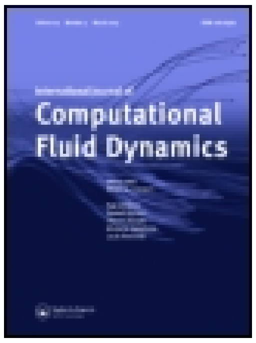


International Journal of Computational Fluid Dynamics

ISSN: 1061-8562 (Print) 1029-0257 (Online) Journal homepage: http://www.tandfonline.com/loi/gcfd20

Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation

F. Salmoiraghi, A. Scardigli, H. Telib & G. Rozza

To cite this article: F. Salmoiraghi, A. Scardigli, H. Telib & G. Rozza (2018): Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation, International Journal of Computational Fluid Dynamics, DOI: 10.1080/10618562.2018.1514115

To link to this article: https://doi.org/10.1080/10618562.2018.1514115

Published online: 06 Sep 2018.

INTERNATIONAL JOURNAL OF COMPUTATIONAL FLUID DYNAMICS
https://doi.org/10.1080/10618562.2018.1514115

Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation

F. Salmoiraghi[a], A. Scardigli[b,c], H. Telib[b] and G. Rozza[a]

[a]SISSA, International School for Advanced Studies, MathLab, Trieste, Italy; [b]OPTIMAD Engineering Srl, Torino, Italy; [c]DISMA – Department of Mathematical Sciences, Politecnico di Torino, Torino, Italy

ABSTRACT
In this work, we provide an integrated pipeline for the model-order reduction of turbulent flows around parametrised geometries in aerodynamics. In particular, free-form deformation is applied for geometry parametrisation, whereas two different reduced-order models based on proper orthogonal decomposition (POD) are employed in order to speed-up the full-order simulations: the first method exploits POD with interpolation, while the second one is based on domain decomposition. For the sampling of the parameter space, we adopt a Greedy strategy coupled with Constrained Centroidal Voronoi Tessellations, in order to guarantee a good compromise between space exploration and exploitation. The proposed framework is tested on an industrially relevant application, i.e. the front-bumper morphing of the DrivAer car model, using the finite-volume method for the full-order resolution of the Reynolds-Averaged Navier–Stokes equations.

ARTICLE HISTORY
Received 8 May 2018
Accepted 16 August 2018

KEYWORDS
Model-order reduction; proper orthogonal decomposition; free-form deformation; aerodynamics; greedy sampling

1. Introduction

We would like to start from a simple calculation to give the motivation of the present work. Let us take a general, fluid dynamics industrial problem. Typically, the Reynolds number requires to exploit turbulence models to solve these kind of problems. The industrial standard is nowadays the Reynolds-averaged Navier–Stokes (RANS) closure model. This algorithm may have a cost of $O(10^2–10^4)$ cpu hours for a standard 3D simulation employing a computational grid of $O(10^7–10^8)$ elements. Even if the license cost is null, the energy consumption cost of modern computers is about 0.05 euro per cpuh, that is, $O(5–5\times10^2)$ euro per each simulation. In an evolutionary optimisation campaign, for non-trivial problems, we perform $O(10^2–10^3)$ simulations that leads to a cost of $O(5\times10^2–5\times10^5)$ euro. Automatic shape optimisation is used only if strategic and addressed in a reasonable amount of time.

Computer performance is constantly improving and the cost of simulations should, theoretically, decrease. In practice, however, it can be observed that numerical models always tend to saturate computational resources, since more complex and accurate models are continuously developed. Computational time does not decrease as computing power increases, and this represents a big challenge for optimisation.

We can rely on the observation of physical phenomena to develop new methodologies to overcome the mentioned limitations. When a physical problem depends on some changing physical/geometrical parameters, it often happens that the solution changes smoothly. Starting every time from scratch to solve the problem is not optimal or even feasible. The intuition is to use a few high-fidelity simulations for some properly selected values of the parameters (related to different configurations) to build a solutions database and then to recycle problem data, solving it by combining the solutions in the database. This is nothing but the rationale that drives the development of several reduced-order methods (ROMs). Thanks to ROMs, we are able to perform simulations of complex phenomena almost in real time, after the construction of the database containing $O(10)$ high-fidelity, indeed very expensive, solutions of the problem for properly selected parameter values.

An overview of different ROMs strategies for industrial and biomedical applications can be found

CONTACT G. Rozza gianluigi.rozza@sissa.it

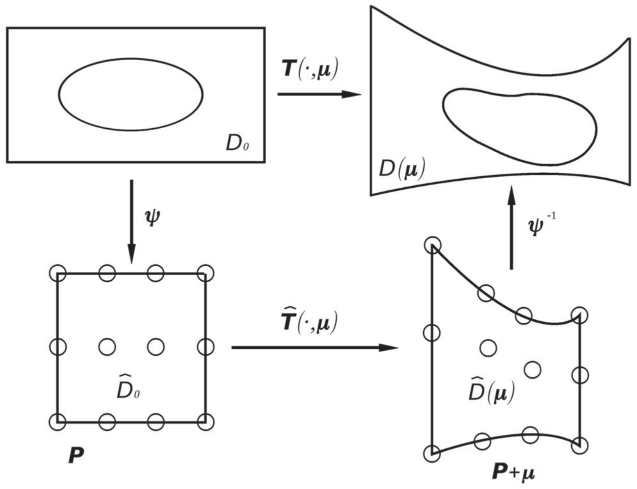
in Salmoiraghi et al. (2016a) and Bergmann et al. (2014).

In the following, we present a new pipeline and different tools for model reduction of complex and industrial problems. In Section 2, we present two different approaches to the geometry parametrisation, exploiting free-form deformation (FFD), then, after Section 3 devoted to full-order methods, in Section 4, we present two different ROMs. In both Sections, we give a theoretical insight and then some algorithm details for the introduced methodology, as well as a properties comparison between them. Finally, in Section 5, we apply the presented methodologies to an automotive industrial benchmark, namely the *DrivAer*[1] model. This model presents interesting features both from the physical and geometrical point of view. In fact, on the one hand, the high Reynolds number requires the use of a RANS model, on the other hand, the geometry is also made up of rotating wheels that cannot be neglected, in order to obtain accurate results, but this increases the complexity of the phenomena involved. Both the turbulence models and rotating mechanical parts represent quite new challenges in the ROM community. This field is quite new in the ROMs community and this is one of the aspect of novelty of the present work.

2. Geometry morphing

In the present work, we provide an approach to perform geometry parametrisation and deformation based on the FFD method, as described in its original version in Sederberg and Scott (1986). Starting from this pioneering work, which is related to solid geometric modelling, FFD has been developed mainly in the frame of computer graphics and only recently employed in aerodynamic shape design problems (see for instance Andreoli, Ales, and Désidéri 2003; Samareh 2004).

Basically, this technique first sets a control lattice surrounding the part of the geometry to be morphed and then deforms the geometry in a continuous and smooth way by moving only the control points of such lattice. This operation is carried out in three steps, shown in Figure 1: first we need to map the actual domain D_0 to the reference one $\hat{D}_0 = [0,1]^d$ (where d is the physical dimension of D_0) through the map $\boldsymbol{\psi} : D_0 \to \hat{D}_0$; second we set a regular grid of unperturbed control points $\boldsymbol{P}$ and we move some control



$\boldsymbol{T}(\cdot,\boldsymbol{\mu})$
D_0
$D(\boldsymbol{\mu})$
$\boldsymbol{\psi}$
$\boldsymbol{\psi}^{-1}$
$\hat{\boldsymbol{T}}(\cdot,\boldsymbol{\mu})$
$\hat{D}_0$
$\hat{D}(\boldsymbol{\mu})$
$\boldsymbol{P}$
$\boldsymbol{P}+\boldsymbol{\mu}$

Figure 1. Sketch of the FFD map construction.

points of a quantity $\boldsymbol{\mu}$ to deform the lattice through the map $\hat{\boldsymbol{T}} : \hat{D}_0 \to \hat{D}$; finally, we map back the resulting domain $\hat{D}$ to the physical domain D through the inverse map $\boldsymbol{\psi}^{-1}$. The FFD map is the composition of the three maps:

$$\boldsymbol{T}(\cdot,\boldsymbol{\mu}) = (\boldsymbol{\psi}^{-1} \circ \hat{\boldsymbol{T}} \circ \boldsymbol{\psi})(\cdot,\boldsymbol{\mu}). \tag{1}$$

For a deeper insight, see Lassila and Rozza 2010; Koshakji, Quarteroni, and Rozza 2013; Forti and Rozza 2014; Sieger, Menzel, and Botsch 2015 for an application very similar to ours. A comprehensive dissertation on different geometry deformation techniques (in particular FFD-based techniques) can be found in Anderson, Aftosmis, and Nemec (2012).

In the present work, we apply the FFD paradigm to morph directly the computational mesh around the geometrical model. In the following, we provide some insight about the general method, whereas in Section 2.2, we focus on its application to our problem, highlighting advantages and drawbacks.

2.1. Morphing features

Let us now provide some general considerations about the FFD technique. Since FFD is independent from the topology of the object to be morphed, it is extremely versatile and suitable to parametrise very complex geometries, including volume meshes, surface triangulations and CAD representations. Moreover, it permits to obtain large deformations as well as small ones. Such strength of the method can turn into a weakness when handling 2D or 3D meshes: the input mesh should be good enough, in terms of number and quality of mesh elements, to guarantee a good description of the deformed geometry. Low-quality inputs will result inevitably in poor deformations. This behaviour can be overcome, for instance, if we perform FFD on the manifold of the geometry, e.g. for the morphing of CAD geometries or as in Salmoiraghi et al. (2016b) for the morphing of isogeometric geometries (NURBS surfaces); alternatively, some curvature-based refinement needs to be introduced for mesh adaptation (see Alliez et al. 2008 for a review of remeshing techniques).

As it is, the method is suited for global shape deformations and when the geometry is unconstrained. Generally speaking, this is not the case for real-life application, where local deformations may be involved as well as the need to impose arbitrary shape constraints. This means that we need some control over the continuity and smoothness of the deformation: even if it is driven by the properties of Bernstein polynomials inside the FFD control box, we can encounter some problems in the interface between the deformed and undeformed part of the domain, such as discontinuities and abrupt changes in the surface curvature. To overcome this issue, we can insert some control points to be kept fixed close to the interface. For the sake of clarity, we provide a simple example. Consider the mesh on a cylinder in Figure 2(a). If we construct the FFD control lattice with three points in the vertical direction and we move only the second one, the results are depicted in Figure 2(b). The mesh in this case has two artificial edges and is $\mathbb{C}_0$ continuous. On the other hand, if we use five control points in the vertical direction and move only the third one, we obtain a smoother, indeed $\mathbb{C}_1$, mesh. A more sophisticated approach (Scardigli et al. 2019) consists in introducing a filter scalar function (with the required order of continuity) that weights the deformation according to the distance from the constraint itself.

Another feature that comes for free with FFD is the straightforward parallelisation of the code. In fact, we can ideally move independently each point of our geometrical object on a different CPU, passing only a limited amount of information among processors. Nevertheless, this may not be the case for the mesh morphing of large-scale problems, where the implementation of deformation constraints and mesh quality controls is more complex.

Thanks to these features, the FFD is a valid option for industrial problems: however, the choice of the preferable FFD approach should depend on the actual application, as pointed out in Section 2.2.

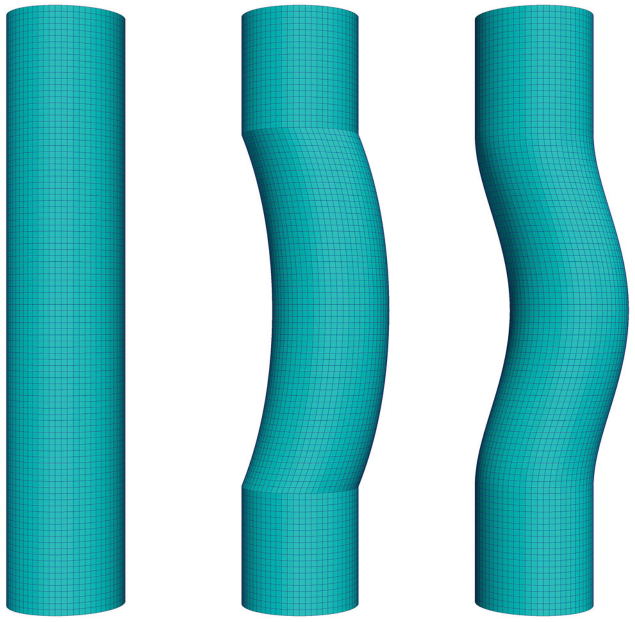


Figure 2. Regularity of the deformation: original mesh (left), $\mathbb{C}_0$ deformation (center) and $\mathbb{C}_1$ deformation (right).

2.2. Mesh morphing strategy

As stated before, we perform FFD directly on a computational mesh around the model. In such a way, provided the mesh for the reference geometry, we extract and map (through FFD) only the coordinates of the vertices, leaving the connectivity and the other properties of the mesh untouched.

In view of reduction strategies, the solution snapshots corresponding to different parameter values need to be defined on the same underlying mesh, in order to build the ROM (see Section 4 for further details). This framework allows us to automatically retrieve the solution on a common mesh, so that we have the field values on the same degrees of freedom without introducing an interpolation between different meshes.

In terms of deformation control, the $\mathbb{C}_0$ continuity between the fixed and the deformable part of the mesh can be easily fulfilled if we do not move the external control points of the FFD bounding box. The non-penetration condition of the cells, instead, is guaranteed by limiting the displacements in order to avoid the overlapping of the control points. However, depending on the deformation type, different drawbacks may affect the cells (e.g. high skewness, high non-orthogonality, etc.) and impair the quality of the simulation results. To prevent such behaviour, a set of application-specific mesh quality constraints has to be satisfied: roughly speaking, the more problematic cells tends to be the ones close to the interface between the deformed and undeformed part of the domain. Increasing the continuity of the deformation allows to overcome this potential problem. This can be done by adding more control points to be kept fixed close to the interface, as explained in Section 2.1. Despite these expedients, guaranteeing the minimum quality of the volume mesh may be infeasible or really difficult to achieve in those cases where non-small deformations are required or the parametrisation of rather complex geometries is involved.

In the case of shape optimisation, aimed at finding the optimal configuration for small deformation of the starting geometry, this drawback is not a great limitation. Moreover, FFD on mesh allows to skip the mesh generation for every new configuration, leading to significant time savings. In fact, even if on large meshes the deformation itself and the quality checks may be time consuming, the mesh generation is usually more demanding in terms of computational time and resources.

This strategy relies on PyGeM[2], which is a *Python* library using Free Form Deformation and Radial Basis Functions to parametrise and morph complex geometries, including CAD representations: such possibility may be useful for large deformations, where the quality of the resulting surfaces is highly dependent on their discretisation, as hinted before.

3. Full-order model

In the present work, the full-order model is represented by the resolution of Navier–Stokes equations (NSE). In broad terms, the NSE are a system of time-dependent, non-linear, partial differential equations which govern the motion of fluids. Further difficulties arise when turbulence is involved, as occurs in many engineering applications: turbulent flows exhibit a chaotic behaviour, characterised by significant and irregular variations in space and time, and their study represents a challenge under both the analytical and numerical point of view (see for instance Pope (2011), for a more comprehensive review of the problem).

Nowadays, there are four main approaches to deal with turbulence: direct numerical simulation (DNS), Reynolds-averaged Navier–Stokes (RANS), Large Eddy simulation (LES) and hybrid LES-RANS models. In DNS, all the scale of motions are solved for one realisation of the flow. Although easy in principle, solving the whole range of spatial and temporal turbulence scales is often not feasible, given the complexity of the phenomena. DNS is indeed very expensive and the computational costs tends to increase cubically with the Reynolds number Re: therefore, this approach can be applied to flows characterised by low or moderate Re, whereas it has prohibitive costs for industrial applications at higher Re. Other techniques for simulating turbulent flows, such as LES (Sagaut 2006; Pope 2011) and hybrid models (Fröhlich and von Terzi 2008), have started to be employed in engineering applications. Nevertheless, the resolution of RANS equations is still the most common approach in industry, especially in early stages of design or during aerodynamic optimisation, when several simulations are required. The general idea behind the RANS approach is to decompose velocity $\boldsymbol{U}$ and pressure p into ensemble-averaged and fluctuating components (Reynolds decomposition), obtaining approximate solutions to the NSE. A turbulence model is required to determine the Reynolds stress, the unknown term which accounts for fluctuations contribution, and to provide closure to the system of equations (Pope 2011).

In the following, we will refer to a finite-volume discretisation of the RANS equations as the high-fidelity/full-order model, as further described in Section 5.1.

4. Reduced-order model

We proposed two different approaches for the construction of the reduced basis, both relying on the proper orthogonal decomposition (POD). The POD allows to extract the modes from a set of solutions of the problem at hand for different values of the parameters. For a deeper insight on POD techniques, see Ravindran 1999; Aubry 1991; Chinesta, Ladeveze, and Cueto 2011; Chinesta and Ladevèze 2014; Chinesta et al. 2017 for the general formulation; see (Benner, Gugercin, and Willcox 2015) for dynamical systems. For the application of POD to the model-order reduction of fluid dynamics problems, we recall Manzoni, Salmoiraghi, and Heltai (2015) for potential flows, (Ballarin et al. 2015; Salmoiraghi et al. 2016b) for viscous flows and (Bui-Thanh, Damodaran, and Willcox 2003) for compressible flows.

4.1. POD at a glance

The idea is to start from a parametric geometrical model and create a database $\Xi = [\boldsymbol{\mu}_1 \mid \dots \mid \boldsymbol{\mu}_M]$ of M parameter values and a database $\boldsymbol{\Theta} = [u(\boldsymbol{\mu}_1) \mid \dots \mid u(\boldsymbol{\mu}_M)]$ of outputs $u(\boldsymbol{\mu}_i)$ thanks to a proper sampling strategy, shown in Section 4.4. Once we have the database we perform the Singular Value Decomposition (SVD) of the sample to extract the POD modes $\boldsymbol{\psi}$

$$\boldsymbol{\Theta} = \boldsymbol{\Psi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T, \tag{2}$$

where $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ are the left and right singular vectors matrices of $\boldsymbol{\Theta}$, and $\boldsymbol{\Sigma}$ is the diagonal matrix containing the singular values in decreasing order. The POD modes $\boldsymbol{\psi}$ are nothing but the first N columns of $\boldsymbol{\Psi} = [\psi_1 \mid \dots \mid \psi_M]$, with $N \le M$.

Alternatively, it is possible to compute the basis through the method of snapshots Sirovich (1987), by solving the equivalent eigenvalue problem (Volkwein 2013)

$$\boldsymbol{\Theta}^T\boldsymbol{\Theta}\phi_i = \lambda_i\phi_i, \tag{3}$$

and setting

$$\psi_i = \frac{1}{\sqrt{\lambda_i}}\boldsymbol{\Theta}\phi_i, \quad \text{for } i = 1, \dots, N. \tag{4}$$

When the snapshot dimension is much greater than M, it is less expensive to solve Problem 3, whereas the first approach is more reliable for badly conditioned matrices (Demmel 1997).

Exploiting the new basis, we can express the reduced solution of the problem as:

$$u^N = \sum_{i=1}^{N} \alpha_i(\boldsymbol{\mu})\psi_i, \tag{5}$$

that is, a combination of the basis functions. The problem shift to find, for each new value of the parameter, the value of the coefficients α_i. Some (Rozza, Huynh, and Patera 2008; Quarteroni, Rozza, and Manzoni 2011; Hesthaven, Rozza, and Stamm 2016) use the weak formulation of the problem and exploit the POD modes as basis functions to find the coefficients (suitable for problems with weak formulation and affine dependence from the parameters), others (Peherstorfer and Willcox 2015b,a) use the measurement of the quantity of interest coming from sensors (suitable for real-time evaluation 'on the field'). In the following, we show two different methods for the POD coefficients evaluation exploiting two different paradigms: proper orthogonal decomposition with interpolation (PODI) and proper orthogonal decomposition combined with domain decomposition techniques (DD-POD).

4.2. PODI

PODI was first introduced by Bui-Thanh (2003) and it has been used recently in aerodynamic applications (Dolci and Arina 2016). The rationale behind PODI regards the evaluation of the POD coefficients by interpolation of the POD coefficients computed for the parameter points $\boldsymbol{\mu}_k \in \Xi$. In these points, the reduced and high-fidelity solutions (u^N and u, respectively) are required to be equal (besides truncation errors), i.e.

$$\forall \boldsymbol{\mu}_k \in \Xi : u(\boldsymbol{\mu}_k) = u^N(\boldsymbol{\mu}_k) = \sum_{i=1}^{N} \alpha_i(\boldsymbol{\mu}_k)\psi_i, \tag{6}$$

thus resulting in an interpolatory method. For each new value of the parameter $\boldsymbol{\mu}_{\text{new}}$, we interpolate the $\alpha_i(\boldsymbol{\mu}_k)$ coefficients to find the new $\alpha_i(\boldsymbol{\mu}_{\text{new}})$ coefficients and evaluate the new reduced solution from Equation (5), i.e.

$$u^N_{\text{new}} = \sum_{i=1}^{N} \alpha_i(\boldsymbol{\mu}_{\text{new}})\psi_i$$

The interpolation is performed exploiting the *Delaunay* triangulation, and its dual *Voronoi* tessellation (see Figure 3), of the parameter space (Fortune 1992; Sartori et al. 2016; Du, Faber, and Gunzburger 1999; Watson 1981). As a consequence, the method can be used efficiently in the N-th dimensional case.

This first model-order reduction strategy relies on EZyRB[3] tool. EZyRB is a *Python* library for model-order reduction based on barycentric triangulation for the selection of the parameter points (see Section 4.4 for algorithm details) and on POD for the selection of the modes. The term easy (EZy) is used with respect to the classical reduced basis method: in RBM, we reconstruct the whole solution of the problem thanks to its weak formulation and an hypothesis of affine dependence from the parameters; in EZyRB, we reconstruct

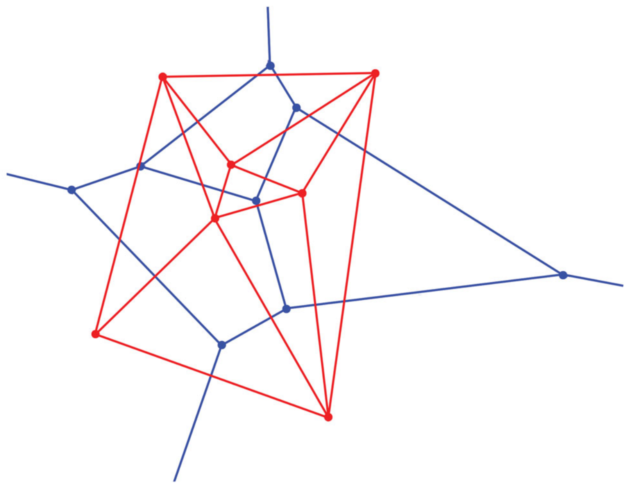


Figure 3. The duality between Voronoi tesselation (wider) and Delaunay triangulation (centered).

only the output of interest thanks to an interpolation of the reduced basis (POD modes) coefficients. It is ideally suited for actual industrial problems, since its structure was designed to handle black-box simulations and can interact with several analysis tools simply providing the output file of the simulations. This tool is suited for the construction of the model-order reduction strategy when we start our sampling from scratch (since it is based on sampling strategy shown in Section 4.4), but it can be used in a suboptimal way even if the database has been already computed in the past.

4.3. DD-POD

The main idea behind the method, as first proposed by Buffoni, Telib, and Iollo (2009), is to split the domain in two parts and use a different approximation method in each region. Generally speaking, POD-based ROMs suffer from the fact that the basis space is a linear combination of the solution space spanned by the snapshots: this means that it is not capable to represent non-linear phenomena, unless a sufficient rich database is provided. To bypass such difficulty, a hybrid low-order/high-order method based on domain decomposition is employed. As shown in Figure 4, the computational domain is decomposed into two regions, i.e. Ω_1 and Ω_2 such that $\Omega_1 \cap \Omega_2 \neq \emptyset$: the canonical CFD solver is used where the effects of non-linearities and geometry variations are predominant (Ω_1), whereas linear and weakly non-linear phenomenology is addressed by the ROM (Ω_2). The two models are coupled through the overlapping region using a Schwarz-type method, resulting in a non-local boundary condition on $\partial\Omega_1$. The main steps of the method are presented in Algorithm 1.

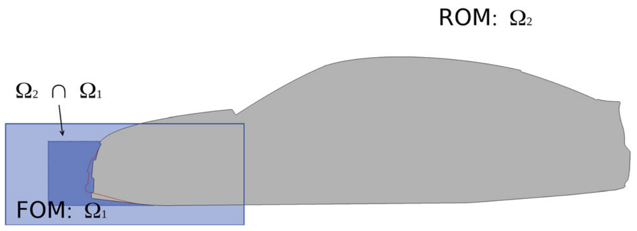


Figure 4. DD-POD example: Ω_1 domain for FOM, Ω_2 domain for ROM and overlapping region $\Omega_1 \cap \Omega_2$.

Algorithm 1 DD-POD algorithm

```
1: set initial boundary conditions for $u_1(\boldsymbol{\mu}_{\text{new}})$ on $\partial\Omega_1$
2: while $convergence$ = false do
3:     evaluate $u_1$ by integrating the governing equations in $\Omega_1$
4:     $\alpha \leftarrow \arg\min_\alpha \left(\| u_1 - \sum_{i=1}^N \alpha_i \psi_i \|_{\Omega_1 \cap \Omega_2}\right)$
5:     $u_2(\boldsymbol{\mu}_{\text{new}})^N \leftarrow \alpha$
6:     evaluate $u_2^N|_{\partial\Omega_1}$
7:     update boundary conditions for $u_1$ on $\partial\Omega_1$
8:     check for $convergence$
9: end while
```

At every solver iteration, the high-order solution u_1 in Ω_1 is used to evaluate the POD coefficients α_i, by solving a least-square problem which minimises the L2-norm of the distance between the CFD and ROM solutions in $\Omega_1 \cap \Omega_2$. This allows to reconstruct the POD solution u_2^N in Ω_2, and in particular its restriction to $\partial\Omega_1$, determining a new set of boundary conditions for the CFD solver.

Such operation has a negligible cost with respect to the iteration and can be integrated in the CFD solver with little effort: for the current application, the algorithm is implemented directly in the *OpenFOAM*® solver.

4.4. Sampling strategy

By construction, the POD basis gives an optimal representation, in terms of energy, of the solution space. This means that the error of both ROMs is ideally zero for each snapshot belonging to the database:

in the limit of no compression, the basis is capable to reproduce exactly the solution. However, a robust ROM should ensure a sufficient accuracy over the entire parameter space, i.e. for each $\boldsymbol{\mu}_{\text{new}}$: to fulfil this requirement, a potentially large number of snapshots may be necessary. Since the computational cost of such evaluations is often prohibitive for industrial applications, the use of an efficient sampling strategy on the parameter space assumes high relevance in this context.

In the present work, we adopt the approach first proposed by Lombardi et al. (2011), which couples Constrained Centroidal Voronoi Tessellations (CCVT) and Greedy methods. In this strategy, new well-spaced points are added iteratively, enriching the database in those regions where a certain error indicator exceeds a fixed tolerance. Our error indicator, i.e. the density function used in the CCVT, is built exploiting a Leave-One-Out Cross-Validation technique. Given an initial database containing the snapshots for the vertices of the parametric domain $\boldsymbol{\Theta}_0 = [u(\boldsymbol{\mu}_1) \mid \ldots \mid u(\boldsymbol{\mu}_{M_0})]$, we compute M_0 POD-bases, one for each $u(\boldsymbol{\mu}_k)$ in $\boldsymbol{\Theta}_0$, by leaving that snapshot out: it is then possible to evaluate $u^N(\boldsymbol{\mu}_k)$ as the projection of $u(\boldsymbol{\mu}_k)$ on the POD-basis spanned by all the remaining snapshots and compute our indicator $e_s = \| u(\boldsymbol{\mu}_k) - u^N(\boldsymbol{\mu}_k) \|$. Such information is used to compute the centroids of the tessellation elements (or the barycentric values of the Delaunay triangulation, which is its dual): among those points, we choose the one where the density function reaches its highest value as the new sampling point. The strategy is summarised in Algorithm 2.

Algorithm 2 Sampling strategy with leave-one-out algorithm

```
1:  $\boldsymbol{\Theta}_0 \leftarrow \boldsymbol{\Xi}_0$
2:  while $\max e_s > tol$ do
3:    for all $\boldsymbol{\mu}_k$ in $\boldsymbol{\Xi}_j$ do
4:      $\boldsymbol{\Theta}_j^{[k]} = \boldsymbol{\Theta}_j \setminus [u(\boldsymbol{\mu}_k)]$
5:      ROM $\leftarrow \boldsymbol{\Theta}_k^{[k]}$
6:      evaluate $u^N(\boldsymbol{\mu}_k)$
7:      error $e_s(k) = \| u(\boldsymbol{\mu}_k) - u^N(\boldsymbol{\mu}_i) \|$
8:    end for
9:    for all simplex in Delaunay triangulation do
10:     error $e_t = \mathtt{Area} * \sum_{\text{vertices}} e_s$
11:   end for
12:   refined simplex $\leftarrow \arg\max e_t$
13:   $\boldsymbol{\mu}_{\text{new}} = \frac{\sum X e_s}{\sum e_s}$
14:   $\boldsymbol{\Xi}_{j+1} = [\boldsymbol{\Xi}_j \mid \boldsymbol{\mu}_{\text{new}}]$
15:   $\boldsymbol{\Theta}_{j+1} = [\boldsymbol{\Theta}_j \mid u(\boldsymbol{\mu}_{\text{new}})]$
16: end while
```

An important drawback of the methodology is represented by the curse of dimensionality. In fact, in order to start the sampling algorithm, we need to compute the solution of all the vertices of the parametric domain, that is, 2^n solutions, where n is the number of parameters involved. This means that the computational cost for the evaluation of $\boldsymbol{\Theta}_0$ will grow exponentially with n and may become prohibitive for large-scale industrial problems. This characteristic is a bottleneck of the present framework. Alternative approaches, based for instance on stochastic processes, may be considered in order to mitigate such problem.

4.5. Critical comparison between the two methods

Let us now provide some considerations and comparisons between the two techniques. Thanks to the simplicity of the method, with PODI we can build the reduced-order model on the solution only in the region of interest (e.g. the surface of the body) and not in the whole computational domain. Therefore, PODI on output is faster and quicker both in the construction and evaluation of the reduced-order model. In fact, during the construction step, we import, assemble and perform SVD on matrices containing the output that commonly are $O(10^{-3})$ smaller than the ones containing the whole solution. During the evaluation step, we only evaluate the new coefficients by interpolation and then we perform a matrix-vector product, which is a trivial operation. On the other hand, with the second approach, we can reconstruct the whole, even non-linear, solution, since it partially solves the high-fidelity problem during the online phase. If the field reconstruction is not sufficiently good, instead of adding new snapshots, it is possible to enlarge the domain where we solve the high-fidelity model and obtain a more accurate result: obviously, the more we extend the inner domain, the less we gain in terms of computation speed-up. For such reasons, the second strategy typically requires fewer snapshots. Moreover, the DD-POD approach does not rely on the parametrisation: given a set of high-fidelity simulations, it can be used straightforwardly even if the snapshots are generated with different parametrisations. This is not the case for the PODI approach, where the geometry parametrisation has to be known in order to perform interpolation. These features give a good flexibility to the second method, but the model reduction is less severe, leading to an online evaluation step slower than the one related to the first strategy.

5. Results on DrivAer model

The results of this integrated approach, from geometrical morphing to model reduction are performed on the *DrivAer* model, which is a realistic generic car model developed by TU Munich in collaboration with Audi AG and BMW Group and made available in several configurations for research purposes. Based on the two medium-size cars (see Heft, Indinger, and Adams 2011, 2012 for further details), the model represents a good compromise between complex production cars and strongly simplified models. Generic models like the Ahmed body (Ahmed, Ramm and Faltin 1984) or the SAE model (Cogotti 1998) are widely used in vehicle aerodynamics to investigate basic flow structures but fail to predict more complex phenomena, due to the oversimplification of the geometries in relevant regions, e.g. rear end, underbody and wheelhouses. On the other hand, real cars are unlikely to be used for validation purposes, due to data access restrictions. In this scenario, the DrivAer model constitutes a solid benchmark for industrial applications, merging a realistic and detailed geometry description with the availability of numerical and experimental validation data.

We adopt a fastback configuration with mirrors, rotating wheels and smooth underbody. All the numerical investigations are carried out with ground simulation and at realistic Reynolds numbers, increasing the complexity of the phenomena: the flow is fully three-dimensional and turbulent, characterised by separation, recirculation and unsteady wakes (Hucho 2013).

5.1. High-fidelity model: simpleFoam

High-fidelity evaluations are carried out using *OpenFOAM*®[4], an open source software for computational fluid dynamics. More specifically, we decide to use *simpleFoam*, the steady-state solver for incompressible flows based on a semi-implicit method for pressure-linked equations (SIMPLE) algorithm, implementing a cell-centred finite volume method for RANS equations (see Ferziger and Peric (2002) for a comprehensive overview of solution methods). In principle, the steady-state solver is not able to describe the unsteady behaviour of the flow: nevertheless, this approach represents a common practice for automotive applications, in order to capture the average aerodynamic performance of a vehicle. It should be noted that the proposed approaches do not rely on the choice of the solution methods and software but can be coupled with any canonical CFD solver.

A three-dimensional hex-dominant mesh of about $15 \cdot 10^6$ cells is generated around the car model by the *snappyHexMesh* utility, introducing symmetry in the longitudinal plane.

The Reynolds number of the simulations, based on the free-stream velocity and on the car length, is set to $4.87 \cdot 10^6$. The Spalart–Allmaras turbulence model (Spalart and Allmaras 1992) is employed and we introduces wall-functions to describe the near-wall flow. Despite such expedients, the computational cost of each high-fidelity evaluation is of $O(10^2)$ cpu hours on a last generation supercomputer, without considering the mesh generation.

Once we get the solution, we post-process it in order to extract the output of interest, namely the pressure on the surface p_w, the wall shear stress $\boldsymbol{\tau}_\mathrm{w}$ and the drag coefficient, defined as

$$C_\mathrm{x} = \frac{\int_S -p\boldsymbol{n} \cdot \boldsymbol{x} + \int_S \boldsymbol{\tau} \cdot \boldsymbol{x}}{\frac{1}{2}\rho U^2 A_\mathrm{f}} \tag{7}$$

where $\boldsymbol{n}$ is the outward-pointing versor normal to the model surface, $\boldsymbol{x}$ and $\boldsymbol{z}$ are the versors in the longitudinal and vertical directions, respectively, ρ is the air density, U the unperturbed speed and A_f is the frontal area of the model. Due to the unsteadiness characterising our application, all the quantities of interest are evaluated as the average over the solver iterations, and convergence is attained when the corresponding series become stationary.

Compared with experimental data (see Heft, Indinger, and Adams 2012), our setup leads to a 5% error on the output of interest, which is considered acceptable for the work purpose (and its applications).

5.2. Geometry morphing

For the problem at hand, we wrap the mesh close to the car front bumper in a lattice of $6 \times 3 \times 4$ control points and we keep fixed all the points except the ones highlighted in red in Figure 5 (points with indices $[n_x, n_y, n_z] = [1, 1, 1] \vee [1, 2, 1] \vee [2, 1, 1] \vee [2, 2, 1] \vee [3, 1, 1] \vee [3, 2, 1]$). Such points are allowed to move by the same quantity in the longitudinal and vertical directions, resulting in a two-dimensional parameter space, $[\mu_1, \mu_2]$, with bounds $[\mu_1, \mu_2]_{min} = [-0.18, -0.3]$ and $[\mu_1, \mu_2]_{max} = [0.18, 0.3]$, i.e. a little less of the distance between two consecutive points in the corresponding direction. The chosen parametrisation guarantees the overall satisfaction of our set of mesh quality constraints.

In Figure 6, we show the resulting deformed mesh for one of the vertices of the parameter space. Since the FFD is applied directly to the volumetric mesh, we do not need to rebuild the mesh for the others configurations, which can be a non-trivial operation: consequently this approach makes easier and faster even the offline evaluation of the problem, presented in the following Section.

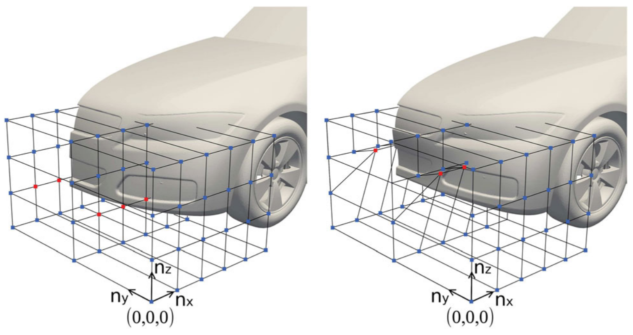


Figure 5. FFD lattice and a possible deformation (due to moved control points): $[\mu_1, \mu_2] = [0.18, 0.30]$.

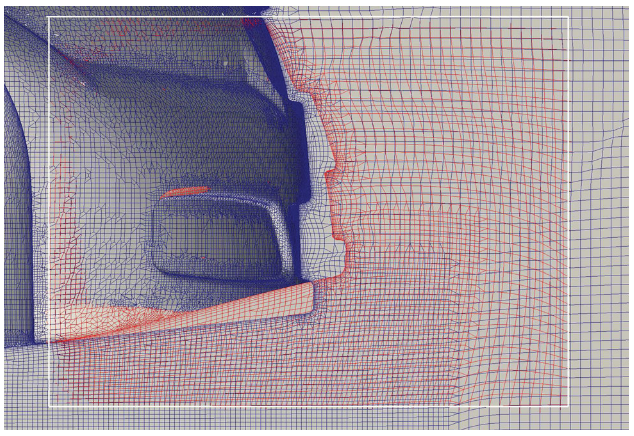


Figure 6. Mesh morphing of the DrivAer front bumper: original mesh (darker lines) vs modified mesh (advanced lighter shape) for $[\mu_1, \mu_2] = [-0.18, 0.3]$. The box identifies the deformed region.

5.3. Offline step

In order to initialise both the algorithms (PODI and DD-POD), we need to evaluate a database of solutions. As shown in Figure 7, we start from a set of 9 points uniformly distributed in the parameter space and we iteratively add more points as chosen by Algorithm 2. Since we are interested in p_w and $\boldsymbol{\tau}_w$ on the car in order to evaluate the drag coefficient, we compute the error estimator e_s using only the restriction of the solution to the surface of the model. The results of the parameter selection are reported in Figures 8–9, where we plot the initial and the final error estimator all over the parametric domain. It is worth noting that the algorithm chooses new parameter values which are very close using an error estimator build either on p_w or on $\boldsymbol{\tau}_w$. This means the physics in a given parametric point can be approximated by the physics in the already computed points in the same way for pressure and stresses. Thus, we decided to evaluate the solution only in the point identified by the pressure error estimator: it is slightly suboptimal but allow us to halve the number of simulations. In Table 1, we report the maximum and average error for each iteration: the most critical zone, i.e. the one characterised by larger errors, is the south-east quadrant (see Figures 8 and 9) but the algorithm tends to explore also regions with fewer points, resulting in average improving of the outputs reconstruction all over the parametric domain.

Once all the 13 solutions are computed, we extract the POD bases we will use in the online step and

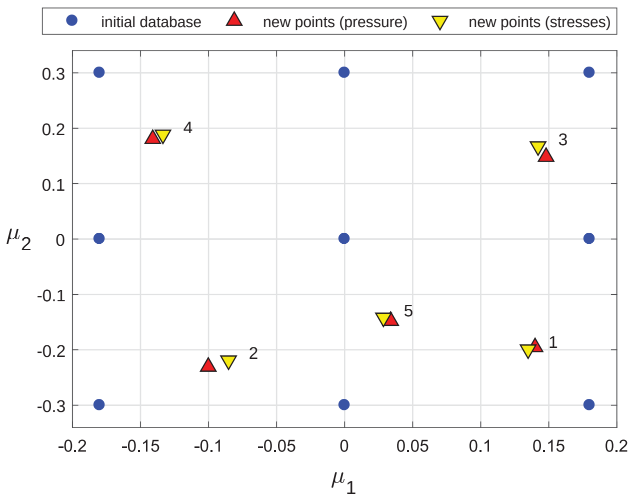
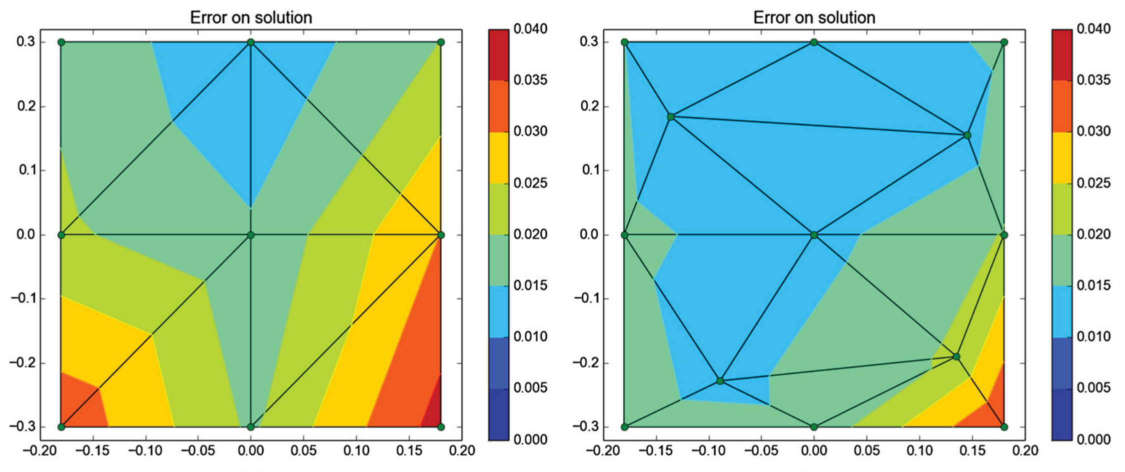
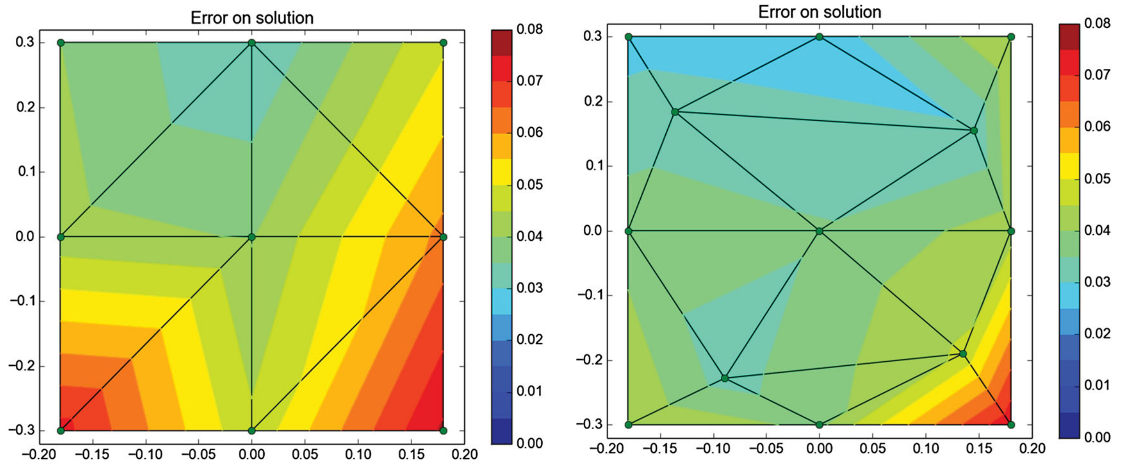


Figure 7. Initial points and new parameter values for each iterations of the sampling algorithm.



(a) (b)

Figure 8. L2 relative error in the parameter space for the pressure field.



(a) (b)

Figure 9. L2 relative error in the parameter space for the stresses field.

we build the triangulation of the POD coefficients over the parametric domain (for the PODI strategy). For the PODI strategy, we build the reduced-order model on the solution only in the region of interest (i.e. the car surface), in order to speed-up the online evaluation, whereas for the DD-POD approach the POD modes are defined in the whole domain (see Figure 10). In PODI, the modes are calculated only for p_{w} and $\boldsymbol{\tau}_{\mathrm{w}}$; in DD-POD, since we use the POD reconstruction to impose the boundary conditions, a separated basis is computed for $p, \boldsymbol{U}$ and Spalart–Allmaras turbulent quantity $\tilde{\nu}$. The decomposition for the DD-POD is chosen according to the criterion proposed by Scardigli et al. (2019), resulting in a domain Ω_1 of about $2 \cdot 10^6$ cells which include the deformable part of the mesh. In Figure 11, we plot both the singular values and the POD eigenvalues for the fields of interest.

Table 1. Error estimator max e_s, average error $\bar{e}_s$ and new parameter values μ for each iteration of the algorithm

	Pressure			Stresses		
Iteration	max e_s	$\bar{e}_s$	$\boldsymbol{\mu}$	max e_s	$\bar{e}_s$	$\boldsymbol{\mu}$
0	0.03692	0.02329	[0.1399, −0.1953]	0.07592	0.05056	[0.1348, −0.1994]
1	0.03697	0.02255	[−0.1002, −0.2305]	0.07528	0.05036	[−0.0853, −0.2193]
2	0.03544	0.01932	[0.1480, 0.1483]	0.07411	0.04474	[0.1421, 0.1673]
3	0.03535	0.01789	[−0.1410, 0.1806]	0.07411	0.04149	[−0.1335, 0.1885]
4	0.03508	0.01684	[0.0338, −0.1477]	0.07408	0.03947	[0.0285, −0.1419]

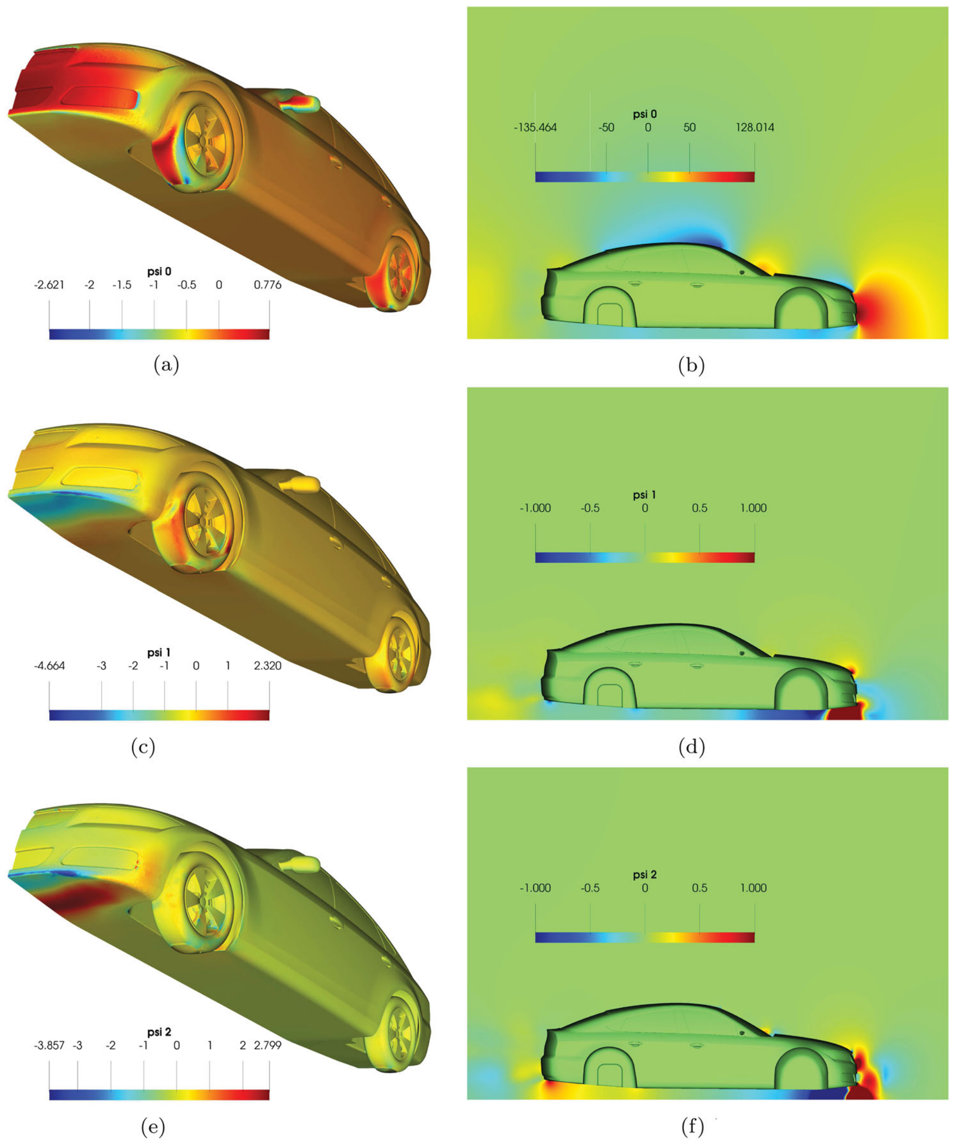


Figure 10. First three POD modes for the pressure: PODI (a,c,e) and DD-POD (b,d,f). (a) PODI: mode 0; (b) DD-POD: mode 0; (c) PODI: mode 1; (d) DD-POD: mode 1; (e) PODI: mode 2; (f) DD-POD: mode 2.

5.4. Online step

In order to validate the reliability of the procedure, we evaluate four configurations out of database, chosen according to a near-random criterion (i.e. latin hypercube sampling). Such configurations are: $\boldsymbol{\mu} = [0.135, -0.19]$, $\boldsymbol{\mu} = [-0.089, -0.228]$, $\boldsymbol{\mu} = [0.145, 0.155]$ and $\boldsymbol{\mu} = [-0.136, 0.184]$.

In Figure 12, we plot the convergence of the averaged errors according to the dimension of the database and, subsequently, the number of POD basis, starting from the initial database of solutions, characterised by $M = 9$. Given the limited number of solution snapshots, in this analysis we do not truncate the POD bases, hence $N = M$. The errors are computed as $\| u - u^N \| \,/\, \| u^{\text{base}} \|$ for the fields of interest and as $|C_{\text{x}} - C_{\text{x}}^N|/|C_{\text{x}}^{\text{base}}|$ for the drag coefficient, normalised with respect to the baseline configuration (i.e. $\boldsymbol{\mu} = [0, 0]$). The errors are bigger for the reconstruction of the fields, in particular the vectorial ones, as $\boldsymbol{U}$ and $\boldsymbol{\tau}_{\text{w}}$. This behaviour is intuitive, since in the vectorial case reduced-order models have to reconstruct not only the magnitude but also the direction. On the other hand, the error is smaller for the drag coefficient, meaning that the integration is compensating the errors, leading to a better approximation.

In Figure 13, we plot the error between the full-order and reduced-order solution for the worst out-of-database configuration. For both methods, the larger

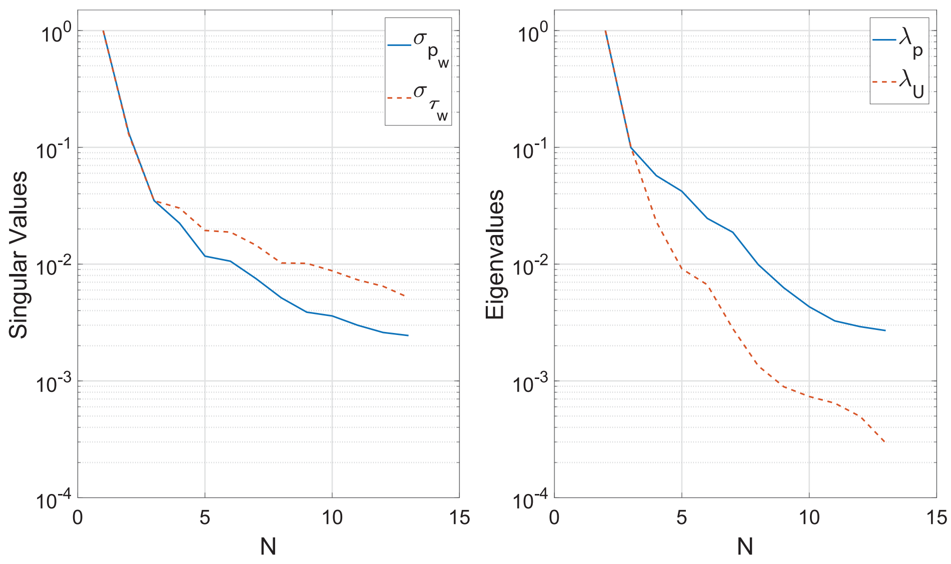


Figure 11. POD Singular Values for p_w and $\boldsymbol{\tau}_w$ (left, PODI approach) vs POD eigenvalues for p and $\boldsymbol{U}$ (right, DD-POD approach).

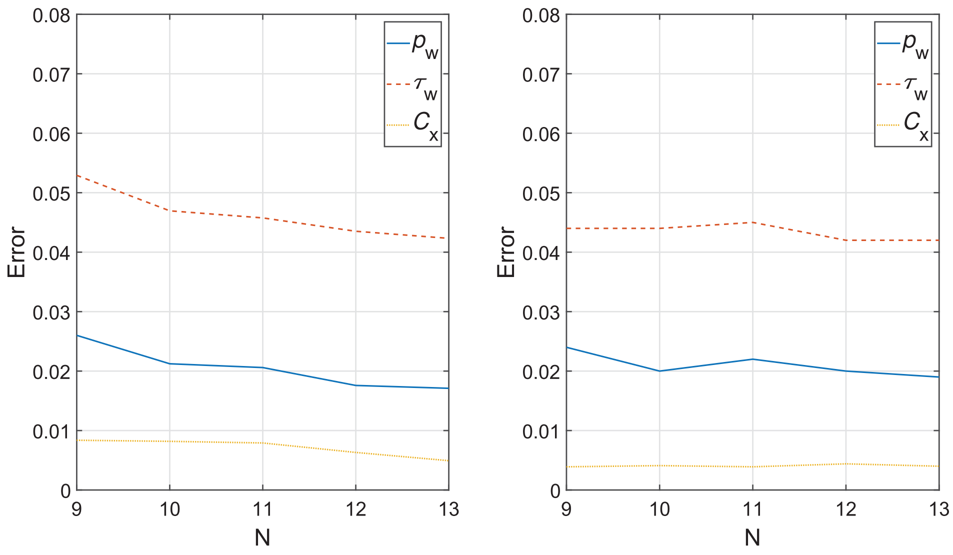
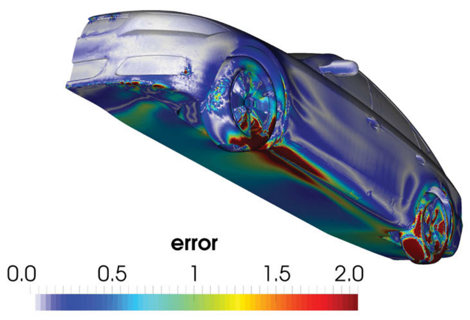


Figure 12. Error convergence enriching the POD bases for the outputs of interest: PODI (left) and DD-POD (right).

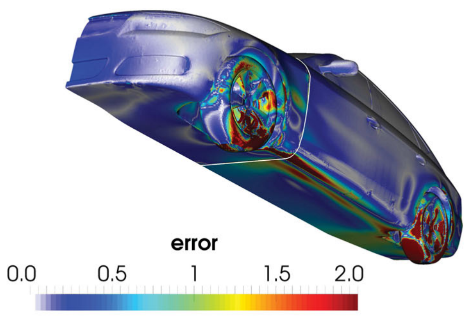
errors are localised near the wheels and their wakes, in addition to the car underbody, where the effects of the bumper variations are more relevant. In this configuration, the DD-POD approach tends to behave better than PODI in the full-order region (especially if we consider $\boldsymbol{\tau}_w$), whereas the outside errors are comparable (Table 2).

6. Conclusions and future work

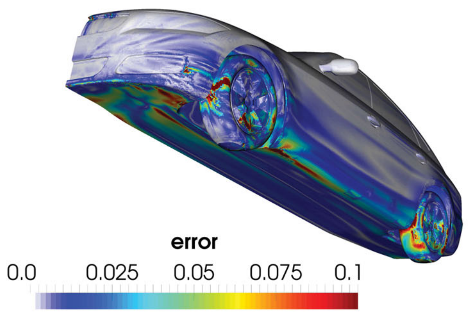
The present work has shown a new complete pipeline for the model-order reduction. First, we have introduced a strategy for the geometry morphing of parametrised shapes, based on direct mesh deformation, showing the strength and weakness of the approach. Then, we have introduced two different strategies for the model-order reduction, both based on a POD techniques but differing in the exploitation of the extracted POD modes. Finally, we have tested and validated the present framework with an industrial benchmark coming from the automotive field, that is, the DrivAer model. In particular, in view of automatic shape optimisation, we have been able to reconstruct the pressure and shear stress field on the surface of the model and drag coefficient with acceptable errors and a considerable speed-up, with respect to the high-fidelity solver, i.e. *OpenFOAM*. Nevertheless, we highlight that both the strategies proposed do not rely on the chosen



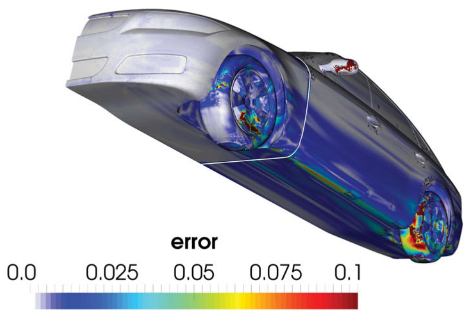
(a) PODI: p_{w} error



(b) DD-POD: p_{w} error



(c) PODI: $\boldsymbol{\tau}_{\mathrm{w}}$ error



(d) DD-POD: $\boldsymbol{\tau}_{\mathrm{w}}$ error

Figure 13. p_{w} and $\boldsymbol{\tau}_{\mathrm{w}}$ errors for the worst (in terms of accuracy) out-of-database configuration. For the DD-POD approach (right) the white line identifies the boundary between the FOM and ROM regions.

Table 2. Results of online step of PODI and DD-POD framework using 13 basis functions.

		PODI	DD-POD
Errors	p_{w}	1.7%	1.9%
	$\boldsymbol{\tau}_{\mathrm{w}}$	4.2%	4.2%
	C_x	0.5%	0.4%
Evaluation cost		$O(10^{-2}s)$	$O(30\ cpuh)$
Computational speed-up		$O(10^7)$	$O(10)$

discretisation method: in particular, the PODI approach treats the high-fidelity solver completely as a black box, whereas the DD-POD one requires only runtime access to the boundary conditions. This feature allows to exploit the user-preferred software, even commercial ones. Thus, this new framework could easily be integrated in every technical design pipeline with not much further effort. Following this rationale, in the next future, we want to apply the present pipeline to other industrial fields, such as, for instance, in naval, nautical and aerospace engineering. These fields, in fact, even if they can appear very different, present very similar features: high Reynolds numbers, complex geometries, rotating mechanical parts (propellers and wheels) and the very same output of interest (pressure and wall shear stresses). The main open issue that we have to tackle and improve is the choice of the starting sampling: the dimension of the initial database can become quickly prohibitive if we want to use a large number of design parameters. Still, the present tool is ready-to-use for several industrial and biomedical applications.

Notes

1. http://www.drivaer.com
2. Python Geometrical Morpher – https://github.com/mathLab/PyGeM/
3. Easy Reduced Basis method – https://github.com/mathLab/EZyRB/
4. http://www.openfoam.org

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the UBE – Underwater Blue Efficiency project (PAR-FSC programme Regione Friuli Venezia Giulia); INDAM-GNCS; and Automobili Lamborghini S.p.A. (2007-2013)

ORCID

G. Rozza http://orcid.org/0000-0002-0810-8812

References

Ahmed, S. R., G. Ramm, and G. Faltin. 1984. *"Some Salient Features of the Time-Averaged Ground Vehicle Wake."* SAE Techincal Paper 840300.

Alliez, P., G. Ucelli, C. Gotsman, and M. Attene. 2008. "Recent Advances in Remeshing of Surfaces." In *Shape Analysis and Structuring,* edited by L. De Floriani and M. Spagnuolo, 53–82. Heidelberg: Springer.

Anderson, G. R., M. J. Aftosmis, and M. Nemec. 2012. "Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design." In 50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, AIAA 2012–965.

Andreoli, M., J. Ales, and J.-A. Désidéri. 2003. "Free-form-deformation Parameterization for Multilevel 3D Shape Optimization in Aerodynamics." PhD thesis, INRIA.

Aubry, N. 1991. "On the Hidden Beauty of the Proper Orthogonal Decomposition." *Theoretical and Computational Fluid Dynamics* 2: 339–352.

Ballarin, F., A. Manzoni, A. Quarteroni, and G. Rozza. 2015. "Supremizer Stabilization of POD–Galerkin Approximation of Parametrized Steady Incompressible Navier–Stokes Equations." *International Journal for Numerical Methods in Engineering* 102 (5): 1136–1161.

Benner, P., S. Gugercin, and K. Willcox. 2015. "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems." *SIAM Review* 57 (4): 483–531.

Bergmann, M., T. Colin, A. Iollo, D. Lombardi, O. Saut, and H. Telib. 2014. "Reduced Order Models at Work in Aeronautics and Medicine." In *Reduced Order Methods for Modeling and Computational Reduction,* edited by A. Quarteroni and G. Rozza, Vol. 9, 305–332. Springer.

Buffoni, M., H. Telib, and A. Iollo. 2009. "Iterative Methods for Model Reduction by Domain Decomposition." *Cumputers & Fluids* 38: 1160–1167.

Bui-Thanh, T. 2003. "Proper Orthogonal Decomposition Extensions and their Applications in Steady Aerodynamics." PhD thesis, Singapore-MIT Alliance.

Bui-Thanh, T., M. Damodaran, and K. Willcox. 2003. "Proper Orthogonal Decomposition Extensions for Parametric Applications in Compressible Aerodynamics." In 21st AIAA Applied Aerodynamics Conference, AIAA 2003–4213.

Chinesta, F., A. Huerta, G. Rozza, and K. Willcox. 2017. *Model Order Reduction*. 2nd ed. New York: John Wiley & Sons. Encyclopedia of Computational Mechanics.

Chinesta, F., and P. Ladevèze. 2014. *Separated Representations and PGD-Based Model Reduction: Fundamentals and Applications. Vol. 554*. Heidelberg: Springer.

Chinesta, F., P. Ladeveze, and E. Cueto. 2011. "A Short Review on Model Order Reduction Based on Proper Generalized Decomposition." *Archives of Computational Methods in Engineering* 18 (4): 395–404.

Cogotti, A. 1998. "A Parametric Study on the Ground Effect of a Simplified Car Model." Philadelphia, US: SAE Techincal Paper 980031.

Demmel, J. W. 1997. *Applied Numerical Linear Algebra*. Philadelphia, PA: SIAM.

Dolci, V., and R. Arina. 2016. "Proper Orthogonal Decomposition as Surrogate Model for Aerodynamic Optimization." *International Journal of Aerospace Engineering* 2016. 3115

Du, Q., V. Faber, and M. Gunzburger. 1999. "Centroidal Voronoi Tessellations: Applications and Algorithms." *SIAM Review* 41 (4): 637–676.

Ferziger, J. H., and M. Peric. 2002. *Computational Methods for Fluid Dynamics*. 3rd ed. New York: Springer-Verlag.

Forti, D., and G. Rozza. 2014. "Efficient Geometrical Parametrisation Techniques of Interfaces for Reduced-Order Modelling: Application to Fluid–Structure Interaction Coupling Problems." *International Journal of Computational Fluid Dynamics* 28 (3–4): 158–169.

Fortune, S. 1992. "Voronoi Diagrams and Delaunay Triangulations." *Computing in Euclidean Geometry* 1: 193–233.

Fröhlich, J., and D. von Terzi. 2008. "Hybrid LES/RANS Methods for the Simulation of Turbulent Flows." *Progress in Aerospace Sciences* 44 (5): 349–377.

Heft, A. I., T. Indinger, and N. A. Adams. 2011. "Investigation of Unsteady Flow Structures in the Wake of a Realistic Generic Car Model." Proceedings of 29th AIAA Applied Aerodynamics Conference, Honolulu, HI, 1–14.

Heft, A. I., T. Indinger, and N. A. Adams. 2012. "Introduction of A New Realistic Generic Car Model for Aerodynamic Investigations." SAE Techincal Paper 2012-01-0168.

Hesthaven, J. S., G. Rozza, and B. Stamm. 2016. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Heidelberg: Springer.

Hucho, W.-H. 2013. *Aerodynamics of Road Vehicles: From Fluid Mechanics to Vehicle Engineering*. Amsterdam: Elsevier.

Koshakji, A., A. Quarteroni, and G. Rozza. 2013. "Free Form Deformation Techniques Applied to 3D Shape Optimization Problems." *Communications in Applied and Industrial Mathematics* 4. doi:10.1685/journal.caim.452.

Lassila, T., and G. Rozza. 2010. "Parametric Free-form Shape Design with PDE Models and Reduced Basis Method." *Computer Methods in Applied Mechanics and Engineering* 199 (23–24): 1583–1592.

Lombardi, E., M. Bergmann, S. Camarri, and A. Iollo. 2011. "Low-order Models. Optimal Sampling and Linearized Control Strategies." *Journal Europèen des Systèmes Automatisès* 45 (7–10): 575–593.

Manzoni, A., F. Salmoiraghi, and L. Heltai. 2015. "Reduced Basis Isogeometric Methods (RB-IGA) for the Real-Time Simulation of Potential Flows about Parametrized NACA Airfoils." *Computer Methods in Applied Mechanics and Engineering* 284: 1147–1180.

Peherstorfer, B., and K. Willcox. 2015a. "Dynamic Data-Driven Reduced-Order Models." *Computer Methods in Applied Mechanics and Engineering* 291: 21–41.

Peherstorfer, B., and K. Willcox. 2015b. "Online Adaptive Model Reduction for Nonlinear Systems Via Low-Rank

Updates." *SIAM Journal on Scientific Computing* 37 (4): A2123–A2150.

Pope, S. B. 2011. *Turbulent Flows*. Cambridge, UK: Cambridge University Press.

Quarteroni, A., G. Rozza, and A. Manzoni. 2011. "Certified Reduced Basis Approximation for Parametrized Partial Differential Equations and Applications." *Journal of Mathematics in Industry* 1 (1): 3.

Ravindran, S. S. 1999. "Proper Orthogonal Decomposition in Optimal Control of Fluids." *International Journal for Numerical Methods in Fluids* 34: 425–448.

Rozza, G., D. B. P. Huynh, and A. T. Patera. 2008. "Reduced Basis Approximation and a Posteriori Error Estimation for Affinely Parametrized Elliptic Coercive Partial Differential Equations." *Archives of Computational Methods in Engineering* 15 (3): 229–275.

Sagaut, P. 2006. *Large Eddy Simulation for Incompressible Flows: An Introduction*. New York: Springer-Verlag.

Salmoiraghi, F., F. Ballarin, G. Corsi, A. Mola, M. Tezzele, and G. Rozza. 2016a. "Advances in Geometrical Parametrization and Reduced Order Models and Methods for Computational Fluid Dynamics Problems in Applied Sciences and Engineering: Overview and Perspectives." In Proceedings of ECCOMAS Congress, Crete, Greece, ECCOMAS. June.

Salmoiraghi, F., F. Ballarin, L. Heltai, and G. Rozza. 2016b. "Isogeometric Analysis-Based Reduced Order Modelling for Incompressible Linear Viscous Flows in Parametrized Shapes." *Advanced Modeling and Simulation in Engineering Sciences* 3 (1): 21.

Samareh, J. 2004 "Aerodynamic Shape Optimization Based on Free-form Deformation." In 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA 2014–4630.

Sartori, A., A. Cammi, L. Luzzi, and G. Rozza. 2016. "A Reduced Basis Approach for Modeling the Movement of Nuclear Reactor Control Rods." *Journal of Nuclear Engineering and Radiation Science* 2 (2): 021019–021019. 8.

Scardigli, A., R. Arpa, A. Chiarini, and H. Telib. 2019. "Enabling of Large Scale Aerodynamic Shape Optimization Through POD-Based Reduced-Order Modeling and Free Form Deformation." In *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences,* edited by J. Periaux, N. Gauger, K. Giannakoglou, E. Minisci, M. Vasile, and D. Quagliarella, Vol. 48, 49–63. NewYork: Springer.

Sederberg, Thomas W. and Scott R. Parry. 1986. "Free-form Deformation of Solid Geometric Models." *ACM SIGGRAPH Computer Graphics* 20 (4): 151–160.

Sieger, D., S. Menzel, and M. Botsch. 2015. "On Shape Deformation Techniques for Simulation-Based Design Optimization." In *New Challenges in Grid Generation and Adaptivity for Scientific Computing,* edited by S. Perotto and L. Formaggia, Vol. 5, 281–303. Milano: Springer.

Sirovich, L. 1987. "Turbulence and the Dynamics of Coherent Structures. Part I, II and III." *Quarterly of Applied Mathematics* 45 (3): 561–590.

Spalart, P. R., and S. R. Allmaras. 1992. "A One-Equation Turbulence Model for Aerodynamic Flows." In 30th Aerospace Sciences Meeting and Exhibit, 439.

Volkwein, S. 2013. "Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling." Lecture Notes, University of Konstanz, Department of Mathematics and Statistics.

Watson, D. F. 1981. "Computing the N-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes." *The Computer Journal* 24 (2): 167–172.