. An investigation, using a lift-constrained drag minimization, to assess which step of the chain introduces a deviation, if any, and to which degree this affects the final result is presented.

Nomenclature

$\{\boldsymbol{B}\}$ = Vector of the Delaunay vertices

$C_{D,L}$ = Drag and lift coefficient

[1] Ph.D. Candidate, University of Sheffield, AIAA Student Member
[2] Ph.D. Candidate, University of Sheffield, AIAA Student Member
[3] Corresponding author, n.qin@sheffield.ac.uk, AIAA Associate Fellow

$\{\boldsymbol{D}\}$ = Vector of the shape design variables

$[\boldsymbol{E}]$ = Delaunay volume ratio coefficient ratio matrix

I = Cost function

$[\boldsymbol{K}]$ = Linear elasticity stiffness matrix

$\mathcal{L}$ = Lagrangian operator

$\{\boldsymbol{S}\}$ = Surface mesh

$\{\boldsymbol{R}\}$ = Flow discretized residual vector

$\{\boldsymbol{T}\}$ = Mesh movement residual vector

$\{\boldsymbol{W}\}$ = State variables vector

$\{\boldsymbol{X}\}$ = Volume mesh

$\{\boldsymbol{\Lambda}_F\}$ = Flow-adjoint vector

$\{\boldsymbol{\Lambda}_G\}$ = Mesh-adjoint vector

$\{\boldsymbol{\Lambda}_G\}_{DGM}$ = Mesh-adjoint vector w.r.t. the Delaunay graph method

$\{\boldsymbol{\Lambda}_G\}_{LE}$ = Mesh-adjoint vector w.r.t. the linear elasticity

Introduction

IN aerodynamic shape optimization, the discrete adjoint method is the useful tool to optimize the entire lifting surface with very large number of design variables [1, 2]. An adjoint-based aerodynamic shape optimization involves the differentiation of the entire chain, i.e. flow, mesh and shape derivatives. By doing so, and assuming the shape has been parametrized, three types of sensitivity are needed, namely flow, grid and shape sensitivity. In this work, the flow adjoint is calculated using the discrete formulation where the discrete adjoint equation is derived after the discretisation of the governing equations [1, 2]. This paper focuses on using non-consistent methods for both the mesh movement and grid sensitivity parts of the optimisation chain.

There are two types of grid sensitivity based on the distinction between volume and surface mesh. The volumetric mesh sensitivity is the sensitivity of the volume mesh, i.e. $\{\boldsymbol{X}\}$, w.r.t. (with respect to) the surface mesh, $\{\boldsymbol{S}\}$. On the other hand, the surface mesh sensitivity is the sensitivity of the parametrised surface mesh w.r.t. the parametric DVs (Design Variables). The mesh sensitivity is represented by a large matrix of dimensions that scale with the $N_{VM} \times N_{SM}$ (number of volume mesh points times number of surface mesh points)

and is related to the mesh movement. The surface sensitivity defined above is represented by a matrix of dimension $N_{SM} \times N_{DV}$ (where N_{DV} is the number of DVs). The analytical availability of this derivative depends on the choice of the shape parameterization which is generally easily differentiable.

The computation of the gradient of the surface mesh w.r.t. the geometric DVs, i.e. $[\partial \boldsymbol{S}/\partial \boldsymbol{D}]$, can be obtained with ease, however the gradient of the volume mesh w.r.t. the surface mesh, i.e. $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$, shows a complex dependency which is dictated by the mesh movement to be differentiated. Morris et al. [3] proposed to use finite differences for $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$, whereas Hicken and Zingg [4] used the more sophisticated complex differences in conjunction with the mesh-adjoint approach by Nielsen and Park [5].

While working with the discrete approach the volumetric mesh term cannot be eliminated. This means that, when large grids are considered, the computation of the volumetric term, as proven by Mura et al. [6], is in general expensive both in terms of CPU time and memory requirements. These issues still persist (although not at the same time) even when the easy-to-apply finite differences and its more accurate counterpart, i.e. complex differences, are used.

There are several methods available in the literature to compute the grid sensitivity: finite differences, complex differences, analytical and adjoint methods. This classification can also be further split between methods that apply either to explicit or implicit mesh movements [6]. Explicit methods are defined as those where the volume mesh update is available without inverting the mesh deformation matrix which maps the link between surface and volume mesh nodes movement, such as for the DGM (Delaunay Graph Mapping) [7]. Implicit methods are defined as those where the same volume mesh update requires the inversion of the mesh deformation matrix, such as for the LE (Linear Elasticity) [8], RBF (Radial Basis Functions) [9] and spring analogy [10]. The mesh-adjoint method [5] can be applied to any implicit and iterative mesh movement, whereas for any explicit non-iterative strategy, the linearized grid sensitivity is available explicitly by definition. Both techniques provide the sensitivity at a cost nearly independent from the N_{DV} (the slope of the line of cost versus N_{DV} is very shallow).

However, Mura et al. [6] noted that even if the direct dependency from the N_{DV} is eliminated, any explicit non-iterative mesh update strategy, such as the DGM for instance, provides the grid sensitivity at a cost which is far less expensive in terms of memory and CPU time over the mesh-adjoint approach.

This paper presents an investigation of the influence of a non-consistent approach in terms of mesh movement and mesh sensitivity calculation in a discrete adjoint-based optimization loop. This is interesting because, should

any large difference be highlighted, this would mean that one cannot choose the non-consistent approach without incurring in a large deviation w.r.t. the fully consistent approach. On the other hand, in absence of appreciable differences, one would be allowed to choose based on efficiency or robustness grounds.

Some applications in the literature have been reported in Tab.1. There are different approaches that have been investigated as standalone cases, but a quantitative comparison of what happens when different combinations of mesh movement and mesh sensitivity are considered has not been published yet.

Table 1 Mesh movement and mesh sensitivity approaches used in the literature.

Authors	Mesh movement	Approach used to calculate the mesh sensitivity	Consistency
Nemec and Zingg [11] Martins et al. [12]	Algebraic method	Finite differences	Consistent
Morris et al. [3]	RBF	Finite differences	Consistent
Zhu and Qin [13] Hinchliffe and Qin [14] Bobrowski et al.[15]	RBF	Mesh-adjoint on LE	Non-consistent
Jakobsson and Amoignon [16]	RBF	Direct approach	Consistent
Truong et al. [17] Nielsen and Park [5] Nambu et al. [18]	LE	Mesh-adjoint on LE	Consistent
Hicken and Zingg [4]	LE	Complex differences	Consistent
Mavriplis [19] Maute et al. [20]	Linear spring analogy (LSA)	Mesh-adjoint on LSA Direct approach	Consistent
Burgreen and Baysal [21] Le Moigne and Qin [2]	Algebraic method	Analytical	Consistent

I Sensitivity Chain in Adjoint Approach

The DLR TAU-Code [22] is a fully parallelised code solving the unsteady Navier-Stokes equations. Space discretisation is performed using the finite volume method. The resulting system of discretised equations is solved using the preconditioned LU-SGS (Symmetric Gauss Seidel) linear solver and a 3W-type geometric multigrid is used to speed up the solution. Its discrete flow-adjoint version is used along with the one-equation

SA turbulence model [23]. The flow solutions presented in this work are obtained using a steady-state RANS time averaging solution. The SA is used in its fully linearised version, i.e. no frozen turbulent viscosity, in order to maintain consistency between the primal and the dual solutions.

Two Lagrangian multipliers are constructed to obtain the augmented objective function, $\mathcal{L}$, namely one for the non-linear flow residual constraint, $\{\boldsymbol{R}\}$ and the other for the linear mesh movement residual constraint, $\{\boldsymbol{T}\}$:

$$\mathcal{L}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X},\boldsymbol{\Lambda}_F,\boldsymbol{\Lambda}_G) = I(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X}) + \{\boldsymbol{\Lambda}_F\}^T\{\boldsymbol{R}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X})\} + \{\boldsymbol{\Lambda}_G\}^T\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\}, \qquad \forall\{\boldsymbol{\Lambda}_F\},\{\boldsymbol{\Lambda}_G\} \tag{1.1}$$

where $\{\boldsymbol{\Lambda}_F\}$ and $\{\boldsymbol{\Lambda}_G\}$ are the flow and mesh-adjoint vector respectively. Differentiating the augmented cost function w.r.t the DVs and considering only pure geometric changes yields to:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{W}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right\}\left[\frac{d\boldsymbol{W}}{d\boldsymbol{D}}\right] + \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_G^T\frac{d\boldsymbol{T}}{d\boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] + \{\boldsymbol{\Lambda}_G\}^T\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{1.2}$$

For a pure aerodynamic shape optimization, where only the solid wall is updated, the DVs influence only the flow-field solution and objective function through the grid variations. As a consequence $\{\partial I/\partial \boldsymbol{D}\} = \{\boldsymbol{0}\}$ and $[\partial \boldsymbol{R}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$. Since Eq. (1.1) is valid for all the DVs, the Lagrangian is identical to the original objective function, $\mathcal{L} = I$. There are two expensive terms in Eq. (1.2). The first one is the sensitivity of the flow variables, $\{\boldsymbol{W}\}$, w.r.t. the DVs, $\{\boldsymbol{D}\}$, whereas the second one is the sensitivity of the volume mesh nodes, w.r.t. the DVs. The latter becomes increasingly expensive as large unstructured meshes are considered. After solving the flow-adjoint and if necessary also the mesh-adjoint linear system of equations, the objective function gradient can be written as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \{\boldsymbol{\Lambda}_G\}^T\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{1.3}$$

From Eq. (1.3), it is clear that the differentiation of the mesh movement residual is necessary in the discrete adjoint approach. However, this can happen in a consistent or non-consistent manner w.r.t. the chosen mesh movement. Furthermore, it is useful to differentiate Eq. (1.1) w.r.t. the volume mesh which yields:

$$\left\{\frac{dI}{d\boldsymbol{X}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\} \tag{1.4}$$

where both the objective function and the residual contains the pressure and the viscous part. To better understand the physical meaning of these terms, each partial derivative is analyzed separately. The first RHS

(Right Hand Side) term expresses the variation of the target function due to the variation of the surface (or line in 2D) of integration at constant pressure and skin friction coefficient [24]. The second RHS side term expresses the variation of the target function due to the variation of the pressure and skin friction coefficient, both induced by a variation of the geometry [24].

Therefore, substituting Eq. (1.4) back in Eq. (1.2) yields to:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{dI}{d\boldsymbol{X}}\right\}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{1.5}$$

where is it important to note that within the scope investigated in this paper, the only element that is changing is the second RHS as it depends on the mesh movement employed.

Mathematically, a consistent approach is preferred because the same method is used which saves implementation time and secondly because consistency in the chain is maintained. However non-consistent approaches may be practically more useful because different types of mesh sensitivity could be used based on accuracy, robustness or efficiency grounds. This paper explores the magnitude and effect of any errors which are introduced by using a non-consistent approach.

Before the non-consistent approach can be used, some research questions need to be addressed. It is important to find whether non-consistent approaches lead to different optima as compared with the consistent approaches. This work investigates whether these deviations (if present) are due to different grid sensitivities or different mesh movements. To address these points, it is important to analyze each step of the chain as follows.

- Different mesh movements: LE, DGM and RBF
- Different mesh movement linearizations: $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ and $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{DGM}$
- Product between $\{dI/d\boldsymbol{X}\}$ and $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ and the distribution of each factor over the computational domain

For each one of these points the mesh, i.e. $\{\boldsymbol{X}\}$, is involved and should not be considered asa standalone factor, but an integrated part of the optimization chain.

II Methods for Mesh Movements and Mesh Sensitivities

At each iteration of an aerodynamic optimization loop there is first a change in the parametrized surface mesh followed by a change in the volume mesh.

LE [8], RBF [9] and DGM [7] are some of the options available to deform the mesh. The last one is an algebraic explicit and non-iterative method, whereas the first two are implicit and iterative methods. One of the most important feature is that both DGM and LE are capable of providing the gradient $\{dI/d\boldsymbol{S}\}$ which expresses the sensitivity of the objective function w.r.t. all the surface mesh. On the other hand, the RBF (at least in most of its applications) can only provides the gradient w.r.t. the RBF interpolation control points [15]. The RBF could in theory provide the gradient $\{dI/d\boldsymbol{S}\}$ (see Eq. (1.5) for $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$), but the cost of inverting its constitutive matrix would be prohibitive, as explained earlier.

The DGM creates a one-to-one explicit map between surface and volume meshes, which can be written as a linear system:

$$\{\boldsymbol{X}\} = [\boldsymbol{E}]\{\boldsymbol{B}\} \tag{2.1}$$

where matrix $[\boldsymbol{E}]$ contains all the volume ratio coefficients computed as shown by Liu et al. [7] and $\{\boldsymbol{B}\}$ is the vector containing the Delaunay boundary vertices. This generally comprises of points from different boundaries, i.e. $\{\boldsymbol{B}\} = \{\boldsymbol{B}_{ff}, \boldsymbol{B}_{sw}, \boldsymbol{B}_{sb}\}$ where $\{\boldsymbol{B}_{sw}\}$ represents the Delaunay vertices that coincide with the solid wall mesh nodes, thus $\{\boldsymbol{B}_{sw}\} \equiv \{\boldsymbol{S}\}$, $\{\boldsymbol{B}_{sb}\}$ are the supporting box points used to improve the quality of the Delaunay map, and $\{\boldsymbol{B}_{ff}\}$ represents the Delaunay vertices that coincide with the far-field surface mesh nodes. Differentiating Eq. (2.1) w.r.t. the design variables, $\{\boldsymbol{D}\}$ yields [6]:

$$\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] = [\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.2}$$

where by implication:

$$[\boldsymbol{E}] \equiv \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right] \tag{2.3}$$

Note that the farfield and the supporting box points are not being deformed, therefore $[\partial \boldsymbol{B}_{ff}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$ and $[\partial \boldsymbol{B}_{sb}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$. Furthermore, $[\partial \boldsymbol{B}_{sw}/\partial \boldsymbol{D}] = [\partial \boldsymbol{S}/\partial \boldsymbol{D}]$ because all the surface mesh points are used as Delaunay vertices. In order to improve the quality of the Delaunay triangulation a supporting box around the

wing can be constructed extracting some suitable volume mesh points as shown by Mura et al. [25]. There are two requirements that must be considered in this process. The first requirement is to place the supporting box at a distance that does not interfere with the surface deformation. The second requirement is to consider all the volume mesh nodes where the gradient $\{dI/d\boldsymbol{X}\}$ has significant influence. Using the supporting box means the metric terms are not differentiated outside the region within the solid wall and supporting box. Furthermore, the Delaunay boundaries with the addition of the supporting box's points are now represented by $\{\boldsymbol{B}\} = \{\boldsymbol{B}_{ff}, \boldsymbol{B}_{sw}, \boldsymbol{B}_{sb}\}$, where $\{\boldsymbol{B}_{sb}\}$ is the Delaunay vertices that coincide with the supporting box nodes. However, from the point of view of the differentiation it is clear that $[\partial \boldsymbol{B}_{sb}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$. When the supporting box is used the DGM is referred to as mDGM otherwise the DGM is referred to as oDGM.

The expenses in terms of memory and CPU time associated with the computation of the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ have prompted researchers to study ways of reducing it. The study published by Nielsen and Anderson [26], aimed at establishing the ideal reduction in the volume mesh nodes differentiation. They concluded that the influence of the mesh sensitivity gradually decays away from the wall and that accurate results are obtained if the points within the wall and half of the distance from wall to the far-field are included. In this paper, the same methodology is repeated for the mDGM in order to understand the effects of using a reduced volume mesh differentiation.

The LE method creates an implicit map between the surface and the volume mesh, which can be written in matrix-vector form as:

$$[\boldsymbol{K}]\{\boldsymbol{X}\} = \{\boldsymbol{S}\} \tag{2.4}$$

where $[\boldsymbol{K}]$ is the mesh deformation matrix. Note that, unlike Eq. (2.1), the matrix is on the left hand side of Eq. (2.4) and therefore the equation is implicit w.r.t. $\{\boldsymbol{X}\}$.

Eq. (2.4) is then cast in FEM form and discretised using a Galerkin method. The resulting linear system of equations is then solved using a restarted ILU (Incomplete Lower Upper) preconditioned GMRES (Generalised Minimal RESidual) strategy with a drop in the residual of 14 orders of magnitude [27]. Differentiating Eq. (2.4) w.r.t. the geometric design variables, $\{\boldsymbol{D}\}$ yields [6]:

$$[\boldsymbol{K}]\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.5}$$

From Eq. (2.5), it follows that $[\partial \boldsymbol{X}/\partial \boldsymbol{S}] = [\boldsymbol{K}]^{-1}$. As proven by Mura et al. [6], the iterative inversion of the elastic matrix $[\boldsymbol{K}]$ is costly both in terms of memory and in CPU time. This expense is comparable with the cost of a single steady-state flow solution, thus it make sense to address it. Similarly, the RBF mesh movement method creates an implicit map between the surface and the volume mesh nodes, which can be written in a compact matrix-vector form as:

$$[\boldsymbol{M}]\{\boldsymbol{X}\} = \{\boldsymbol{S}\} \tag{2.6}$$

where $[\boldsymbol{M}]$ is a matrix containing the interpolating RBF coefficients. The extent of the volume metric differentiation stored in this matrix is governed by the RBF interpolating points chosen a priori. The linearization of Eq. (2.6) w.r.t. the DVs reads as:

$$[\boldsymbol{M}]\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.7}$$

Rendall and Allen [28] noted that for a direct mesh movement, the cost of RBF scales with $N_{SM} \times N_{VM}$. This was an observation made by the authors on a structured mesh. It goes without saying that, for unstructured mesh this issue would be more severe, making their approach unfeasible. Based on this observation, they developed an alternative method that iteratively selects, based on the minimization of an error function, fewer interpolation points at the wall, i.e. N_{IP}. Therefore, since $N_{IP} \ll N_{SM}$, their method results in a consistent saving both in memory and computational time. Based on this observation, Eq. (2.7) becomes:

$$[\boldsymbol{M}]\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}_{IP}}\right] = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}_{IP}}\right] \tag{2.8}$$

Having said this, Eq. (2.8) provides the sensitivity w.r.t. the RBF control points only [15, 29]. Therefore, Eq. (2.8) is unable to provide the gradient of the objective function w.r.t. each surface mesh point (i.e. $\{dI/d\boldsymbol{S}\}$), which provides useful design information as shown by Hinchliffe and Qin [14].

Fig. 1 shows the resulting RBF, LE, oDGM and mDGM-based deformed meshes for the same arbitrary surface mesh update. While using the oDGM, it can be clearly seen that the deformed volume mesh nodes closely follow the Delaunay computational domain decomposition (constructed using all the solid wall and the far-field points). The mDGM follows the same upwards direction as the oDGM, but the deformation is truncated at where the supporting box points are located. Regarding LE and RBF, it is interesting to see how the deformation is global

compared to the local DGM-deformed mesh. This can be explained by the fact that the o/mDGM are by definition based on an anisotropic operator (i.e., it follows the shape of the original Delaunay triangulation as shown in Ref. [6, 25]), whereas both LE and RBF, at least for the implementation used in this work, are based on an isotropic operator. However, it is recognised that both LE and RBF can be made anisotropic if deemed necessary.

Having analyzed the mesh movements and their differentiated versions, the next step is to study the differentiation of the mesh movement constraint in more details. From Eq. (1.3), the differentiation of the mesh deformation residual is required. This reads as:

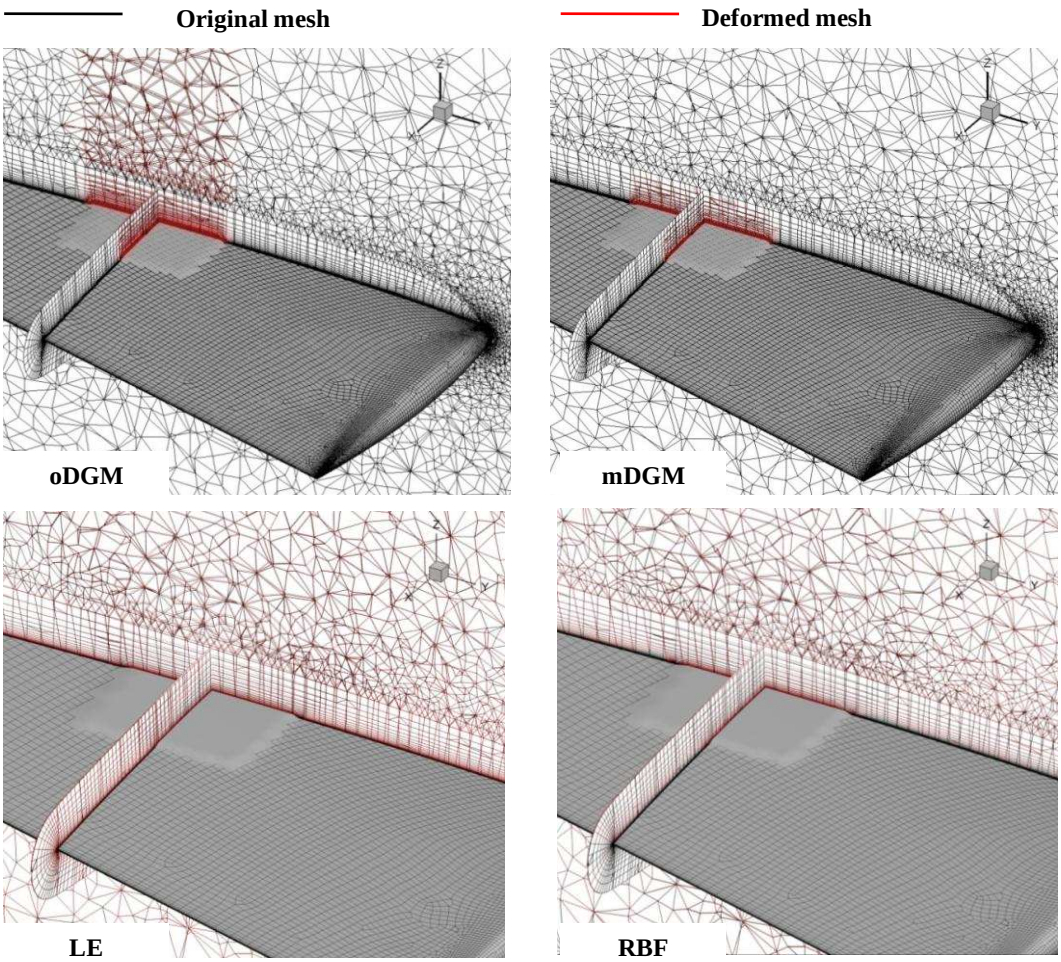

Figure 1 Comparison between different mesh movements. The wing is the ONERA M6.

$$\left[\frac{d\boldsymbol{T}}{d\boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{2.9}$$

Considering the residual associated with Eq. (2.1), i.e. $\{\boldsymbol{T}\} = \{\boldsymbol{X}\} - [\boldsymbol{E}]\{\boldsymbol{B}\}$, the following relations hold true:

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right]_{DGM} = [\boldsymbol{I}] \tag{2.10}$$

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right]_{DGM} = -\left[\boldsymbol{E}\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.11}$$

where $[\boldsymbol{I}]$ is the identity matrix. Hence, considering Eq. (1.3), it follows:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T [\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.12}$$

where $\{\boldsymbol{\Lambda}_{G,DGM}\}$ is the mesh-adjoint vector associated with DGM. Mura et al. [6] have proven that for explicit non-iterative mesh movements the following relation holds:

$$\left\{\frac{dI}{d\boldsymbol{X}}\right\} = \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T \left(= \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}\right) \tag{2.13}$$

The same derivation can also be followed for the residual associated with LE, i.e. $\{\boldsymbol{T}\} = [\boldsymbol{K}]\{\boldsymbol{X}\} - \{\boldsymbol{S}\}$:

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right]_{LE} = -[\boldsymbol{K}] \tag{2.14}$$

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right]_{LE} = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.15}$$

Therefore:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{LE} = -\left\{\boldsymbol{\Lambda}_{G,LE}\right\}^T \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.16}$$

where $\{\boldsymbol{\Lambda}_{G,LE}\}$ is the mesh-adjoint vector associated with LE and it is obtained by solving the following linear system of equations:

$$\left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_{G,LE}^T \boldsymbol{K}\right\} = \{\boldsymbol{0}\} \tag{2.17}$$

If each surface mesh point is used as a DV, i.e. $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$, Eqs. (2.12) and (2.16) reduce to respectively:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T [\boldsymbol{E}] \tag{2.18}$$

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{LE} = -\{\boldsymbol{\Lambda}_{G,LE}\}^T \tag{2.19}$$

The verification of the gradients expressed in Eqs. (2.12) and (2.16) can be found in Mura et al. [6] and Dwight and Brezillion [30] respectively.

III Analysis of the gradient calculation chain

There are two important quantities that need to be addressed. The first one is the gradient $\{dI/d\boldsymbol{X}\}$ and the second is the Jacobian $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$ which is available explicitly or implicitly if either DGM or LE is used respectively. The former is the one which describes the variation of the objective function w.r.t. a change in the volume mesh, whereas the latter describes the variation of volume mesh w.r.t. the surface mesh nodes. Since these two quantities are multiplied together, i.e. $\{dI/d\boldsymbol{X}\}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$, it is necessary to check that no cancelling effect is taking place. To do so, their distribution over the computation domain is further studied. To compare their distributions over the entire volume mesh, a line drawn perpendicularly from the wall to the far-field is selected and a series of point in its neighborhood are considered. These points are then used to evaluate the n.d.r. (normal decay rate). The ONERA M6 described in Section V is used to study the n.d.r. sampled values. Referring to Fig. 2 and its 3D representation on the lower left, a series of points along the normal to the upper surface were selected and the values interpolated. A semi-log on the x-axis (describing the distance from the wall) was used in order to highlight the rate of change of quantities that have their values distributed over different orders of magnitude. Fig. 2 clearly shows that, regardless of the different mesh movements used, all the Jacobians $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE,mDGM}$ start to visibly decay further from the wall, where $\{dI/d\boldsymbol{X}\}$ has almost decayed. In this scenario, no information about the objective function gradient is lost as $\{dI/d\boldsymbol{X}\}$ decays before any significant decay of $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$, for all mesh movements. It is the authors' opinion that this last scenario is always the case because it is preferable to distribute the deformations as evenly as possible over the entire mesh in order to reduce the risk of mesh element cross over. In fact, having a smaller decay rate of the Jacobian $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$ is an indication of the capability to uniformly spread the deformations away from the wall.

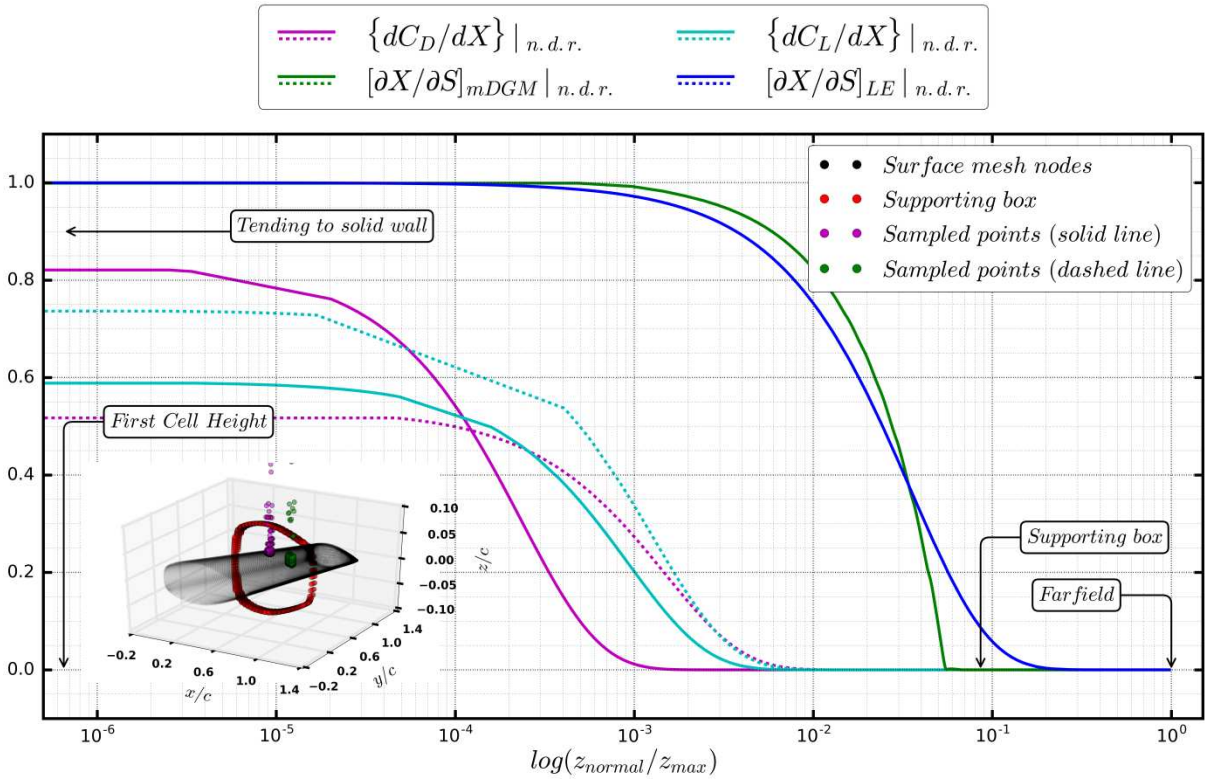

Figure 2 Comparison between different n.d.r.(s) over the computational domain.

IV Optimization Framework

TheONERA M6 wing at freestream Mach number of 0.833 and constant lift coefficient of 0.2733 [31] is considered. The Reynolds number is 11.72 Mil. based on the MAC (Mean Aerodynamic Chord). The hybrid unstructured quad-dominant mesh consists of 2.5 Mil. nodes and it was generated using the unstructured quad-dominant software SOLAR [32]. The flow solution is solved by the DLR TAU-Code [22] using a one-equation SA (Spalart-Allmaras) [23] turbulence model. The adjoint version of the solver and turbulence model is hand-differentiated and the frozen turbulent viscosity is not used.

The path toward the minimum, which is here used to investigate the effect of different non-consistent strategies, could be contaminated either by errors in the flow or flow-adjoint solution. On this matter, Nadarajah and Jameson [33] established that the accuracy of the gradients obtained solving the adjoint system depends on the flow solution level of convergence only. Furthermore, in general as long as the flow solution is converged to an acceptable level, the level of convergence of the flow-adjoint is generally consider to be satisfactory when stopped at one or two order of magnitude lower than the respective flow solution. By monitoring the

aerodynamic coefficients and the change in the derivatives, the following convergence values were considered satisfactory: $R_{flow} = 10^{-8}$ and $R_{adjoint} = 10^{-6}$.

A lift-constrained drag reduction optimization is considered by addressing the question of consistent versus non-consistent mesh movement and mesh sensitivity approaches. At each iteration the framework requires one flow solution, two flow-adjoint solutions (lift and drag), two mesh-adjoint solution (if used) and one search direction. These steps are repeated until convergence has been reached. The choice of the mesh movement and subsequently of its linearization which addresses the question of consistency is highlighted in Fig. 3.

The parameterization chosen for this study is the FFD (Free-Form Deformation) in its original formulation [34]. The FFD offers the advantage of not requiring an inverse geometric fitting on the original profile. FFD consists of creating a box-like lattice around the object which is then mapped using a linear combination of Bernstein Polynomial, $BP_{i,j,k}$, and the parametric control points vector, $\{\boldsymbol{D}\}$. Mathematically, this is expressed as a trivariate volume tensor product:

$$\{\boldsymbol{S}(u,v,w)\} = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{p} D_{i,j,k} \cdot BP_i(u) \cdot BP_j(v) \cdot BP_k(w) \tag{4.1}$$

where the physical coordinates (x, y, z) need to be mapped into local normalized coordinates $(u, v, w) \in [0,1] \times [0,1] \times [0,1]$ and n, m, p are the curve degree in each direction which also correspond to the number of control lattice subdivisions. The differentiation of Eq. (4.1) w.r.t. the control points, i.e. $\{\boldsymbol{D}\}$, is simply the product of the BPs along the x, y and z-axis:

$$\left[\frac{\partial \boldsymbol{S}(u,v,w)}{\partial \boldsymbol{D}}\right] = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{p} BP_i(u) \cdot BP_j(v) \cdot BP_k(w) \tag{4.2}$$

Where n, m, p representing the degree of the BPs which take respectively the value of 10, 6, 6. There are a total of 10 DVs which control only the upper surface and were chosen in order to maintain a fixed planform. The search direction updates are computed by the second order quasi-Newton L-BFGS (Low-memory Broyden-Fletcher-Goldfarb-Shanno) optimizer with line search [35].

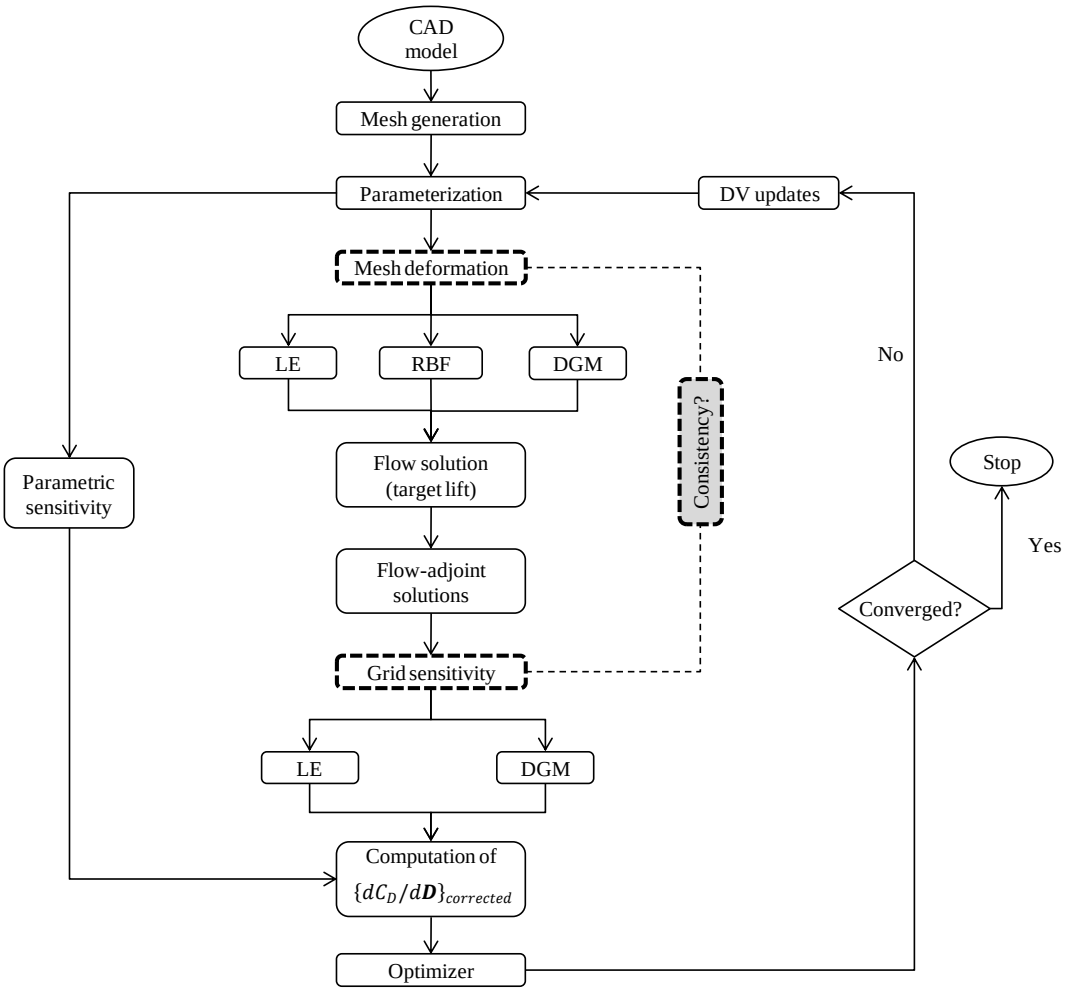

Figure 3 Workflow for the consistent vs. non-consistent framework.

The fixed-lift, i.e. $C_L = C_{L,target}$, is controlled by the flow solver. To be more specific the angle of attack is adjusted by the flow solver in order to respect the following inequality, i.e. $C_L(1-10^{-4}) < C_{L,target} < C_L(1+10^{-4})$. In this application, the fixed-lift is considered as an explicit constraint, this means that the gradient needs to be corrected in order to take into account the different angle-of-attack [36]:

$$I_{corrected} = C_D - \left(\frac{\frac{dC_D}{d\alpha}}{\frac{dC_L}{d\alpha}}\right)(C_L - C_{L,target}) \tag{4.3}$$

which means that the gradients needs to be corrected in order to take into account the different angle-of-attack [36]:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{corrected} = \left\{\frac{dC_D}{d\boldsymbol{D}}\right\} - \left(\frac{\frac{dC_D}{d\alpha}}{\frac{dC_L}{d\alpha}}\right)\left\{\frac{dC_L}{d\boldsymbol{D}}\right\} \tag{4.4}$$

where one wants to note that $\{dC_{L,target}/d\boldsymbol{D}\} = \{\boldsymbol{0}\}$ since the target lift does not depend on the DVs, in fact it is always adjusted solely by the flow solver and not by the optimizer. Furthermore, derivatives $(dC_{(D,L)}/d\alpha)$ are computed analytically using the flow-adjoint solution as follows:

$$\left(\frac{dC_{D,L}}{d\alpha}\right) = \left(\frac{\partial C_{D,L}}{\partial \alpha}\right) + \{\boldsymbol{\Lambda}_F\}^T \left\{\frac{\partial \boldsymbol{R}}{\partial \alpha}\right\} \tag{4.5}$$

This effectively requires two flow adjoint solutions: one for the drag and the other for the lift coefficient. The advantages is that the derivatives computed in Eq. (4.5) are accurate since they are based on the flow-adjoint solution. This correction prevents the optimizer to reduce the drag by simply reducing the lift.

V Test Cases

To investigate the effect of the extent of the grid differentiation (i.e. using the supporting box) and consistent vs. non-consistent approaches, a series of test cases have been devised which involve several combinations of oDGM, mDGM, RBF and LE for the mesh movement and grid sensitivity strategies in the optimization chain.

In order to simplify the nomenclature of the strategies being investigated, the following shorten version is proposed. Consider the $\Delta\boldsymbol{X}_{mDGM}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}$ case which describes a consistent approach where the mDGM is used both in the mesh movement, i.e. $\Delta\boldsymbol{X}_{strategy}$ and in the differentiated mesh movement step, i.e. $\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{strategy}$. On the other hand, for instance the $\Delta\boldsymbol{X}_{LE}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}$ case represents a non-consistent approach where the LE is used as a mesh movement in conjunction with the differentiated version of the mDGM.

V.A Metric Term Differentiation Extent

The goal of this section is to investigate the effect of supporting box distance from the wing surface. The supporting boxes are roughly 30, 50, 100 and 150 % of MAC. The fifth case does not use the supporting box which means the calculation of $\{dI/d\boldsymbol{X}\}$ is performed over the entire volume domain. As can be seen in Fig. 4, the introduction of the supporting box introduces a negligible difference in the final value of the optima for the

cases with supporting box of 50% MAC and above. However, it starts to deviate significantly for the case where the supporting box was placed at 30% MAC. This is because, such a low supporting box height is cutting the information from the gradient $\{dI/d\boldsymbol{X}\}$. This shows that the gradient $\{dI/d\boldsymbol{X}\}$ needs to be captured accurately and care needs to be taken when using the supporting box approach.

V.B Consistent Approaches

This section tries to establish two reference cases, using consist approaches which will then be compared against the non-consistent approaches. The optimizations being performed will use mDGM or LE to perform an optimization using a consistent approach. The red and blue lines in Fig. 5 represents the optimizations for mDGM and LE respectively. All optimizations will be started from the same initial design to guarantee a fair comparison between each methodology. The optimization history shows a very similar progression for both consistent methods and give a similar drag reductions at the final optimization iteration.

Interestingly, when the DV gradients (see Fig. 6) are analyzed, they do not have the same starting values and follow different search. This is not reflected in deviations in the initial DV values depicted in Fig. 7, although there is a deviation in the search paths. Towards the end of the optimization the difference of this value reduces. In this case, the differences are solely due to the different mathematical formulations of both mesh updates and mesh sensitivities.

This study has compared two consistent approaches and the following studies aim to establish the relative effect of the different mesh movement and grid sensitivity methods by analysis of a series of different non-consistent approaches.

V.C The Effect of Non-consistent Mesh Sensitivities

The goal of this section is to numerically investigate what happens when a particular mesh movement is used in conjunction with a differentiated mesh movement based on a different mesh update strategy. This is representative of situations where, for instance, the grid sensitivity routine update (based on the chosen mesh movement) is either too memory or computationally expensive.

In order to investigate this case, a specific mesh movement is chosen and used in each optimization study first with its differentiated version and then with different mesh sensitivity strategies. The cases compared in this section are: $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ and $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ for mDGM mesh movement, $\Delta\boldsymbol{X}_{RBF}\,[\partial\boldsymbol{X}/$

$\partial \boldsymbol{S}]_{mDGM}$ and $\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ for RBF mesh movement and $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ and $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ for LE mesh movement. Symbols $\Delta \boldsymbol{X}$ and $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ designate the mesh movement and grid sensitivity, respectively. Referring to Fig. 5, the optimizations are started from the same initial non-optimized geometry and at iteration No. 2, a slight difference is registered in terms of computed drag coefficient which is maintained up to the end of the optimization.

Fig. 7 shows the DV updates and as can be seen all the cases have the same DV starting values. Furthermore, a close look of the DV paths reveals that only for some DVs there is a noticeable deviation. The deviations introduced are initiated by the grid sensitivity strategies employed. As per the case presented in Section V.A, the deviations at iteration No, 1 do not lead to large differences in the optima because are not big enough for the optimiser to predict a different DV updates.

These small differences in the gradients (see Fig. 6) computed at iteration No. 1 can be visualized by using gradient $\{dI/d\boldsymbol{S}\}$ and not $\{dI/d\boldsymbol{D}\}$ because they differ by a Jacobian factor, i.e. $[\partial \boldsymbol{S}/\partial \boldsymbol{D}]$ which is the same for both methods. The gradients are sampled along a spanwise cut depicted in Fig. 8 (note that the scale on the z-axis (right) is true to the profile coordinates only). The resulting 2D vectors are shown in the same figure. The trend is equally described by the two methods, but large deviations are registered at the trailing edge and in the maximum values of some points especially at the shock area.

In Section III, it was established that both differentiated mesh movements, i.e. mDGM and LE, are not cutting the physical quantity $\{dI/d\boldsymbol{X}\}$. This last quantity is the only part of the chain that carries physical information and is the same regardless of the mesh movement chosen as clearly shown in Eq. (1.4). Furthermore, $\{dI/d\boldsymbol{X}\}$ is based on the flow-adjoint solution and its accuracy depends on the flow solution as Nadarajah and Jameson [33] have proven. Of course, the flow solution is obviously influenced by the mesh density and quality, but these factors cannot justify the small differences in the gradients as highlighted in Fig. 8 because they are used by both the mDGM and LE-based chain with no changes. Now that vector $\{dI/d\boldsymbol{X}\}$ has been excluded as a cause of the small differences, the attention is moved to the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$.

One of the main difference between LE and mDGM is that the LE elastic matrix $[\boldsymbol{K}]$ needs to be inverted iteratively (i.e. $[\boldsymbol{K}]^{-1} = [\partial \boldsymbol{X}/\partial \boldsymbol{S}]$), whereas the mDGM provides the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}](= [\boldsymbol{E}])$ analytically, thus without the need to use an iterative method. Therefore, the values stored in the DGM-based Jacobian could be taken as an example to match, but this is conceptually wrong because the two methods are mathematically

different and therefore, in theory, are not expected to yield the same results. However, it is anticipated that the differences between the two methods are not large.

The second step is to actually study the magnitude of the difference between $[\boldsymbol{K}]^{-1}$ and $[\boldsymbol{E}]$ matrices over the computational domain. As can be seen from Fig. 2, the n.d.r. computed over the neighboring points (introduced in Section III) of these two Jacobians show the same trend close to the wall where the values of gradient $\{dI/d\boldsymbol{X}\}$ are not zero. However, just before vector $\{dI/d\boldsymbol{X}\}$ reaches zero, the two Jacobian distributions start to show a different trend. Since the region where the two Jacobians are equal covers most of region (near the wall) where the values of gradient $\{dI/d\boldsymbol{X}\}$ are not zero, but differs in a small region (away from the wall) where the values of gradient $\{dI/d\boldsymbol{X}\}$ tend to zero, it is possible to explain why they are largely equal, but still present minor differences.

V.D The Effect of Non-consistent Mesh Movement

The goal of this section is to investigate numerically what happens when a particular grid sensitivity is used in conjunction with different mesh movements. This is representative of situations where another mesh movement, different from the one used in the mesh adjoint is deemed either more robust or less computationally intensive. In order to investigate this case, a specific mesh sensitivity strategy is chosen and used in each optimization study first with its mesh movement then with different mesh update strategies. The cases compared in this section are: $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$, $\Delta\boldsymbol{X}_{LE}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ and $\Delta\boldsymbol{X}_{RBF}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ for the grid sensitivity based on the mDGM and $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$, $\Delta\boldsymbol{X}_{LE}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ and $\Delta\boldsymbol{X}_{RBF}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ for the grid sensitivity based on the LE.

In Fig. 5 the test cases start from the same initial non-optimized geometry, but at iteration No. 2, there is a very small difference in terms of computed drag coefficient. This is maintained up to the end of the simulation. In Fig. 7, all the DVs follow a similar path with only minor deviations.

The small differences in drag at iteration No. 2, are due to the different mesh movements used which introduces deviations in the near-wall mesh orthogonality even if the first surface mesh update is the same.

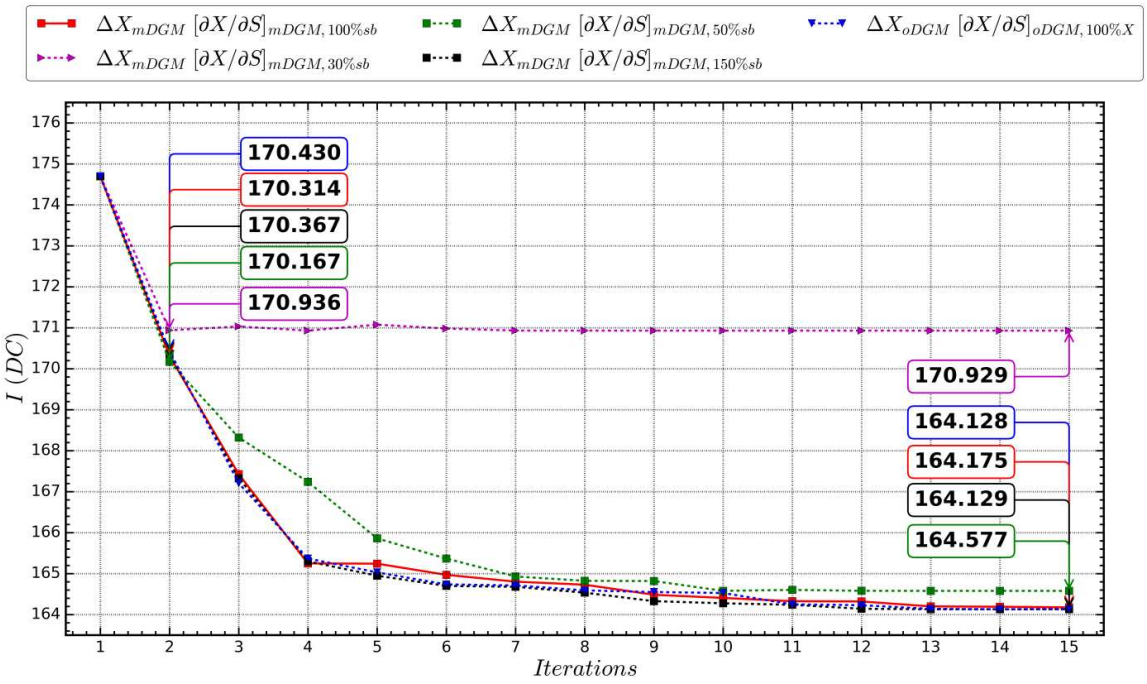

Figure 4 Convergence history comparison for different mesh size differentiations.

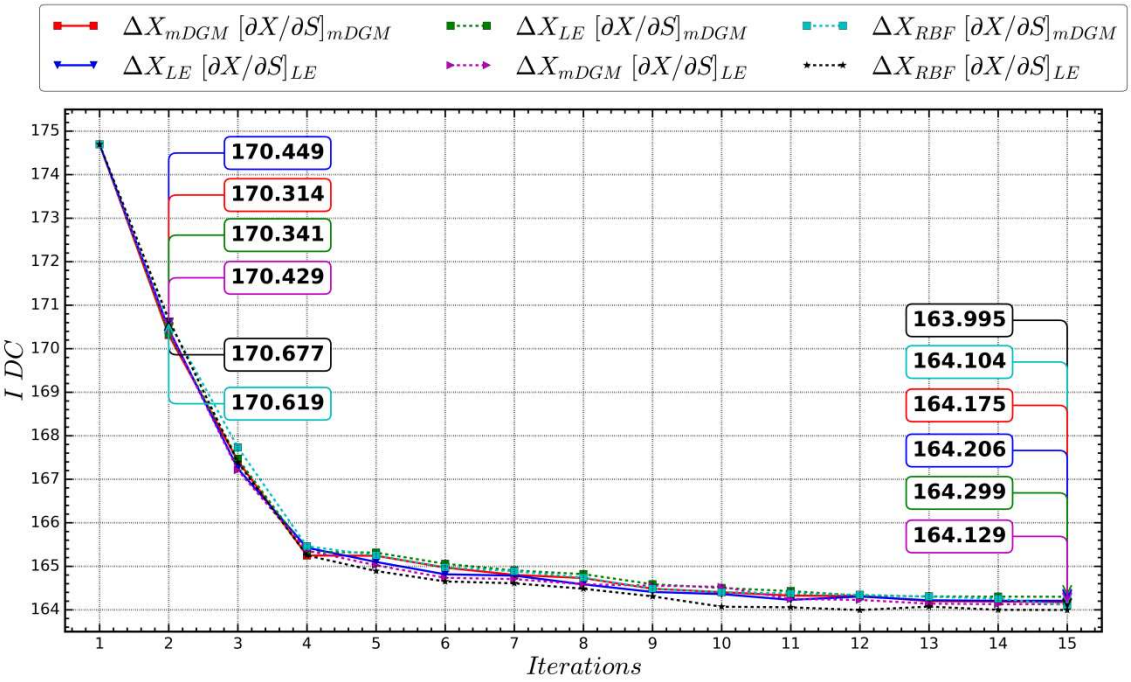

Figure 5 Convergence history comparison for the consistent and non-consistent cases.

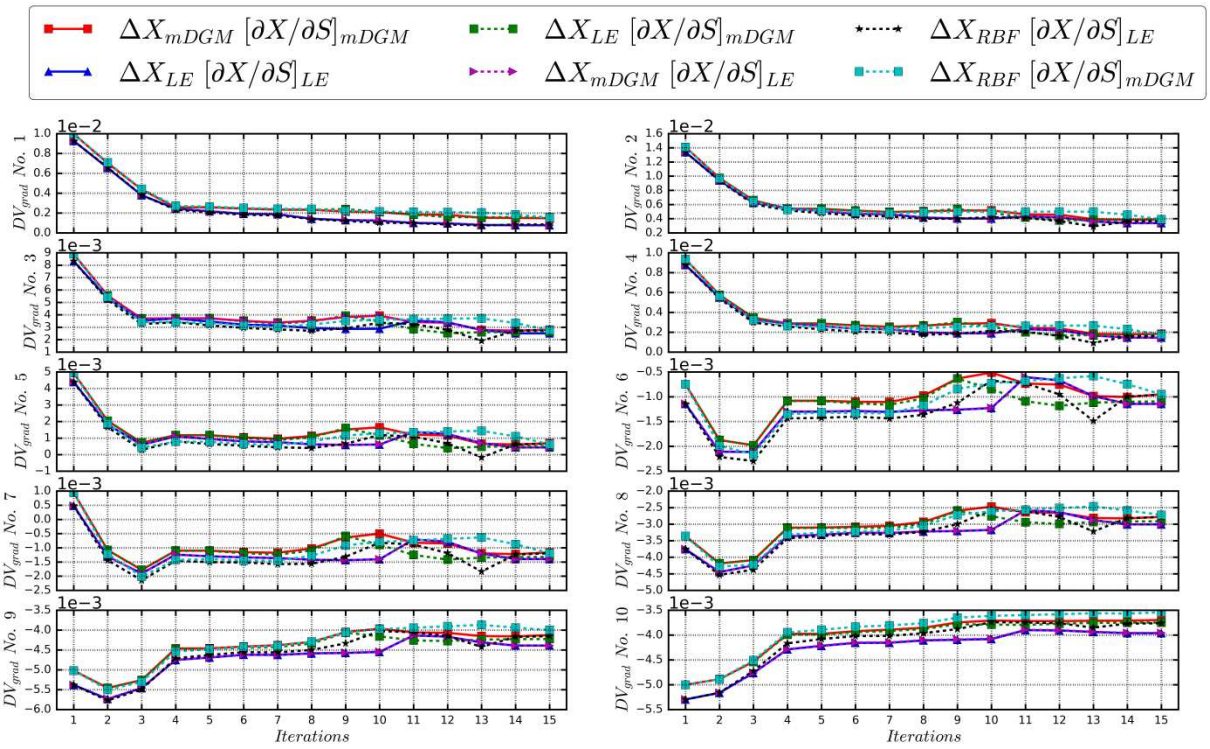

Figure 6 DV's gradients (corrected for the change in the AoA) history comparison for the consistent and non-consistent cases.

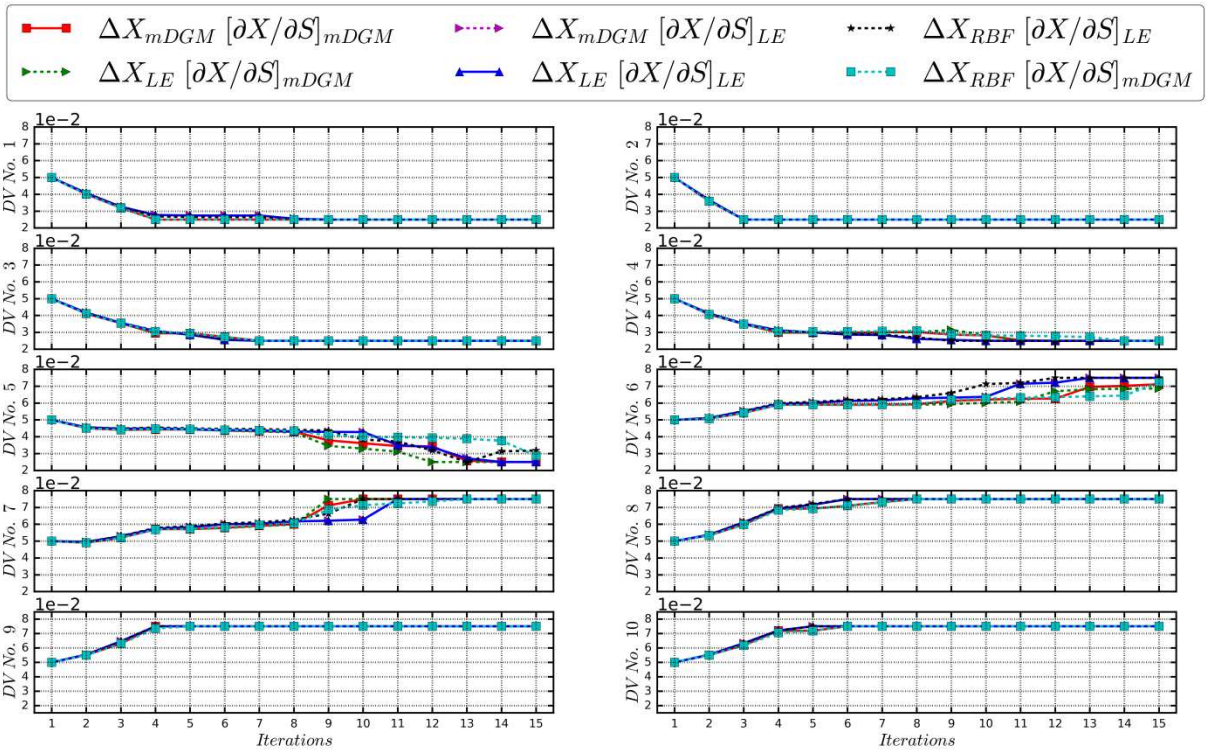

Figure 7 DV's updates history comparison for the consistent and non-consistent cases.

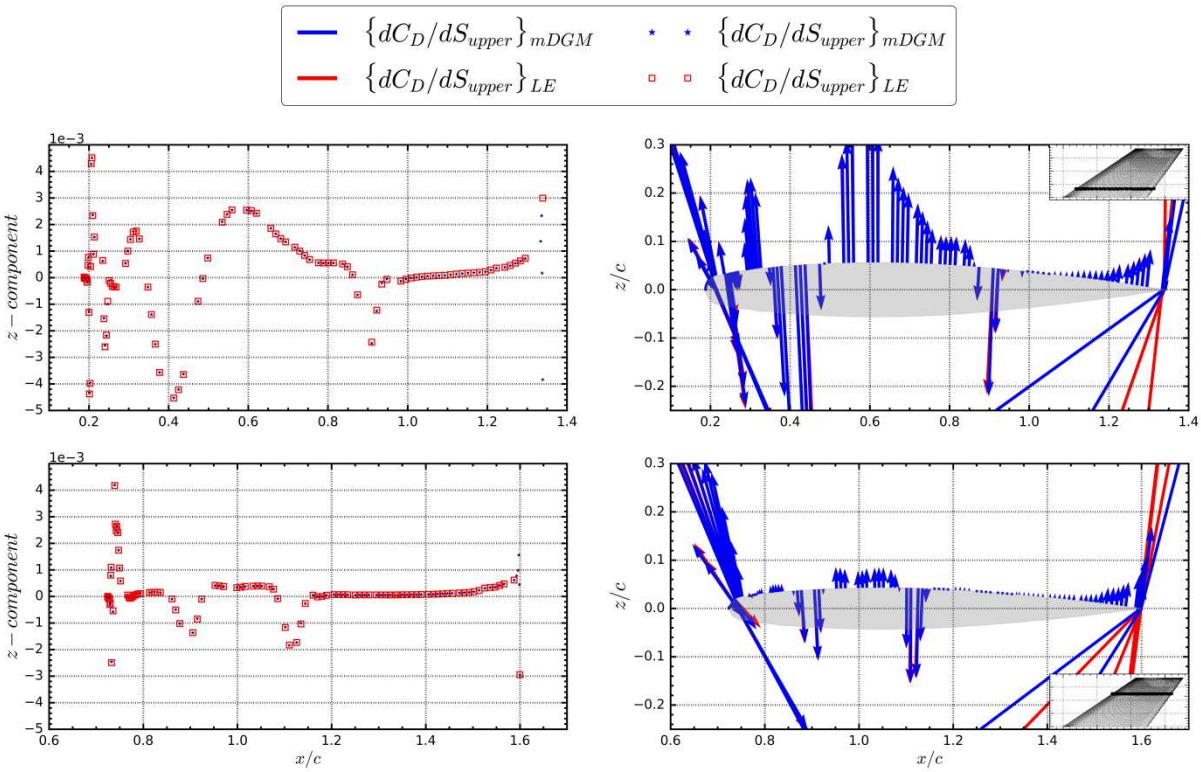

Figure 8 Selected point-to-point gradients comparison computed based on the flow solution at iteration No. 1.

VI Memory and CPU time Comparison

Mesh deformation and mesh sensitivity are compared in terms of memory and CPU time requirements assessed on a single Intel Xeon 2.67GHz processor with 70.8Gb of memory. The cost of the first iteration is analyzed separately from the others because this is where the mesh pre-processing is performed. This refers to the computation of matrices $[\boldsymbol{E}]$, and $[\boldsymbol{K}]$ which is performed only once and not repeated afterwards because the mesh connectivity is not changed.

Referring to Tab. 2 the following conclusions can be drawn. It is evident that whenever either the $\Delta\boldsymbol{X}_{RBF}$ or $\Delta\boldsymbol{X}_{LE}$ approach is used there is an increase in both CPU time and memory over the consistent $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach. However, these advantages are slightly offset by the lower pre-processing CPU time offered by both LE and RBF. This makes the $\Delta\boldsymbol{X}_{RBF}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach, at iteration No. 1, 19.88% more CPU time intensive than the $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach because of the two pre-processing steps and the additional $\Delta\boldsymbol{X}_{RBF}$ CPU time.

In terms of memory, the cost is comparable only for the $\Delta \boldsymbol{X}_{mDGM}$ and the $\Delta \boldsymbol{X}_{RBF}$ approach, whereas the $\Delta \boldsymbol{X}_{LE}$ approach requires memory and CPU time which are two orders of magnitude bigger. The same trend, although less severe, is also registered while computing the gradient. The computation of gradient $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ requires only 1.2Gb of memory and 358s of CPU time compared to the 15Gb and 1,726s required by the $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$. Note that, although Eqs. (2.4) and (2.17) share the same Jacobian $[\boldsymbol{K}]$, the RHS and LHS are essentially different vectors. In fact, the memory needed to compute $\{dI/d\boldsymbol{X}\}$ is greater than the one needed to compute just vector $\{\boldsymbol{X}\}$ in Eq. (2.4). This explains the different memory values reported in Tab 2.

What is interesting is that neither the $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ nor $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ require different memory as one varies the objective function, i.e. $C_{D,L}$. On the other hand, the $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ term shows a CPU time which gives an uneven split of time between each objective function. The $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ shows a total CPU time of 358s which is equally distributed (i.e. 179s each) to compute the gradient in Eq. (2.18) for both the lift and the drag coefficient. On the other hand the $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ approach requires a total CPU time of 1,726s which is split between C_D, which took 685s, and C_L, which took 1,041s. One explanation for this is that the memory is proportional to the size of the system, i.e. $N_{VM} \times N_{SM}$, whereas the CPU time is proportional to how long it takes to solve the linear system which cannot be considered constant as one changes the objective function.

After the first iteration, the advantage of the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ strategies over the other approaches becomes evident as the data in Tab. 3 clearly shows. In contrast, for each iteration after the first, there is an increase in CPU time for the $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ approach of almost 8 times compared to the consistent $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ approach. There is also a large computational cost for the non-consistent case where the $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ is used. This gives a CPU time equal to 4.3 times greater than the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$. If the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ is used the CPU time registered is 3.35 times greater than the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ approach.

It is possible to estimate the linear cumulative computational time due to both the differences in mesh movement and sensitivity. In order to do so, the CPU time to build matrices $[\boldsymbol{E}]$, and $[\boldsymbol{K}]$ is not considered after iteration No. 1 and the remaining CPU time is projected linearly (using the data reported in Tab. 3) because it does not vary much between the iterations. This is shown in Fig. 9.

Table 2 CPU time and memory requirements comparison at iteration No. 1.

Strategy	Memory (Gb)		CPU time (s)			
	Mesh movement	Gradient computation	Pre-processing	Mesh movement	Gradient computation	Total
$\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	0.232_{mDGM}	1.2_{mDGM}	598_{mDGM}	50_{mDGM}	358_{mDGM}	1,006
$\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		15_{LE}	$598_{mDGM} + 121_{LE}$		$1{,}726_{LE}$	2,495
$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$	14_{LE}	15_{LE}	121_{LE}	$1{,}807_{LE}$	$1{,}726_{LE}$	3,654
$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$		1.2_{mDGM}	$598_{mDGM} + 121_{LE}$		358_{mDGM}	2,884
$\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	0.321_{RBF}	1.2_{mDGM}	$55_{RBF} + 598_{mDGM}$	195_{RBF}	358_{mDGM}	1,206
$\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		15_{LE}	$55_{RBF} + 121_{LE}$		$1{,}726_{LE}$	2,097

For a simulation of 15 iterations the final CPU time is almost 13hrs less expensive than the $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ approach. It is interesting to see how the inconsistent $\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ approach does not lag behind as much as the others, in fact, after the first iteration, the increase at each iteration is of about 35.53% over $\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$. This is estimated over a period of 15 iteration to give a CPU time of roughly only 0.62hr greater than the $\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ consistent approach.

Table 3 CPU time and memory requirements comparison per iteration(after iteration No. 1).

Strategy	Memory (Gb)		CPU time (s)		
	Mesh movement	Gradient computation	Mesh movement	Gradient computation	Total
$\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	0.232_{mDGM}	1.2_{mDGM}	50_{mDGM}	358_{mDGM}	408
$\Delta \boldsymbol{X}_{mDGM}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		15_{LE}		$1{,}726_{LE}$	1,776
$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$	14_{LE}	15_{LE}	$1{,}807_{LE}$	$1{,}726_{LE}$	3,533
$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$		1.2_{mDGM}		358_{mDGM}	2,165
$\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	0.321_{RBF}	1.2_{mDGM}	195_{RBF}	358_{mDGM}	553
$\Delta \boldsymbol{X}_{RBF}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		15_{LE}		$1{,}726_{LE}$	1,921

This analysis confirms that in the case, where a non-consistent approach is considered more robust either in its mesh movement or differentiated mesh movement update, it can still be used without impacting the final result. For example, for a large mesh movement (e.g. wing bending/twisting) the RBF method may be chosen for its robustness, however the Delaunay method will still provide a faster gradient computation.

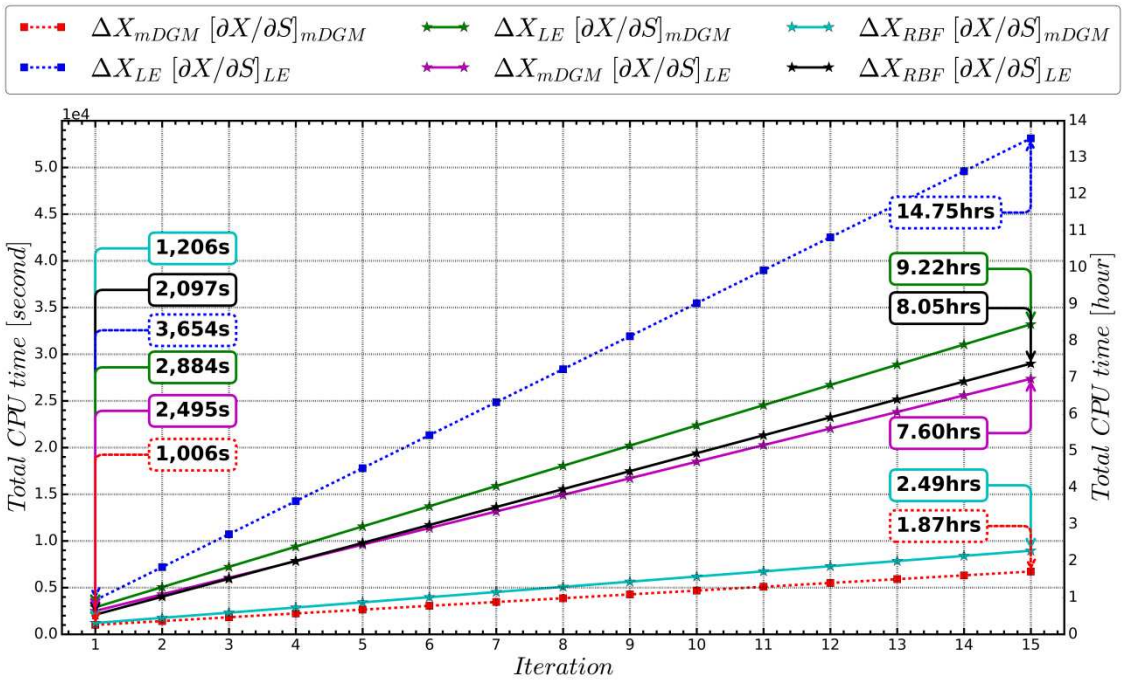

Figure 9 Cumulative CPU times comparison for different approaches.

In order to further understand the impact of the saving offered by the mDGM method, the cost of each step is compared against the overall CPU time of a full (single) optimization iteration. To simplify the analysis, only the second iteration (representative of the others iterations except the first) is considered (this choice is justified by the fact that the first one does not require to move the baseline mesh). Since the comparison needs to consider the relative time between the steps, the total cost (computed as the sum of one flow solution, two flow-adjoint solutions, one mesh movement and two grid sensitivities) is considered as a reference value.

For the case under analysis, a single flow-adjoint takes roughly 80% of the flow solution, however this value is likely to be comparable and in some cases higher than the flow solution. Both LE-based mesh movement and sensitivity steps take separately roughly 30% of the flow solution but the cost w.r.t. the overall cumulative time (as depicted in Fig. 10) is as low as 9%. These values are in line with the data reported by Nielsen and Park [5].

Thus, both LE-based mesh movement and grid sensitivity steps represent together roughly 27% of the total CPU time. Fig. 11 shows that the saving offered by the mDGM-based fully consistent chain is significant both in the mesh movement and sensitivity steps. The 18% of the total LE-based CPU time spent in computing the two grid sensitivities is greatly reduced down to just 1%, a saving of 17% w.r.t. the overall cost. On the other hand, the CPU time used for the mesh movement is reduced to an insignificant percentage, i.e. 0.5%, of the total CPU time.

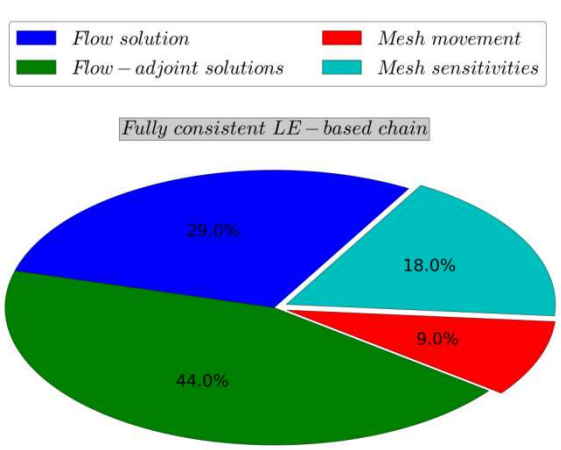

Figure 10 - Cumulative CPU time breakdown for the fully consistent LE-based chain.

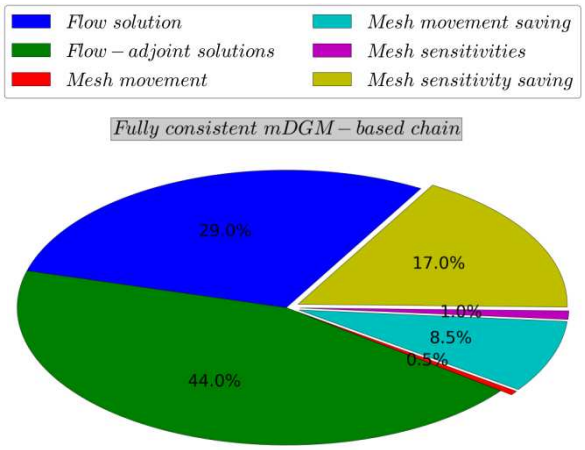

Figure 11 - Cumulative CPU time breakdown for the fully consistent DGM-based chain.

VIII Concluding Remarks

The paper addresses the question as to what happens when a non-consistent grid approach, either in the mesh movement or in the differentiated mesh movement, is used in the discrete adjoint optimization framework. The different optimization results found in this paper cannot be distinguished within the current RANS accuracy level and can be considered essentially representative of the same minimum.

When calculating the gradients using the adjoint chain it is important that the grid sensitivity (i.e. $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$) does not significantly decay before $\{dI/d\boldsymbol{X}\}$ has decayed. This was demonstrated by using an artificial cut-off by placing the supporting box at different distances from the aerodynamic surface. It was found that a supporting box of 30% MAC did not perform well during optimization due to the cut-off of $\{dI/d\boldsymbol{X}\}$ values. This is essential for the accuracy of the objective function gradients within the optimization process, especially when non-consistent approaches are used.

In particular, small changes in the gradients (caused by non-consistent grid sensitivities) were not enough to trigger different DV updates (computed by the optimiser). Also small changes in the first (deformed) grid updates (caused by non-consistent grid movements) were not enough to trigger a difference in the objective functions (computed by the flow solver).

Based on the framework and numerical results presented in this paper, non-consistent approaches can be used whenever deemed necessary without incurring in a large deviation in the minimum found. This allows non-consistent approaches to be used in mesh movement and mesh sensitivity calculations in the adjoint based aerodynamic shape optimization.

Acknowledgements

The authors acknowledge the financial support by the University of Sheffield, the UK Engineering and Physics Research Council (EPSRC) and Airbus for carrying out the research. They also like to thank Caslav Ilic and Andrei Merle at the German Aerospace Center (DLR), Braunschweig, for the help received regarding the use of the TAU code and the implementation of the proposed method in the code.

References

[1] Jameson, A., Pierce, N. A., and Martinelli, L., “Optimum Aerodynamic Design Using the Navier–Stokes Equations,” 35th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, AIAA 97-0101, January 1997.
doi: 10.2514/6.1997-101

[2] Le Moigne, A., and Qin, N., "Variable-Fidelity Aerodynamic Optimization for Turbulent Flows Using a Discrete Adjoint Formulation," AIAA Journal, Vol. 42, No. 7, 2004, pp. 1281-1292.
doi: 10.2514/1.2109

[3] Morris, A. M., Allen, C. B., and Rendall, T. C. S., “Domain-Element Method for Aerodynamic Shape Optimization Applied to a Modern Transport Wing,” AIAA Journal, Vol. 47, No. 7, 2009, pp. 1647– 1659.
doi: 10.2514/1.39382

[4] Hicken, J. E., and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," AIAA Journal, Vol. 48, No. 2, 2010, pp. 400-413.
doi: 10.2514/1.44033

[5] Nielsen, E. J., and Park, M. A., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," AIAA Journal, Vol. 44, No. 5, 2006, pp. 948-953.

doi: 10.2514/1.16052

[6] Mura, G. L., Hinchliffe, B. L., Qin, N., and Brezillon, J., "Efficient Method to Eliminate Mesh Sensitivity in Adjoint-Based Optimization", AIAA Journal, Vol. 55, No. 4, 2017, pp. 1140-1151.

https://doi.org/10.2514/1.J055212

[7] Liu, X., Qin, N., and Xia, H., "Fast Dynamic Grid Deformation Based on Delaunay Graph Mapping," Journal of Computational Physics, Vol. 211, No. 2, 2006, pp. 405–423.

doi: 10.1016/j.jcp.2005.05.025

[8] Baker, T. J., and Cavallo, P. A., "Dynamic Adaption for Deforming Tetrahedral Meshes," 37th AIAA Aerospace Sciences Meeting and Exhibit, Norfolk, USA, AIAA 1999-3253, January 1999.

doi: 10.2514/6.1999-3253

[9] de Boer, A., van der Schoot, M. S., and Bijl, H., "Mesh Deformation Based on Radial Basis Function Interpolation," Computers and Structures," Vol. 85, No. 11-14, 2007, pp. 784–795.

doi.org/10.1016/j.compstruc.2007.01.013

[10] Batina, J. T., "Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-aircraft Aerodynamic Analysis", AIAA Journal, Vol. 29, No. 3, 1991, pp. 327-333.

doi.org/10.2514/3.10583

[11] Nemec, M., and Zingg, D. W., "Newton–Krylov Algorithm for Aerodynamic Design Using the Navier–Stokes Equations," AIAA Journal, Vol. 40, No. 6, 2002, pp. 1146–1154.

doi: 10.2514/2.1764

[12] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," Optimization and Engineering, Vol. 6, No. 1, 2005, pp. 33–62.

doi: 10.1023/B:OPTE.0000048536.47956.62

[13] Zhu, F., and Qin, N., "Using Mesh Adjoint for Shock Bump Deployment and Optimisation on Transonic Wings," 53rd AIAA Aerospace Sciences Meeting and Exhibit, Kissimmee, USA, AIAA 2015-1488, January 2015.

doi: 10.2514/6.2015-1488

[14] Hinchliffe, B. L., and Qin, N., “Using Surface Sensitivity from Mesh Adjoint for Transonic Wing Drag Reduction,” AIAA Journal, 2017, Vol.55: pp. 818-831.

doi: 10.2514/1.J055319

[15] Bobrowski, K., Ferrer, E., Valero, E., and Holger, B., “CAD-based Aerodynamic Shape Optimization Using Geometry Surrogate Model and Adjoint Methods,” 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Washington, USA, AIAA 2016-3831, June 2016.

doi: 10.2514/6.2016-3831

[16] Jakobsson, S., and Amoignon, O., “Mesh Deformation Using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization,” Computers and Fluids, Vol. 36, No. 6, 2007, pp. 1119–1136.

doi: 10.1016/j.compfluid.2006.11.002

[17] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh Movement for a Discrete-adjoint Newton-Krylov Algorithm for Aerodynamic Optimization," AIAA Journal, Vol. 46, No. 7, 2008, pp. 1695-1704.

doi: 10.2514/1.33836

[18] Nambu, T., Mavriplis, D. J., and Mani, K., "Adjoint-based Shape Optimization of High-lift Airfoils Using the NSU2D Unstructured Mesh Solver," 52nd Aerospace Sciences Meeting, National Harbor, USA, AIAA 2014-0554, January, 2014.

doi: 10.2514/6.2014-0554

[19] Mavriplis, D. J., “Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes,” AIAA Journal, Vol. 44, No. 1, 2006, pp. 42-50.

doi: 10.2514/1.15696

[20] Maute, K., Nikbay, M., and Farhat, C., “Sensitivity Analysis and Design Optimization of Three-Dimensional Non-Linear Aeroelastic Systems by the Adjoint Method,” International Journal for Numerical Methods in Engineering, Vol. 56, No. 6, 2003, pp. 911–933.

doi: 10.1002/nme.599

[21] Burgreen, G. W., and Baysal, O., “Three-Dimensional Aerodynamic Shape Optimization Using Discrete Sensitivity Analysis,” AIAA Journal, Vol. 34, No. 9, 1996, pp. 1761–1770.

doi: 10.2514/3.13305

[22] Gerhold, T., Galle, M., Friedrichs, O., and Evans, J., “Calculation of Complex Three-Dimensional Configurations Employing the DLR TAU-Code," 35th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, AIAA 1997-0167, January 1997.

doi.org/10.2514/6.1997-167

[23] Spalart, P. R., and Allmaras, S. R., “A One-Equation Turbulence Model for Aerodynamic Flows,” 30th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, AIAA 1992-439, January 1992.

doi: 10.2514/6.1992-439

[24] Jacques, P., Nguyen-Dinh, M., and Trontin, P., "Goal Oriented Mesh Adaptation Using Total Derivative of Aerodynamic Functions with Respect to Mesh Coordinates with Applications to Euler Flows," Computers & Fluids, Vol. 66, 2012, pp. 194-214.

https://doi.org/10.2514/6.2012-158

[25] Mura, G. L., Hinchliffe, B. L., and Qin, N., “Robust Delaunay Graph Based Mesh Movement for Adjoint Mesh Sensitivity,” Proceedings of the RAeS Applied Aerodynamics Conference, Bristol, UK, July 2016.

[26] Nielsen, E. J., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations," AIAA Journal, Vol. 37, No. 11, 1999, pp. 1411-1419.

doi: 10.2514/2.640

[27] Dwight, R., “Robust Mesh Deformation Using the Linear Elasticity Equations,” in Computational Fluid Dynamics 2006 (H. Deconinck and E. Dick, eds.), pp. 401–406, Springer, 2006.

DOI: 10.1007/978-3-540-92779-2_62

[28] Rendall, T. C. S., and Allen, C. B. "Efficient Mesh Motion Using Radial Basis Functions with Data Reduction Algorithms," Journal of Computational Physics, Vol. 228, No. 17, 2009, pp. 6231-6249.

doi: 10.1016/j.jcp.2009.05.013

[29] Jakobsson, S., and Amoignon, O., “Mesh Deformation Using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization,” Computers and Fluids, Vol. 36, No. 6, 2007, pp. 1119–1136.

doi: 10.1016/j.compfluid.2006.11.002

[30] Dwight, R. P., and Brezillon, J., "Effect of Approximations of the Discrete Adjoint on Gradient-based Optimization," AIAA Journal, Vol. 44, No. 12, 2006, pp. 3022-3031.

doi.org/10.2514/1.21744

[31] Schmitt, V., and Charpin, F., “Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," AGARD AR-138, May 1979.

[32] Leatham, M., Stokes, S., Shaw, J. A., Cooper, J., Appa, J., and Blaylock, T. A., "Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations," Fluids 2000 Conference and Exhibit, Denver, USA, AIAA 2000-2247, June 2000.

doi: 10.2514/6.2000-2247

[33] Nadarajah, S., and Jameson A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, AIAA 2000-0667, January 2000.

doi.org/10.2514/6.2000-667

[34] Sederberg, T. W., and Parry, S. R., “Free-form Deformation of Solid Geometric Models,” ACM SIGGRAPH Computer Graphics, Vol. 20, No. 4, 1986, pp. 151-160.

doi: 10.1145/15922.15903

[35] Nocedal, J., “Updating Quasi-Newton Matrices with Limited Storage,” Mathematics of Computation, Vol. 35, 1980, pp. 773-782.

doi:10.1090/S0025-5718-1980-0572855-7

[36] Reuther, J. J., Alonso, J., Rimlinger, M. J., and Saunders, D., “Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers,” Journal of Aircraft, Vol. 36, No. 2, 1999, pp. 61-74.

doi: 10.2514/2.2414