Combining an OpenFOAM®-Based Adjoint Solver with RBF Morphing for Shape Optimization Problems on the RBF4AERO Platform

E. M. Papoutsis-Kiachagias, K. C. Giannakoglou, S. Porziani, C. Groth, M. E. Biancolini, E. Costa and M. Andrejašič

Abstract This chapter presents a combination of an OpenFOAM®-based continuous adjoint solver and a Radial Basis Function (RBF)-based morpher forming a software suite able to tackle shape optimization problems. The adjoint method provides a fast and accurate way for computing the sensitivity derivatives of the objective functions (here, drag and lift forces) with respect to the design variables. The latter control a group of RBF control points used to deform both the surface and volume mesh of the CFD domain. The use of the RBF-based morpher provides a fast and robust way of handling mesh and geometry deformations with the same tool. The coupling of the above-mentioned tools is used to tackle shape optimization problems in automotive and aerospace engineering. This work was funded by the RBF4AERO "*Innovative benchmark technology for aircraft engineering design and efficient design phase optimisation*" project funded by the EU 7th Framework Programme (FP7-AAT, 2007-2013) under Grant Agreement No. 605396 and the presented methods are available for use through the RBF4AERO platform (www.rbf4aero.eu).

E. M. Papoutsis-Kiachagias (✉) · K. C. Giannakoglou
Parallel CFD & Optimization Unit,
National Technical University of Athens (NTUA), Athens, Greece
e-mail: vaggelisp@gmail.com

K. C. Giannakoglou
e-mail: kgianna@central.ntua.gr

S. Porziani · E. Costa
D'Appolonia S.p.A., Rome, Italy
e-mail: stefano.porziani@dappolonia.it

E. Costa
e-mail: emiliano.costa@dappolonia.it

C. Groth · M. E. Biancolini
University of Rome Tor Vergata (UTV), Rome, Italy
e-mail: corrado.groth@uniroma2.it

M. E. Biancolini
e-mail: biancolini@ing.uniroma2.it

M. Andrejašič
PIPISTREL d.o.o. Ajdovščina, R&D, Department of Aerodynamics, Ajdovščina, Slovenia
e-mail: matej.andrejasic@pipistrel.si

J. M. Nóbrega and H. Jasak (eds.), *OpenFOAM®*,
https://doi.org/10.1007/978-3-319-60846-4_5

Nomenclature

RBF	Radial basis functions
SD	Sensitivity derivatives
FD	Finite differences
PDE	Partial differential equation
SI	Surface integrals
FI	Field integrals
E-SI	Enhanced surface integrals

1 Introduction

During recent years, CFD-based aerodynamic shape optimization has been attracting the interest of both academia and industry. The constituents needed for executing an automated shape optimization loop include the flow solver, the geometry parameterization (the parameters of which act as the design variables), an optimization method capable of computing the optimal values of the design variables and a way to adapt (or regenerate) the computational mesh to each candidate solution.

In the studies presented herein, the steady-state flow solver of the open-source CFD toolbox OpenFOAM® is used to numerically solve the Navier-Stokes equations for incompressible, turbulent flows.

Shape parameterization techniques can be divided into two categories, i.e., those parameterizing only the surface to be optimized and those that simultaneously also deform the surrounding nodes of the interior mesh. The great advantage of the latter is that the interior of the mesh is also deformed, thus avoiding costly re-meshing and allowing for the initialization of the flow field from the solution of the previous optimization cycle, since the mesh topology is preserved. Here, a number of parameters controlling the positions of groups of RBF control points are used as the design variables, using technology and methods of the RBF Morph software [2].

Gradient-based optimization methods require a high degree of effort to develop but can have a cost per optimization cycle that does not scale with the number of design variables, if the adjoint method is used to compute the sensitivity derivatives (SD). In this work, a continuous adjoint method that takes into consideration the differentiation of the turbulence model PDE is used to compute the SD of the force objective function w.r.t. the design variables [7]. The adjoint solver has been implemented in-house, based on the OpenFOAM®software.

The above-mentioned tools are combined in order to form an automated optimization loop. Two applications are presented, namely the drag minimization of the DrivAer car model [3], and the lift-to-drag ratio maximization of a glider plane, developed by Pipistrel, a partner of the RBF4AERO project.

2 Continuous Adjoint Formulation

The derivation of the adjoint equations, their boundary conditions and the SD expression, concerning flows governed by the Navier–Stokes equations coupled with the Spalart–Allmaras model, starts with the definition of the augmented objective function L,

$$L = J + \int_{\Omega} u_i R_i^v d\Omega + \int_{\Omega} q R^p d\Omega + \int_{\Omega} \widetilde{\nu}_a R^{\widetilde{\nu}} d\Omega, \tag{1}$$

where J is the objective function. We assume that J is defined only along the boundary S of the flow domain Ω, so $J = \int_S J_{S,i} n_i dS = \int_{S_W} J_{S_W,i} n_i dS + \int_{S_O} J_{S_O,i} n_i dS$, where $S = S_W \cup S_O$, S_W is the controlled solid wall, S_O any other boundary of Ω and n_i the outward unit normal vector to S. Apart from J, L also includes the integrals of the residuals of the momentum ($R_i^v = 0$), continuity ($R^p = 0$) and turbulence model ($R^{\widetilde{\nu}} = 0$) equations, multiplied by the fields of the adjoint velocities u_i, adjoint pressure q and adjoint turbulence model variable $\widetilde{\nu}_a$, [8]. Dropping the last integral on the r.h.s. of Eq. 1 results in the so-called "frozen turbulence" assumption, which neglects variations in the eddy viscosity field and leads to reduced SD accuracy, possibly even to wrong sensitivity signs [8]. A review of continuous adjoint methods for turbulent flows can be found in [7].

A literature survey shows that continuous adjoint can be formulated in two different ways, which both give the same adjoint field equations and boundary conditions, yet different expressions for the gradient of J with respect to (w.r.t.) b_n, where b_n, $n = 1, \ldots, N$ are the design variables.

The first formulation leads to SD expressions with Field Integrals (FI), including the variations in the spatial coordinates $\mathbf{x}$ w.r.t. $\mathbf{b}$, a.k.a. grid sensitivities. A typical way to compute $\delta\mathbf{x}/\delta\mathbf{b}$ is through finite differences (FD) at a cost that scales linearly with N. The *FI*approach starts by formulating

$$\begin{aligned}\frac{\delta L}{\delta b_n} = \frac{\delta J}{\delta b_n} + \int_{\Omega} \left(u_i \frac{\delta R_i^v}{\delta b_n} + q \frac{\delta R^p}{\delta b_n} + \widetilde{\nu}_a \frac{\delta R^{\widetilde{\nu}}}{\delta b_n} \right) d\Omega \\ + \int_{\Omega} \left(u_i R_i^v + q R^p + \widetilde{\nu}_a R^{\widetilde{\nu}} \right) \frac{\delta (d\Omega)}{\delta b_n},\end{aligned} \tag{2}$$

where the last integral vanishes, since $R_i^v = R^p = R^{\widetilde{\nu}} = 0$ in Ω. By developing Eq. 2, [4], integrals of expressions multiplied by $\delta v_i/\delta b_n$, $\delta p/\delta b_n$ and $\delta\widetilde{\nu}/\delta b_n$ arise. By zeroing those expressions in Ω, the field adjoint equations are formulated [7, 8]:

$$R^q = -\frac{\partial u_j}{\partial x_j} = 0 \tag{3a}$$

$$R_i^u = u_j \frac{\partial v_j}{\partial x_i} - \frac{\partial (v_j u_i)}{\partial x_j} - \frac{\partial \tau_{ij}^a}{\partial x_j} + \frac{\partial q}{\partial x_i} + \widetilde{\nu_a} \frac{\partial \widetilde{\nu}}{\partial x_i} - \frac{\partial}{\partial x_l} \left(\widetilde{\nu_a} \widetilde{\nu} \frac{\mathscr{C}_Y}{Y} e_{mjk} \frac{\partial v_k}{\partial x_j} e_{mli} \right) = 0, \quad i = 1, 2, 3 \tag{3b}$$

$$R^{\widetilde{\nu_a}} = -\frac{\partial (v_j \widetilde{\nu_a})}{\partial x_j} - \frac{\partial}{\partial x_j} \left[\left(\frac{\nu + \widetilde{\nu}}{\sigma} \right) \frac{\partial \widetilde{\nu_a}}{\partial x_j} \right] + \frac{1}{\sigma} \frac{\partial \widetilde{\nu_a}}{\partial x_j} \frac{\partial \widetilde{\nu}}{\partial x_j} + 2 \frac{c_{b2}}{\sigma} \frac{\partial}{\partial x_j} \left(\widetilde{\nu_a} \frac{\partial \widetilde{\nu}}{\partial x_j} \right) + \widetilde{\nu_a} \widetilde{\nu} \mathscr{C}_{\widetilde{\nu}} + \frac{\partial v_t}{\partial \widetilde{\nu}} \frac{\partial u_i}{\partial x_j} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + (-P(\widetilde{\nu}) + D(\widetilde{\nu}))\, \widetilde{\nu_a} = 0, \tag{3c}$$

where $P(\widetilde{\nu})$ and $D(\widetilde{\nu})$ are the production and dissipation terms of the Spalart–Allmaras RANS turbulence model, τ_{ij}^a are the components of the adjoint stress tensor and functions $\mathscr{C}_Y$, $\mathscr{C}_{\widetilde{\nu}}$ can be found in [8]. Adjoint boundary conditions are derived by zeroing the expressions multiplying $\delta p/\delta b_n$, $\delta v_i/\delta b_n$, $\delta \widetilde{\nu}/\delta b_n$ and $\delta \tau_{ij}/\delta b_n$ in the surface integrals of $\delta L/\delta b_n$ [7]. The remaining terms in $\delta L/\delta b_n$ yield the SD expression

$$\left. \frac{\delta J}{\delta b_n} \right|_{FI} = \int_\Omega \left(-u_i v_j \frac{\partial v_i}{\partial x_k} - u_j \frac{\partial p}{\partial x_k} - \tau_{ij}^a \frac{\partial v_i}{\partial x_k} + u_i \frac{\partial \tau_{ij}}{\partial x_k} + q \frac{\partial v_j}{\partial x_k} + A_{jk}^t \right) \frac{\partial}{\partial x_j} \left(\frac{\delta x_k}{\delta b_n} \right) d\Omega + W(0), \tag{4}$$

where

$$\begin{aligned} W(\phi) = &- \int_{S_W} \left(-u_k n_k + \frac{\partial J_{S_W,k}}{\partial \tau_{lz}} n_k n_l n_z \right) \left(\tau_{ij} \frac{\delta (n_i n_j)}{\delta b_n} + \phi \frac{\partial \tau_{ij}}{\partial x_k} \frac{\delta x_k}{\delta b_n} n_i n_j \right) dS \\ &- \int_{S_W} \frac{\partial J_{S_W,k}}{\partial \tau_{lz}} n_k t_l^I t_z^I \left(\tau_{ij} \frac{\delta (t_i^I t_j^I)}{\delta b_n} + \phi \frac{\partial \tau_{ij}}{\partial x_k} \frac{\delta x_k}{\delta b_n} t_i^I t_j^I \right) dS + \int_{S_W} J_{S_W,i} \frac{\delta (n_i dS)}{\delta b_n} \\ &- \int_{S_W} \left(\frac{\partial J_{S_W,k}}{\partial \tau_{lz}} n_k (t_l^{II} t_z^I + t_l^I t_z^{II}) \right) \left(\tau_{ij} \frac{\delta (t_i^{II} t_j^I)}{\delta b_n} + \phi \frac{\partial \tau_{ij}}{\partial x_k} \frac{\delta x_k}{\delta b_n} t_i^{II} t_j^I \right) dS \\ &- \int_{S_W} \frac{\partial J_{S_W,k}}{\partial \tau_{lz}} n_k t_l^{II} t_z^{II} \left(\tau_{ij} \frac{\delta (t_i^{II} t_j^{II})}{\delta b_n} + \phi \frac{\partial \tau_{ij}}{\partial x_k} \frac{\delta x_k}{\delta b_n} t_i^{II} t_j^{II} \right) dS \end{aligned} \tag{5}$$

and t_i^I, t_i^{II} are the components of the tangential-to-the-surface unit vectors (in 3D shapes). In Eq. 4, one should notice the presence of the field integral containing the spatial gradient of the grid sensitivities.

The alternative SI formulation, with an SD expression comprised of Surface Integrals, is based on the Leibniz theorem for integral variations, namely

$$\frac{\delta L}{\delta b_n} = \frac{\delta J}{\delta b_n} + \int_{\Omega} \left(u_i \frac{\partial R_i^v}{\partial b_n} + q \frac{\partial R^p}{\partial b_n} + \widetilde{\nu}_a \frac{\partial R^{\widetilde{\nu}}}{\partial b_n} \right) d\Omega + \int_S \left(u_i R_i^v + q R^p + \widetilde{\nu}_a R^{\widetilde{\nu}} \right) n_k \frac{\delta x_k}{\delta b_n} dS. \quad (6)$$

The last integral in Eq. 6 is usually ignored [6], by making the debatable assumption that the flow PDEs are satisfied along the boundary. The *SI* formulation (by excluding the last integral in Eq. 6) in shape optimization, leads to the SD expression

$$\left. \frac{\delta J}{\delta b_n} \right|_{SI} = \int_{S_W} \left[- \left(\tau_{ij}^a n_j - q n_i + \frac{\partial J_{S_W,l}}{\partial v_i} n_l \right) \frac{\partial v_i}{\partial x_k} + \frac{\partial J_{S_W,i}}{\partial x_k} n_i \right] \frac{\delta x_k}{\delta b_n} dS + W(1). \quad (7)$$

The *SI* formulation is, by far, less expensive than the *FI* formulation in problems with many design variables. However, because of the elimination of the last surface integral in Eq. 6, the accuracy of the *SI* formulation is not guaranteed. In contrast, the *FI* formulation provides accurate SD.

A new, third formulation, referred to as the E-*SI* (Enhanced-*SI*) adjoint formulation, was recently proposed in [4] by the NTUA group and is intended to alleviate the accuracy issue of the *SI* formulation, while having almost the same computational cost. Since the E-*SI* formulation was developed after the commencement of the RBF4AERO project, it is not included in this software suite. In Fig. 1, the sensitivity derivatives computed by the three different adjoint formulations are compared in an indicative turbulent flow problem, in which the loss of accuracy caused by the utilization of the *SI* approach is showcased.

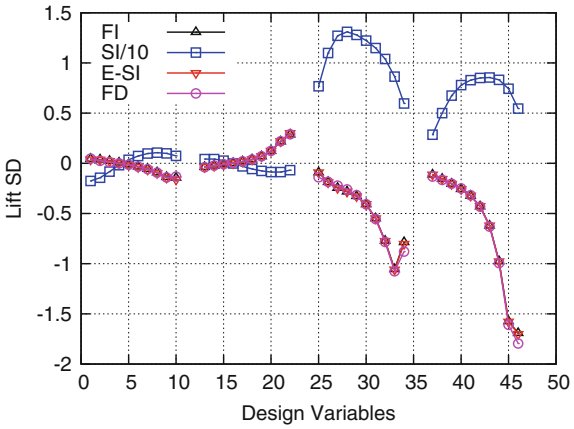


Fig. 1 Turbulent flow around the NACA0012 airfoil ($Re = 10^6$, $a_{\text{inf}} = 3°$, average $y^+ = 0.2$): Comparison of the lift SD computed by the *FI*, *SI*, *E-SI* and FD methods. The SD are computed w.r.t. the x (first half points in the abscissa) and y (second half) coordinates of 24 NURBS control points parameterizing the pressure and suction sides. For scaling reasons, the *SI* results have been divided by 10

3 RBF-Based Morphing

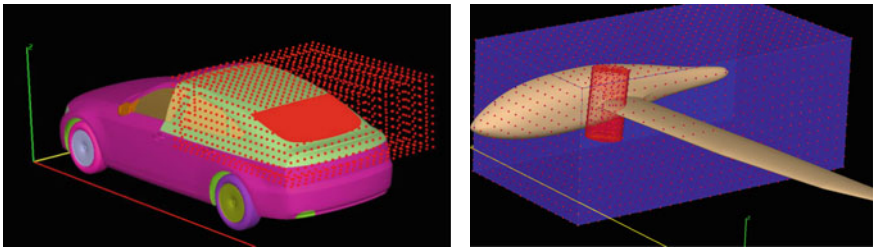
In this section, the mesh morphing algorithm based on RBFs is described. RBFs are mathematical functions able to interpolate data defined at discrete (source) points in an n-dimensional space. The interpolation quality depends on the chosen RBF.

In general, solving the RBF problem requires the solution of a linear system of size M, where M is the source points number. The RBF system solution is computed after defining a set of source points and their displacement. Once the solution has been computed, the displacement of an arbitrary node of the mesh can be expressed as the sum of contributions from all source points. Hence, a desired modification of the mesh nodes position can be rapidly applied preserving mesh topology. RBFs can be classified by the type of support (global or compact) they have, meaning the domain where the chosen RBF is non zero-valued. The interpolation function $s(x) = \sum_{i=1}^{M} \gamma_i \varphi \left(\left\| x - x_{k_i} \right\| \right) + h(x)$ composed of an RBF φ and a polynomial h of order $m-1$, where m is the order of φ, guarantees the compatibility with rigid motions. The degree of the polynomial has to be chosen depending on the kind of the RBF adopted. A radial basis fit exists if the coefficients γ_i and the weights of the polynomial can be found such that the desired function values are obtained at source points and the polynomial terms gives no contributions at source points, i.e., $s(x_{k_i}) = g_i, 1 \leq i \leq M$, $\sum_{i=1}^{M} \gamma_i q(x_{k_i}) = 0$ for all polynomials q with a degree less than or equal to that of polynomial h. The minimal degree of h depends on the choice of the RBF. A unique interpolant exists if the basis function is a conditionally positive definite function [5]. If the RBFs are conditionally positive definite of order $m \leq 2$ [1], a linear polynomial can be used $h(x) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 z$. The subsequent development assumes that the aforementioned hypothesis is valid. The γ and β coefficients are obtained by solving the system for each of the three spatial directions

$$\begin{pmatrix} \mathbf{M} & \mathbf{P} \\ \mathbf{P}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \gamma \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{P} = \begin{pmatrix} 1 & x_{k_1} & y_{k_1} & z_{k_1} \\ 1 & x_{k_2} & y_{k_2} & z_{k_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{k_M} & y_{k_M} & z_{k_M} \end{pmatrix}, \tag{8}$$

where g are the known values at the source points and $\mathbf{M}$ is the interpolation matrix defined by computing all the radial interactions between source points, $M_{ij} = \varphi \left(\left\| x_{k_i} - x_{k_j} \right\| \right), 1 \leq i \leq M, 1 \leq j \leq M$. P is a constraint matrix that arises to balance the polynomial contribution.

The RBF method has several advantages that make it very attractive for mesh smoothing. The key point is that, being a meshless method, only grid points are moved, regardless of which elements are connected to them; this makes the method suitable for parallel implementation. Though meshless, the method is able to exactly prescribe known deformations onto the surface mesh: this is achieved by using all the mesh nodes as RBF centres with prescribed displacements, including the zero field to guarantee that a surface is left untouched by the morphing action.



(a) DrivAer rear window (b) wing-fuselage junction

Fig. 2 Example of RBF points arrangement for the definition of two shape parameters: **a** the height of the rear window of the DrivAer car model is controlled by a cluster of RBF control points; a Box Encapsulation is used to limit the effect of the displacement in the vicinity of the windows; **b** a similar set-up is used to define the deformation of the wing-fuselage junction close to the leading edge

The industrial implementation of RBF morphing poses two challenges: the numerical complexity related to the solution of the RBF problem for a large number of centres and the definition of suitable paradigms to effectively control shapes. RBF Morph deals with both, as it comes with a fast RBF solver capable of fitting large datasets (hundreds of thousands of points in a few minutes) and with a suite of modeling tools allowing the user to set-up shape modifications in an expressive and flexible way. This performance is due to iterative solutions, the Fast Multipole Method and Partition of Unity, as well as shared memory parallelism, the efficiency of which depend on the problem size and has been proven up to 48 cores. RBF Morph allows to extract control points from surfaces and edges, to put points on primitive shapes (boxes, spheres, and cylinders) or to specify them directly by individual coordinates. Two shape modifications used in this study are represented in Fig. 2.

Once the adjoint fields are available, it is possible to compute the SD w.r.t. shape parameters defined by the morphing tool. To take into account the nonlinear fashion of the morphing field, the grid sensitivities are generated through second-order FD of the morphing field around the current design point. It is worth noting that in case the *FI* formulation is used, grid sensitivities are required for the entire grid while, for the *SI* or *E-SI* formulations, these are needed only at the deformable boundaries.

4 Optimization Algorithm

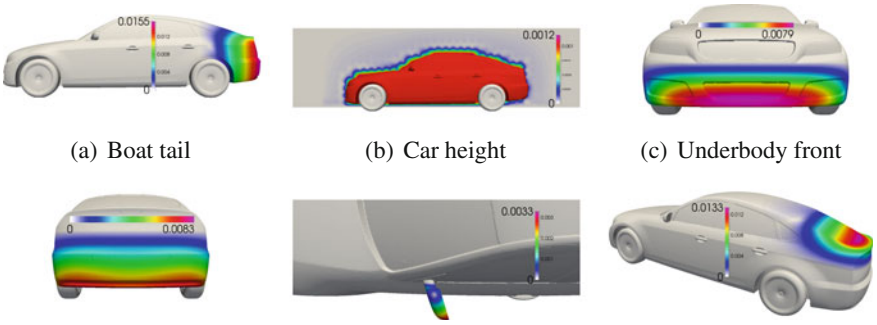
The gradient-based algorithm used to minimize the objective function consists of the following steps: (1) Define the shape modification parameters, and compute the grid sensitivities through FD. These are kept fixed during the entire optimization loop. (2) Solve the flow equations. (3) Compute J. (4) Solve the adjoint equations. (5) Compute the SD. (6) Update the design variables by using a descent method. (7)

Morph the parameterized surface and displace the interior mesh nodes. (8) Unless the stopping criterion is met, go to step 2.

Apart from step 1, which runs in serial using the RBF Morph tools, the rest of the steps are executed within a single OpenFOAM®-based executable, which performs all tasks in parallel. Steps 2 and 4 are the most costly parts of the algorithm, since they require the solutions of PDEs, and have approximately the same cost. If the *FI* formulation is chosen, SD computation can become expensive as well, in case of many (of the order of hundreds) design variables. In case the *E-SI* formulation is used, the cost of computing SD is negligible, as is the cost of the remaining steps.

5 Applications

In the first application, the drag minimization of the DrivAer car model, developed by the Institute of Aerodynamics and Fluid Mechanics of TU Munich [3], is studied. Specifically, the fast-back configuration with a smooth underbody, with mirrors and wheels (F_S_wm_ww) is used. Following the standard practice of the automotive industry, wall functions are used to effect closure on a grid of about 3.8 million cells. Six design variables are defined in total. The part of the car surface parameterized by each of them and the corresponding grid sensitivities are depicted in Fig. 3. The convergence of the optimization algorithm using the *FI* and *SI* adjoint formulations, along with a comparison of the pressure fields between the initial and optimized geometries, is illustrated in Fig. 4. Lowering the rear windshield, creating a spoiler at the end of the trunk and a boat-tail-shaped rear side led to increased pressure on the rear part of the car and, thus, lower drag.



(a) Boat tail (b) Car height (c) Underbody front

(d) Underbody back (e) Mirror rotation (f) Rear window

Fig. 3 DrivAer optimization: Six design parameters are used to morph different parts of the car, by controlling: **a** the boat tail, **b** the car height, **c** the front bumper, **d** the rear bumper, **e** the mirror shape, **f** the rear window shape. In color, one may see $|\delta \mathbf{x}/\delta b_n|$. By computing SD on the initial geometry, the variables depicted in (**a**), (**c**), (**d**) and (**f**) are identified as the ones with the highest optimization potential

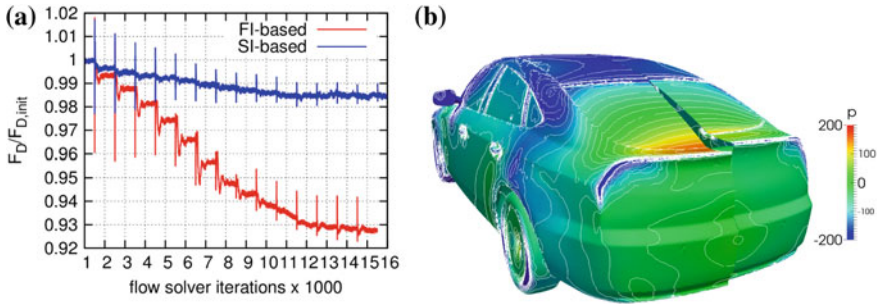


Fig. 4 DrivAer optimization: Left: Evolution of the normalized drag in terms of the number of iterations of the flow solver, following the *FI* and *SI* formulations. In each optimization cycle, the flow solver runs for 1000 iterations. Kinks in the drag value indicate the first iterations after each shape update. With the *FI* formulation, a drag reduction of 7% was achieved, whereas the *SI* gave no more than 1.5% at approximately the same CPU cost. Right: initial (starboard) and optimized (port) (with the *FI* formulation) geometries, colored based on the surface pressure

The second application is concerned with the shape optimization of a glider plane targeting the maximization of the lift-to-drag ratio. The Reynolds number is $Re = 1.55 \times 10^6$ based on the wing chord, the Spalart–Allmaras turbulence model is used, the mesh consists of about 4.7 million cells and the far-field flow angle is 10°. The geometry is parameterized using four RBF-based design variables, depicted in Fig. 5. The FI adjoint formulation is used, the convergence of the optimization

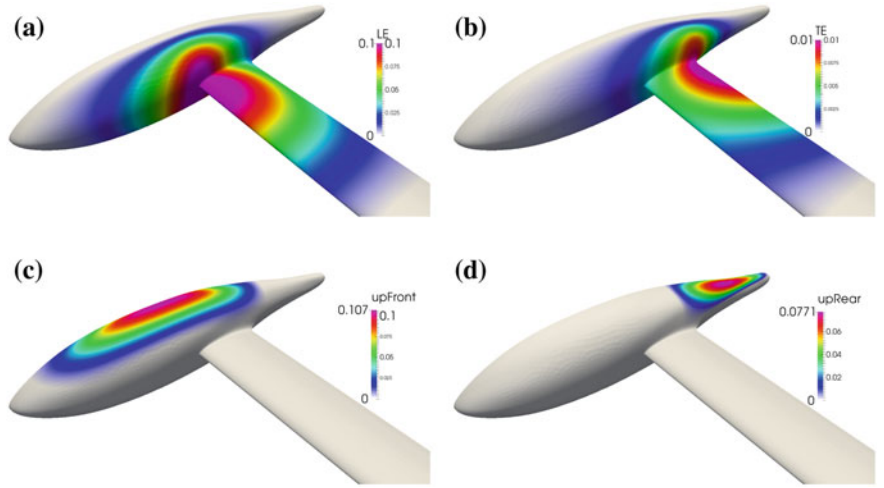


Fig. 5 Glider optimization: the grid sensitivities magnitude for the four design variables. (**a**) and (**b**) parameterize the wing-fuselage junction close to the leading and trailing edges, while (**c**) and (**d**) affect the upper glider surface. All design variables are bounded in order to prevent the generation of non-manufacturable solutions

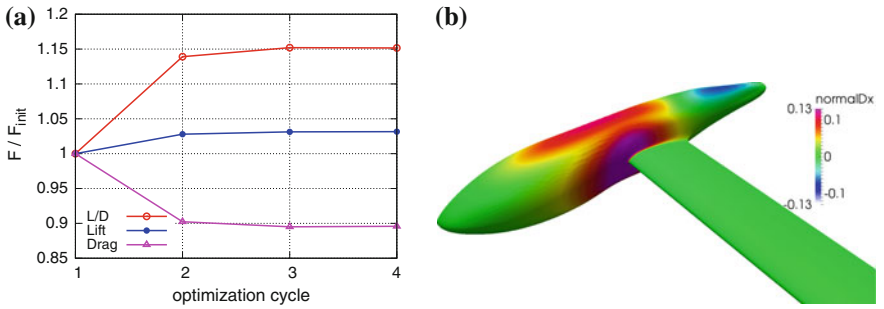


Fig. 6 Glider optimization: **a** convergence of the lift-to-drag ratio (L/D), along with the lift and drag values, normalized with the ones obtained using the initial geometry. A 15% lift-to-drag increase is observed in four optimization cycles by mainly reducing the drag value and slightly increasing lift, **b** the optimized glider geometry, colored based on the projection of the cumulative surface displacement to the surface normal vectors of the initial geometry. Positive numbers indicate an inward displacement while negative ones, an outward movement

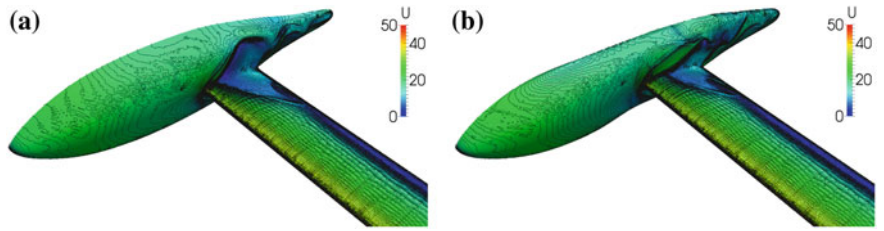


Fig. 7 Glider optimization: near-wall velocity isolines, plotted on the glider surface for the **a** initial and **b** optimized geometries. It can be observed that the low velocity area close to the trailing edge has been considerably reduced

algorithm is shown in Fig. 6a and the optimized geometry is illustrated in Fig. 6b. In Fig. 7, the near-wall velocity isolines are plotted on the glider surface for the initial and optimized geometries.

6 Conclusions

An in-house developed OpenFOAM®-based continuous adjoint solver and an RBF-based morpher, combined into an automated optimization software in the context of a research project funded by the EU, were used as the constituents of a gradient-based optimization algorithm. Two optimization problems of automotive and aerospace engineering were studied, giving a more than 7% drag reduction in the DrivAer case and a 15% lift-to-drag ratio increase in the glider case.

References

1. A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Constructive Approximation*, 5(2):125–134, 2011.
2. M.E. Biancolini. Mesh morphing and smoothing by means of radial basis functions (RBF): A practical example using Fluent and RBF Morph. In *Handbook of Research on Computational Science and Engineering: Theory and Practice (2 vol)*, pages 347–380, 2011.
3. A. Heft, T. Indinger, and N. Adams. Experimental and numerical investigation of the DrivAer model. In *ASME 2012, Symposium on Issues and Perspectives in Automotive Flows*, pages 41–51, Puerto Rico, USA, 8-12 July 2012.
4. I.S Kavvadias, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. On the proper treatment of grid sensitivities in continuous adjoint methods for shape optimization. *Journal of Computational Physics*, 301:1–18, 2015.
5. C. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.
6. D.I. Papadimitriou and K.C. Giannakoglou. A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. *Journal of Computers & Fluids*, 36(2):325–341, 2007.
7. E.M. Papoutsis-Kiachagias and K.C. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering*, 23(2):255–299, 2016.
8. A.S. Zymaris, D.I. Papadimitriou, K.C. Giannakoglou, and C. Othmer. Continuous adjoint approach to the Spalart-Allmaras turbulence model for incompressible flows. *Computers & Fluids*, 38(8):1528–1538, 2009.