OPTIMAD

Doctoral Dissertation
Doctoral Program in Pure and Applied Mathematics (30^{th}cycle)

Aerodynamic Shape Optimization through Reduced-Order Modelling in Industrial Problems

By

Angela Scardigli

******

Supervisor(s):
Prof. Claudio Canuto, Politecnico di Torino
Haysam Telib, Optimad Engineering

Doctoral Examination Committee:
Prof. Davide Carlo Ambrosi, Politecnico di Torino
Prof. Giovanni Naldi, Università degli Studi di Milano
Prof. Sandra Pieraccini, Politecnico di Torino
Prof. Gianluigi Rozza, Referee, Scuola Internazionale Superiore di Studi Avanzati

Politecnico di Torino - Università degli Studi di Torino
2018

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Angela Scardigli
2018

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo) and Università degli Studi di Torino.

Acknowledgements

This work has been funded through the PhD programme in Apprenticeship of Regione Piemonte and supported by Optimad Engineering.

The author would like to acknowledge Automobili Lamborghini S.p.A. for supporting part of this research and CINECA SCAI for providing the necessary HPC resources for part of the simulations. Parts of this work were carried out in the framework of the UBE–Underwater Blue Efficiency project (PAR-FSC programme Regione Friuli Venezia Giulia) and FORTISSIMO project (European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement No. 609029).

The author is also grateful to Angelo Iollo, Michel Bergmann and Andrea Ferrero, from Institut Mathématique Bordeaux INRIA Sud-Ouest, for the scientific discussion.

Finally, special thanks to the Optimad team, for the human and scientific support, and in particular to Rocco Arpa and Edoardo Lombardi, for their precious contribution to this work.

Abstract

In this thesis, a framework for aerodynamic shape optimization of large-scale industrial problems, based on Free-Form Deformation (FFD) parameterization techniques and model-order reduction, is presented. In particular, we address the definition of an accurate Reduced-Order Model (ROM), to be used as surrogate model during the optimization process.

The non-linearity and complexity of the Navier-Stokes Equations (NSE) make the use of ROMs in computational fluid dynamics extremely challenging for real-life parameterized problems, especially for turbulent flow conditions. In order to overcome some of these difficulties, we present a hybrid full-order/reduced-order strategy for model order reduction based on Proper Orthogonal Decomposition (POD) and domain decomposition, that relegates the effects of non-linearities and geometry variations to the canonical discretization of the NSE in a reduced subdomain, whereas linear and weakly non-linear phenomenology is addressed by the ROM. The two models are then coupled in an overlapping region through a modified Schwarz method, resulting in a non-local boundary condition for the full-order solver. We discuss the convergence and stability properties of the algorithm, as well as the capability of the model to perform predictive simulations. It can be shown that the approximation error depends on both the choice of the decomposition and the sampling of the parameters space. In order to identify the most critical regions within the reference domain, we propose a leave-one-out cross-validation strategy, which consists in iteratively projecting one snapshots onto the subspace spanned by the others. This allows us to build an out-of-sample estimate that can be employed both for identifying a suitable decomposition and for efficiently sampling the space, when coupled with greedy methods.

The effectiveness and drawbacks of these approaches are highlighted on large-scale problems representative of real-life applications, such as the aerodynamic shape optimization of automotive and naval components subject to physical and geometrical constraints.

Contents

List of Figures

List of Figures

List of Tables

List of Tables

Nomenclature

Acronyms / Abbreviations

Re	Reynolds number
ABL	Atmospheric Boundary Layer
CCVT	Constrained Centroidal Voronoi Tessellation
DEIM	Discrete Empirical Interpolation Method
DES	Detached Eddy Simulation
DNS	Direct Numerical Simulation
EFF	Expected Feasibility Function
EGO	Efficient Global Optimization
EIF	Expected Improvement Function
EIM	Empirical Interpolation Method
FFD	Free-Form Deformation
FOMs	Full Order Models
GP	Gaussian Process
LES	Large Eddy Simulation
MLE	Maximum Likelihood Estimation
MPE	Missing Point Estimation
NSE	Navier-Stokes Equations

ODE Ordinary Differential Equation

OpenFOAM Open Source Field Operation and Manipulation

PGD Proper Generalised Decomposition

POD Proper Orthogonal Decomposition

PODI POD with Interpolation

RANS Reynolds-Averaged Navier-Stokes

RB Reduced Basis

RIC Relative Information Content

ROMs Reduced Order Models

RSM Reynolds stress models

SBG Surrogate-based Global Optimization

SIMPLE semi-implicit method for pressure-linked equations

SVD Singular Value Decomposition

TVD Total Variation Diminishing

Chapter 1

Introduction

Despite the constant improvement of computer performance and simulation techniques, a variety of challenging problems from different branches of science and engineering still remain intractable. Real-life applications usually require to cope with complex and high-dimensional problems in state space, physical space or parameter space, e.g. for large-scale dynamical systems, control systems and optimization, but standard techniques fail to solve these problems efficiently. In this framework, Model Order Reduction represents a new simulation paradigm in computational science and engineering. Generally speaking, the rationale behind the development of Reduced Order Models (ROMs), is to replace complex models, namely Full Order Models (FOMs), with far simpler ones, that allow one to capture the features of the problem being modelled with a given level of accuracy and at a very competitive cost. By enabling near real-time analysis of complex problems, ROMs open unexplored possibilities in numerical simulation, process and shape optimization, simulation-based control, uncertainty quantification and propagation, finding immediate use in relevant industrial applications.

Nowadays, several ROM approaches are available, including empirical and semi-empirical methods such as Proper Orthogonal Decomposition (POD) (Lumley (1967), Sirovich (1987), Benner et al. (2015)), Proper Generalised Decomposition (PGD) (Chinesta et al. (2013)), Reduced Basis (RB) (Maday et al. (2002), Prud'Homme et al. (2001), Quarteroni and Rozza (2014)), Dynamic Mode Decomposition (Schmid (2010)), Empirical Interpolation Method (EIM) (Barrault et al. (2004)), Discrete Empirical Interpolation Method (DEIM) (Chaturantabut and Sorensen (2010)) and hierarchical model reduction (Perotto et al. (2017)). Such approaches are gaining remarkable attention in both the academic and industrial framework. The complexity of the fluid motion for turbulent

flows, however, makes the usage of these techniques extremely challenging for real-life industrial applications, that constitute the main focus of this thesis, especially in the framework of automotive and naval aerodynamic shape optimization.

While a thorough literature is available for simplified problems, the investigation of complex turbulent flows is still rather limited, starting to be a topic of interest in the ROMs community only recently. An overview of different reduction strategies for computational fluid dynamics, including aeronautics, biomedicine and naval problems can be found for instance in Bergmann et al. (2014) and Salmoiraghi et al. (2016a). Generally speaking, the main bottleneck of the above-mentioned methods is that the number of full-order numerical solutions, required to obtain sufficiently accurate predictive results, strongly depends on the non-linearities of the problem and on the size of the design space. The cost associated with this training phase is often not properly addressed, but it cannot be overlooked in an industrial context. In addition, the proposed solutions often required a very specific tuning of the ROM on the application of interest, making its usage harder for non-expert end-users.

Further details about the definition of the problem of interest and the motivations for this thesis, as well as its contents, will be outlined in the following sections.

1.1 Problem definition

Given a vector of P real parameters $\boldsymbol{\mu} := (\mu_1, \mu_2, \cdots, \mu_P)$ varying in a compact set $\mathcal{M} \subset \mathbb{R}^P$, we want to solve a generally-constrained non-linear programming problem, which can be expressed as follows:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{\mu}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \mathbf{g}(\boldsymbol{\mu}) \leq \mathbf{g}_u \\ & \mathbf{h}(\boldsymbol{\mu}) = \mathbf{h}_e \\ & \boldsymbol{\mu}_l \leq \boldsymbol{\mu} \leq \boldsymbol{\mu}_u \end{aligned} \tag{1.1}$$

where f denotes the objective function, whereas $\mathbf{g}$ and $\mathbf{h}$ represent the inequality constraints and equality constraints, respectively. Inequalities between vectors have to be intended component-wise. In this context $\boldsymbol{\mu}$ is usually referred to as the vector of design variables. For the problems of interest, i.e. aerodynamic shape optimizations problems, the design parameters are those controlling the deformation of geometrical components, whereas the objective function is given by some quantity of interest depending on the

flow around the geometry. This means that at least for every full-order evaluation, we need to solve a fluid dynamic problem. In mathematical terms, it translates in having a family $\Omega(\boldsymbol{\mu})$ of open bounded domains in $\mathbb{R}^D$, with $D = 2, 3$, where we want to solve the equations governing the fluid motion, i.e. the Navier-Stokes Equations (NSE). As shown in Figure 1.1, we assume that only a portion of the boundary depends on the parameter vector $\boldsymbol{\mu}$, whereas the remaining part is independent. In addition, we suppose that the boundary $\partial\Omega(\boldsymbol{\mu})$ of each domain is sufficiently smooth. Denoting by $\mathbf{u} = \mathbf{u}(\boldsymbol{\mu})$ and $p = p(\boldsymbol{\mu})$ the velocity and pressure fields, respectively, we can write the equations and the set of suitable boundary conditions in abstract form as:

$$\begin{aligned} NS(\mathbf{u}, p) &= 0 \qquad \text{in} \quad \Omega(\boldsymbol{\mu}), \\ B(\mathbf{u}, p) &= 0 \qquad \text{on} \quad \partial\Omega(\boldsymbol{\mu}), \end{aligned} \tag{1.2}$$

with an additional initial condition for unsteady flows. Details about the numerical approximation of the NSE and on the discretization methods will be provided in Chapter 3. It is worth noting that in many applications of interest, the flows are fully three-dimensional and turbulent, characterised by rather complex phenomena such as separation, recirculation, shear layers and unsteady wakes. For these reasons, solving the whole range of spatial and temporal scales is often infeasible and additional models need to be introduced in order to simplify the problem. Nevertheless, we still have to deal with large-scale problems, that usually required a computational grid of $O(10^7) - O(10^8)$ elements, in order to obtain a numerical solution which is sufficiently accurate. Thus, these so called *high-fidelity* evaluations are typically very expensive, with a computational cost of approximately $O(10^2) - O(10^3)$ cpu hours.

In terms of geometry parameterization, the models are complex and detailed, characterized by several mechanical parts, including moving components, e.g. rotating wheels. During the shape optimization process, these geometries are deformed locally, using arbitrary parameterizations with a number of design variables which typically ranges from ~ 10 to ~ 20 for automotive and naval applications. In addition, they have to fulfil a wide range of constraints, both geometrical, dictated for instance by manufacturing or operational requirements, and aerodynamic, e.g. specifications on lift, drag, load balance, etc..

Under this premise, if we consider a standard global optimization based on evolutionary algorithms, which typically require $O(10^2) - O(10^3)$ high-fidelity evaluations, we can estimate the optimization cost at $O(10^4) - O(10^6)$ cpu hours. Such cost is usually not affordable in industry, also considering that the computing resources are often sized for

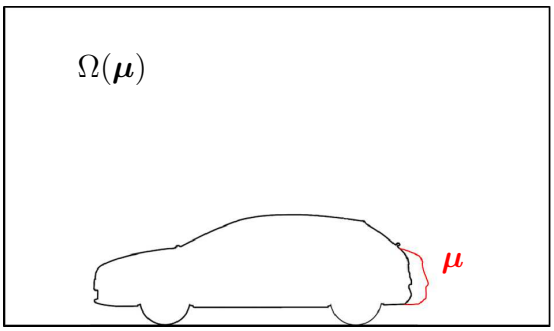


Fig. 1.1 Example of domain definition: the parameter dependency enters only in a limited portion of the boundary (red line).

analysis, rather than optimization. In order to address the problem in a reasonable amount of time and resources, two aspects become crucial: the definition of a versatile and concise parameterization technique and the development of reliable surrogate models. In this framework, the possibility to employ ROMs to speed-up the computations would be extremely advantageous: given the generality and complexity of the problem, however, achieving a sufficient cost reduction is a non-trivial task.

1.2 Thesis outline

Chapter 2 introduces the parameterization techniques used within the thesis, whereas the FOM details, including equations discretization and numerical tools, are presented in Chapter 3.

Then, in Chapter 4, we outline the benchmark test case that will be used in the following chapters to assess the performance and properties of the proposed strategies.

Chapter 5, instead, is devoted to the exhaustive description of the proposed ROM and its numerical properties, and it also includes a literature review of model order reduction techniques, with particular focus on turbulent flows. The accuracy of the method is later addressed in Chapter 6. In order to provide a better understanding of the employed techniques, in both Chapters 5 and 6 we discuss numerical results on the benchmark test case.

The optimization algorithms employed are briefly reported in Chapter 7.

In Chapter 8, we highlight the effectiveness and drawbacks of the presented methodology on two large-scale industrial problems, i.e. the aerodynamic shape optimization of the front bumper of a car model and the mainsail thrust optimization of a sailing boat.

Finally, conclusions and perspectives are collected in Chapter 9.

Chapter 2

Geometry Parameterization

Part of the work described in this chapter has been previously submitted for publication in Scardigli et al. (2019) and Salmoiraghi et al. (2018).

In this chapter we discuss geometry parameterization.

2.1 Parameterization

A crucial aspect of shape optimization problems is represented by the choice of a suitable parameterization technique describing the optimal shape. Such parameterization should be easy to use, versatile and concise: ideally, it should be able to describe a wide range of complex shapes using as few parameters as possible. For shape optimization problems, low-dimensionality is an extremely relevant feature, because geometrical parameters translate directly into design parameters, whose number heavily affects the cost of the optimization process.

Some of the most used approaches, as reviewed in Samareh (2001), include the following ones:

- **Discrete Approach**, where the design variables coincide with the coordinates of the boundary points. This method has the advantage to be easy to implement. Nevertheless, it is usually applied only with adjoint formulations, since an excessive refinement of the geometry is required in order to maintain smoothness. As a consequence, the number of design variables often becomes large, leading to high computational costs and stiff optimization problem (Mohammadi and Pironneau (2010)), which make the method unfeasible for real-life applications.

- **Analytical Approach**: in this formulation, basis shape functions are added to the baseline shape, modelling the geometry through a suitable linear combination:

$$D = B + \sum_i w_i W_i$$

where B and D are respectively the baseline and design shape, whereas W_i represents the design perturbations. The main drawback of this method consists in the definition of a good set of shape perturbations, which may not be an easy task. The approach is very good for wing and airfoil parameterizations, as well as suitable for Multidisciplinary Design Optimization, but it can be difficult to generalize for complex geometries.

- **Polynomial and Spline Approach**, where the shape is described by polynomial and spline curves, leading to a significant reduction of the total number of design variables. The shape is controlled by a limited set of points, namely the control points, whose position in the space defines the shape of the curve itself. Curve types include Bézier, B-splines and NURBS (Non-Uniform Rational B-Spline), and those methods are extremely popular in CAD applications and suitable for shape optimization problems, as shown in several works (Samareh (2001), Andreoli et al. (2003), Désidéri et al. (2007)). However, complex shapes tends to require a large number of control points, reducing the efficacy of the approach and limiting the range of geometries that can be represented.

- **Free-Form Deformation (FFD) Approach**. The FFD is a parameterization technique that has been developed mainly in the frame of Computer Graphics, starting from the original work by Sederberg and Parry (1986): it permits to deform computer-generated objects, regardless of their representation. The basic idea behind the method is to wrap a solid geometric model in a lattice of control nodes and then to deform the geometry in a continuous and smooth way by moving only the control points of such lattice. As a consequence, the whole space embedded in the lattice is manipulated, resulting in the deformation of the object (see Figure 2.1).

  Nowadays the method is widely used by CAE programs for geometrical modelling and it has been employed in the context of aerodynamic shape design problems, for example in aeronautical (Andreoli et al. (2003) and Désidéri et al. (2007)) and, more recently, automotive applications (Sieger et al. (2015)).

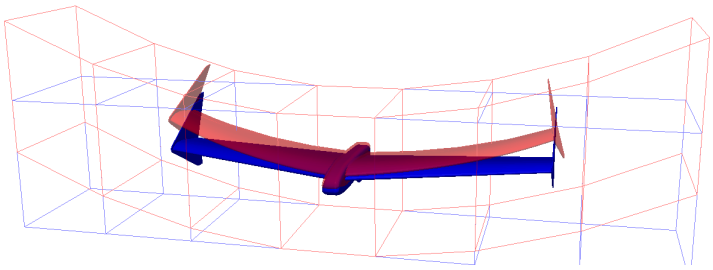


Fig. 2.1 Example of FFD applied to an unmanned aerial vehicle: original shape and lattice (blue) vs deformed ones (red). *Courtesy of Optimad Engineering.*

In addition to these standard techniques for shape parameterization, more physically intuitive quantities may be used, such as radii, thicknesses, angles, lengths, etc., or methods based on parameters that are directly linked to the component geometry, like PARSEC parameters for airfoils and wings description (Sobieczky (1997)). For more complex geometries and non-standard components, however, this may not be an option.

In the present work, we perform geometry parameterization and deformation through an approach derived from the FFD method. Further details are provided in the following sections. For a comprehensive dissertation on different geometry deformation techniques and in particular FFD-based methods, see for instance Anderson et al. (2012).

2.2 Free-Form Deformation

As stated before, the FFD does not manipulate directly the geometry as happens in other parameterization techniques, but acts on the control lattice built around the three-dimensional object. The lattice is basically a Bézier volume, given by the composition of Bézier tensor patches, but also B-spline or NURBS parameterization can be employed. In particular, extensions of FFD to NURBS basis functions allow a non-uniform distribution of the control points in the lattices, providing more flexibility to the method. Generally, the lattices are based on hexahedra, cylinders or spheres which can be easily mapped on their unit elemental primitives, following the steps described in Section 2.2.1.

With respect to other parameterizations based on polynomial curves, that by construction can describe only smooth objects (see Andreoli et al. (2003)), the FFD method uses the Bézier parameterization to represent just the deformation rather than the shape itself. Doing so, the object will be deformed smoothly, even if its original shape is not smooth. In this so-called *Bézier delta formulation*, control points loose the geometrical meaning of position, and the correlation between the displacements and the resulting

deformed geometry is less intuitive. Despite this, the FFD parameterization shows a good accuracy and sensitivity with a low number of parameters (see for instance Amoiralis and Nikolos (2008)).

Since it does not depend on the topology of the geometrical object to be morphed, the FFD is extremely versatile and can be applied to parameterize very complex geometries, including volume meshes, surface triangulations and CAD representations, featuring a good trade-off between simplicity and generality. Moreover, it is suitable for both small and large deformations, provided that the input geometry is good enough, e.g. in terms of number and quality of mesh elements, to guarantee a good description of the deformed geometry. Low quality inputs will results inevitably in poor deformations. This issue can be worked around, for instance, by performing the FFD directly on the manifold of the geometry, e.g. when morphing CAD geometries in the context of Isogeometric analysis, as in Salmoiraghi et al. (2016b); alternatively, some curvature based refinement needs to be introduced for mesh adaptation (see for instance Alliez et al. (2008) for a review of re-meshing techniques) when the deformation becomes large, further complicating this approach.

In the framework of shape optimization, it is also worth stressing that reachable shapes depend on some user-defined parameters, like the number and disposition of control points. Such choices define the mathematical subspace for the optimization: consequently, poor and naive setups will result in a best solution that is far from being optimal for the *true* physical problem. Adaptation procedures (Désidéri et al. (2007); Duvigneau (2006)) may be used to increase the robustness of the process.

In its original formulation, the method is suited for global shape deformations and unconstrained geometries. When dealing with real-life problems, however, it must be taken into account that industrial components have to fulfil a wide range of geometrical constraints, such as manufacturing or operational requirements. Moreover, parts are often deformed locally and thus some continuity constraints between the deformable and the fixed regions must be prescribed. Thanks to the properties of the Bernstein polynomials, the FFD preserves the smoothness of the shape and the continuity of its derivative within the lattice, but there is no guarantee on the boundary. As many other classical parameterization techniques, the method is designed to manipulate the geometrical object globally rather than locally: bounding the displacements of the control points, as proposed in the original paper by Sederberg and Parry (1986), allows one to achieve the desired continuity on the boundaries of the deformed part, as well as to limit the deformation, but this approach is neither intuitive nor efficient for shapes and

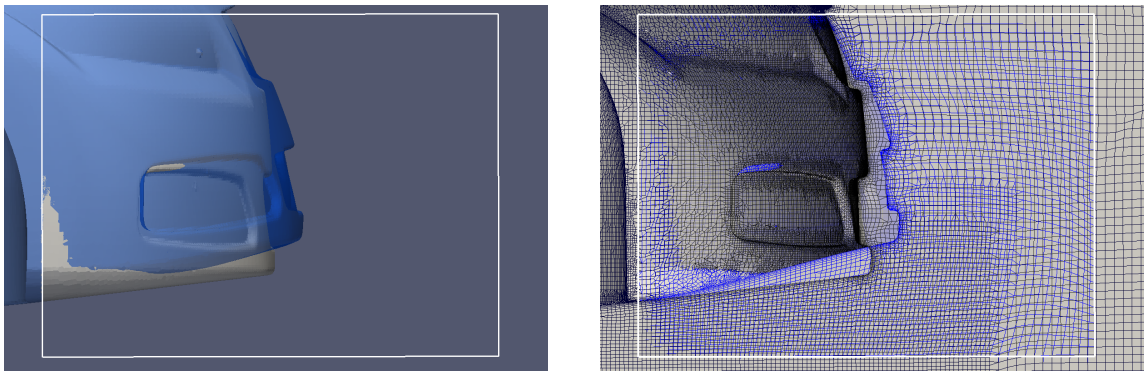
constraints of arbitrary complexity, and it represents one of the main criticisms of the method. In order to overcome this issue, we employ an extension of the FFD, based on level-set methods. As presented in Scardigli et al. (2019), it is possible to efficiently compute the approximate geodesic distance from the constraints through heat kernels and to use this information to locally weight the deformation field, as depicted in Section 2.2.2.

When dealing with discretized geometries, as happens frequently in the industrial context, there are two preferable FFD approaches (see Figure 2.2):

- **Mesh Morphing**. Performing the FFD directly on a volume mesh can be extremely convenient for optimization problems, since it allows to skip the mesh generation phase for every new configuration investigated, leading to significant time savings. On large computational meshes, the deformation itself may be time consuming and difficult to setup, but the mesh generation is usually more demanding in terms of resources. The possibility to have topologically equivalent grids is also interesting in view of model reduction strategies, since the solutions will have the same degrees of freedom and the mapping among different meshes is trivial, without the need of introducing interpolations.

  In terms of deformation control, the FFD does not affect elements connectivity and the non-penetration condition of the cells can be guaranteed by limiting the displacements in order to avoid the overlapping of the control points. Nevertheless, depending on the deformation type, different problems may affect the cells (e.g. high skewness, high non-orthogonality, degenerated cells) and impair the quality of the simulation results. To prevent such behaviours, application-specific mesh quality constraints and deformation propagation strategies have to be implemented. Despite this effort, 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 parameterization of rather complex geometries is involved.

- **Surface Morphing**. In this second option, the FFD is applied on the surface tessellation of the model. As a consequence, the volume mesh needs to be generated at every new parameter evaluation. This approach is then usually more expensive than the previous one, even though the FFD works on a smaller subset. In terms of model order reduction, dealing with meshes with varying topology implies that different solutions are defined on different degrees of freedom: in order to build the reduced model the solutions have to be mapped on a reference domain and an



(a) Surface morphing. (b) Mesh morphing.

Fig. 2.2 Examples of FFD industrial approaches.

additional interpolation step is required, further increasing the computational time and the complexity of the ROM definition.

Nevertheless, surface morphing is generally less problematic than mesh morphing. On the one hand, it is more flexible, supporting also big deformations and allowing one to parameterize arbitrarily complex geometries with less effort. On the other hand, deformation and constraints controls are usually easier to implement and to use, a feature which should not be underrated for industrial applications.

Both strategies benefit from the fact that the so-called FFD map (see Section 2.2.1) can be efficiently evaluated by exploiting an *offline-online* paradigm, where the transformation is precomputed *offline* by a symbolic expression which is cheaply evaluated *online* for the actual parameters and coordinates. Another interesting feature of the original FFD algorithm is that it is embarassingly parallel: in fact, we can ideally move independently each point of our geometrical object on a different CPU, without passing any information among processors, since it requires only the points coordinates and not their connectivity. Nevertheless, this may not be the case where more strict controls over the deformation are required, as for the mesh morphing of large-scale problems. In the following chapters (4 and 8), examples of both strategies will be provided and discussed.

A deeper insight over the FFD and its applications can be found for instance in Andreoli et al. (2003); Ballarin et al. (2014); Forti and Rozza (2014); Koshakji et al. (2013); Lassila and Rozza (2010).

2.2.1 Formulation

Here we recall the construction of the FFD map in the three-dimensional case, shown in Figure 2.3. Given a reference (undeformed) domain Ω that we wish to perturb, we enclose it in a control volume $D \supset \Omega$ and we define an affine map $\boldsymbol{\psi} : (x_1, x_2, x_3) \to (t_1, t_2, t_2)$, so that it is possible to transform the volume D into the unit cube, i.e. $\boldsymbol{\psi} : D \to \hat{D}$ with $\hat{D} = [0,1] \times [0,1] \times [0,1]$. The FFD will be defined later with respect to the coordinates (t_1, t_2, t_2) of the unit cube. Let us introduce a regular grid of $(K+1) \times (L+1) \times (M+1)$ control points over $\hat{D}$

$$\mathbf{p}_{klm} = \begin{bmatrix} k/K \\ l/L \\ m/M \end{bmatrix}, \qquad \text{with } k = 0, \dots, K, \quad l = 0, \dots, L, \quad m = 0, \dots, M$$

and the displacement vector $\boldsymbol{\mu}_{klm} \in \mathbb{R}^3$ of each control point, so that the perturbed point becomes

$$\mathbf{p}^{\star}_{klm}(\boldsymbol{\mu}_{klm}) = \mathbf{p}_{klm} + \boldsymbol{\mu}_{klm}.$$

The deformation of the unitary cube is then obtained through the map $\hat{\mathbf{T}}_{\text{FFD}}(\cdot; \boldsymbol{\mu}) : \hat{D} \to \hat{D}^{\star}(\boldsymbol{\mu})$, defined as

$$\hat{\mathbf{T}}_{\text{FFD}}((t_1, t_2, t_3); \boldsymbol{\mu}) = \sum_{k=0}^{K} \sum_{l=0}^{L} \sum_{m=0}^{M} B_k^K(t_1) B_l^L(t_2) B_m^M(t_3) \mathbf{p}^{\star}_{klm}(\boldsymbol{\mu}_{klm}),$$

where the terms $B_j^J(t) = \binom{J}{j} t^j (j-t)^{(J-j)}$ are the one-dimensional Bernstein base polynomials of the tensor product.

Finally, the FFD map is defined as the composition of the three maps:

$$\mathbf{T}_{\text{FFD}}(\cdot; \boldsymbol{\mu}) = (\boldsymbol{\psi}^{-1} \circ \hat{\mathbf{T}}_{\text{FFD}} \circ \boldsymbol{\psi})(\cdot, \boldsymbol{\mu}).$$

2.2.2 Local deformations

The main idea is to efficiently compute the approximate geodesic distance from the constraints through heat kernels (Crane et al. (2013)) and to use this information to locally weight the deformation field.

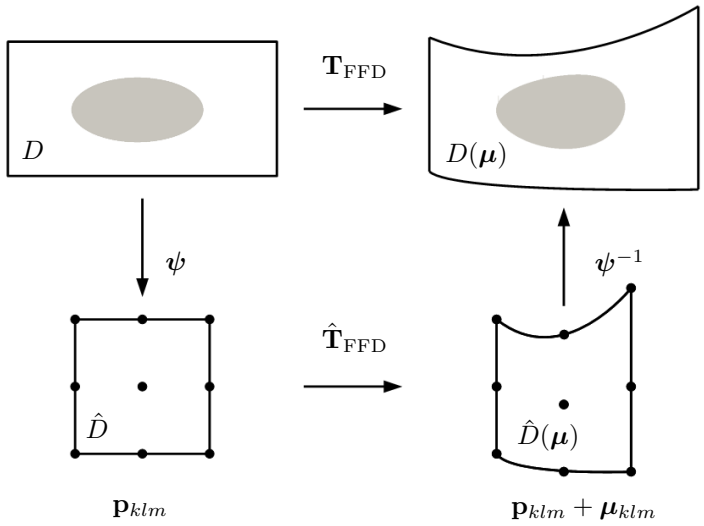


Fig. 2.3 Sketch of the FFD map construction.

Given a set of points $\mathbf{X} = [x_1, \ldots, x_N] \in \mathbb{R}^{D \times N}$, representing for instance a tessellated surface Ω, the deformed set $\mathbf{X}^\star$ can be expressed as:

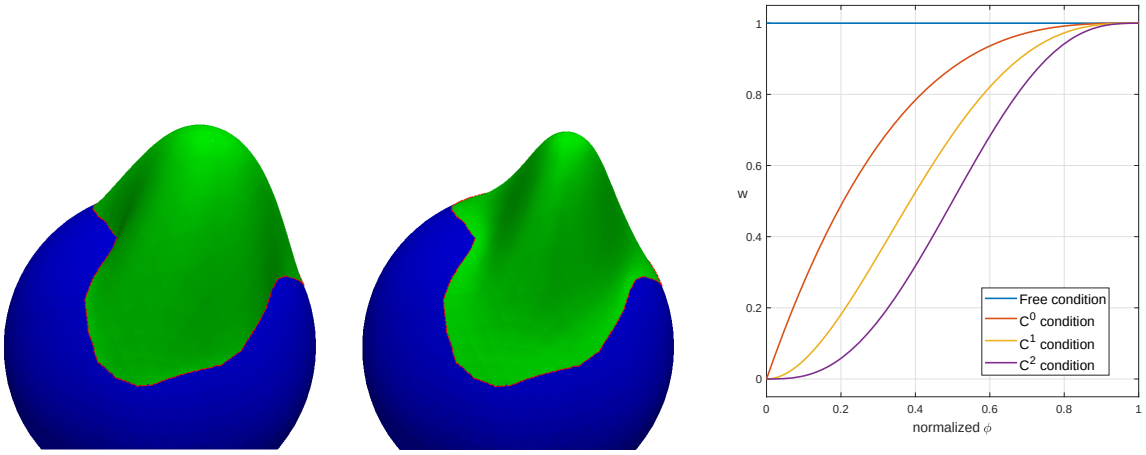
$$\mathbf{X}^\star = \mathbf{X} + \mathbf{T}_{\text{FFD}}(\boldsymbol{\mu}),$$

where $\mathbf{T}_{\text{FFD}}(\boldsymbol{\mu})$ denotes the displacements fields imposed by the FFD through the $\mathbf{p}_{klm}$ control points.

We want to restrict the effect of the deformation on a subregion $\tilde{\Omega} \subseteq \Omega$ and guarantee a certain continuity constraint (C^k, with $k = 0, 1, \ldots, K$) at the interface between these regions, denoted by $\partial\tilde{\Omega}$. In the classical FFD approach, continuity between the deformable and the fixed geometry regions is usually achieved by bounding the displacements of the control points close to the interface, as shown in Sederberg and Parry (1986). Although easy in principle, this approach is neither intuitive nor efficient, particularly for complex geometries and lattices of arbitrary shape, and it tends to increase the number of control points.

A possible solution to this problem consists in introducing a filter scalar function w to weight the deformation, with the following features:

$$\begin{cases} w = 0 & \text{in } \Omega/\tilde{\Omega} \\ 0 \leq w \leq 1 & \text{in } \tilde{\Omega} \\ w, D^k w = 0 & \text{on } \partial\tilde{\Omega} \end{cases} \tag{2.1}$$



(a) C^0 continuity. (b) C^2 continuity. (c) Deformation weight functions.

Fig. 2.4 Examples of deformation of a sphere imposing different continuity constraints on the boundary of the deformable part (green).

and dependent on the function $f(\mathbf{X}_{|\partial\tilde{\Omega}})$, representing the topological information of each point of $\tilde{\Omega}$ with respect to the boundary $\partial\tilde{\Omega}$. This implies that f vanishes at the boundary and that given x_1, x_2 in $\mathbf{X}$, $f(x_1) > f(x_2)$ if x_2 is closer to $\partial\tilde{\Omega}$ than x_1. Therefore, a natural candidate for f is the geodesic distance function evaluated from the boundary, i.e. $\phi(\mathbf{X}_{|\partial\tilde{\Omega}})$. The new constrained deformation can be expressed as:

$$\mathbf{X}^\star = \mathbf{X} + w(\phi(\mathbf{X}_{|\partial\tilde{\Omega}}))\mathbf{T}_{\text{FFD}}(\boldsymbol{\mu}). \tag{2.2}$$

By differentiating Equation (2.2), it is then possible to define the set of properties required to assure the desired continuity of the deformation.

Some of these requirements are easily guaranteed. Since the FFD parameterizes directly the deformation, rather than the shape itself, we know that the deformation field will always be smooth, thanks to the properties of the Bézier curves, that are continuous to any order by construction. On the other hand, the technique preserves the continuity order of the original shape, meaning that the final geometry cannot be smoother than the original one. The weight functions w, instead, are chosen to fulfil the requirements expressed in Equation (2.1). In Figure 2.4 an example of resulting deformations with different continuities on the interface is presented, as well as a set of suitable weight functions (see Figure 2.4c).

The exact geodesic function can be efficiently evaluated through a fast marching technique (see Sethian (1999) for a comprehensive review of level-set and fast marching

methods). However, it may present some discontinuities on the gradients depending on the shape of $\partial\tilde{\Omega}$, resulting in a deformation fields that will be at most C^0 continuous (see for instance Figure 2.5a). In order to improve smoothness, a more regular distance field must be determined. To this end, we substitute the exact geodesic distance with an approximate definition, following the method proposed by Crane et al. (2013), based on the solution of a time-dependent heat equation over the three-dimensional surface mesh. The method consists of three main steps:

1. integrate the heat equation $\frac{\partial u}{\partial t} = \Delta u$ in $\tilde{\Omega}$, up to some time $t^\star$, with zero initial conditions and non-homogeneous Dirichlet boundary conditions on $\partial\tilde{\Omega}$, i.e. by imposing $u(t=0) = 0$ and $u|_{\partial\tilde{\Omega}} = u_D$;

2. evaluate the normalized temperature gradient, in order to get the unit vector field $\mathbf{Z} = -\frac{\nabla u}{\|\nabla u\|}$, pointing along the geodesics;

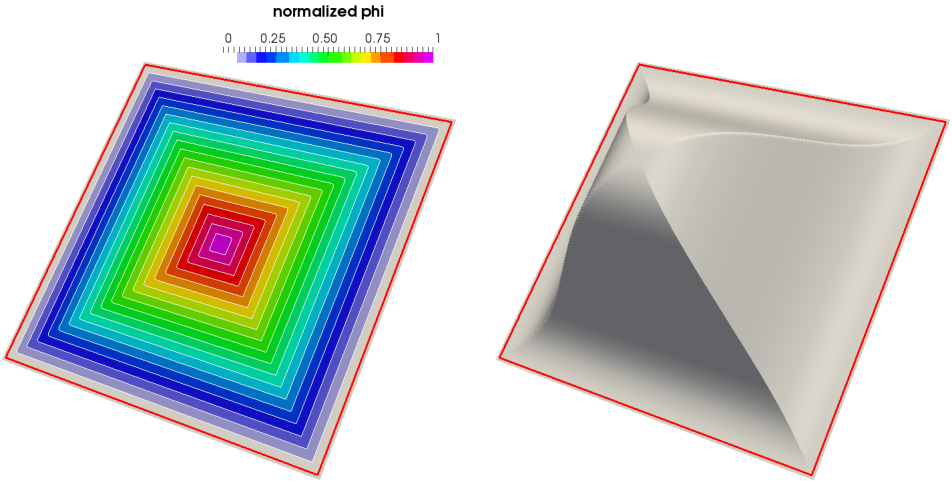
3. solve the Poisson equation $\Delta\phi = \nabla \cdot \mathbf{Z}$ with zero boundary conditions on $\partial\tilde{\Omega}$, to recover the distance.

The desired smoothness of the level set solution is achieved by tuning conveniently the heat propagation time, $t^\star$. A larger time will result in a smoother deformation field, whereas in the limit of $t^\star \to 0$ the approximate geodesic distance tends to the exact one. A comparison of deformations is presented in Figure 2.5: due to the irregularity on the surface contour the exact level-set is not smooth, but this issue can be overcome via the heat kernel method.

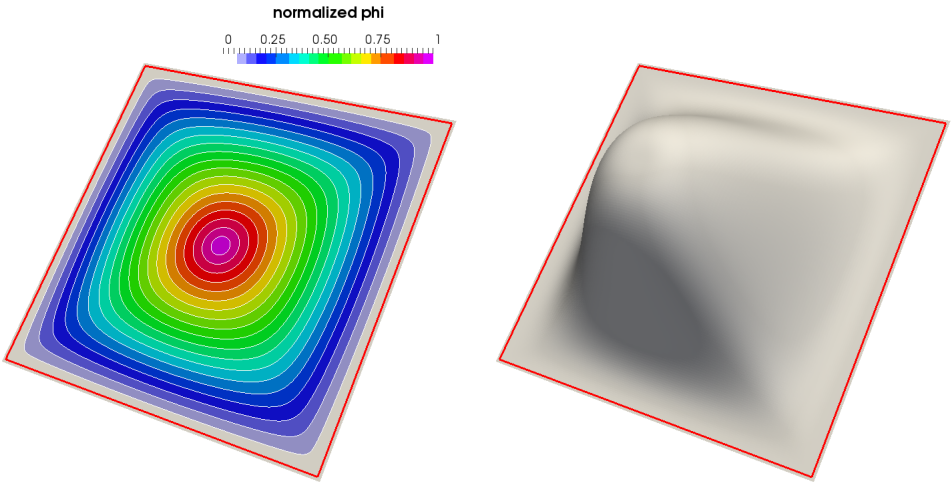
This approach permits to handle a broad class of constrains and can be applied straightforwardly to other parameterization methods, such as Radial Basis Functions (RBF).

2.2.3 Geometrical constraints

As stated above, industrial components must satisfied a set of geometrical constraints during the optimization process. The feasibility of the deformation can be controlled by bounding the design parameters, especially when the constraint is intuitive, or by using some control strategy. Most of the geometrical constraints, e.g. prescribed distances among components or absolute positions, can be translated into volume constraints, where the deformable part is not allowed to occupy a certain volume in the space. To verify the compliance with such requirement, it is possible to use the level set method



(a) Exact geodesic distance function.



(b) Approximated geodesic distance function.

Fig. 2.5 Examples of distance functions from a non-smooth contour (red) and corresponding surface deformations with C^2 continuity constraint.

(distance from the constraint) and efficient ray-tracing and line search algorithms. In principle, this information can be used to rescale the deformation and enforce the feasibility of the new configuration. However, this approach tends to produce plateaus in the optimization, that may slow down the process or be difficult to overcome, depending on the optimization algorithm. For this reason we prefer to simply treat the constraints violation as an inequality constraint for the optimizer, i.e. assigning $g_i(\boldsymbol{\mu}) < 0$ for a feasible deformation controlled by the $\boldsymbol{\mu}$ parameter, and $g_i(\boldsymbol{\mu}) \geq 0$ for an unfeasible one, where $g_i(\boldsymbol{\mu})$ represents the maximum violation distance between the constraint surface and the deformed component.

2.3 Numerical tools

The above features are implemented in a software package for computer-aided surface and mesh morphing, called mimic[1], which is a $C{+}{+}$ library to manipulate three-dimensional objects by means of FFD and Radial Basis parameterization techniques.

For some applications, we also rely on the PyGem[2] python library, that provides an implementation of the standard FFD technique.

[1] http://www.optimad.it/products/mimic/
[2] http://mathlab.sissa.it/pygem

Chapter 3

Full-Order Modelling

Part of the work described in this chapter has been previously submitted for publication in Salmoiraghi et al. (2018) and Bergmann et al. (2018).

This chapter introduces the reference full-order model (Section 3.1) for the problems of interest and provides some insight about the numerical tools and methods (Section 3.2) employed in its solution.

3.1 Fluid modelling

In $\Omega(\boldsymbol{\mu})$ the flow motion is governed by the NSE. Generally speaking, the NSE are a system of non-linear, time-dependent, partial differential equations, that in the conservation form for incompressible newtonian fluids can be written as:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{1}{\rho}\nabla p + \nabla \cdot (\nu \nabla \mathbf{u}), \tag{3.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3.2}$$

where $\mathbf{u} = \mathbf{u}(\boldsymbol{\mu})$ and $p = p(\boldsymbol{\mu})$ denote the velocity and pressure fields respectively, ν is the kinematic viscosity and ρ is the density.

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). In terms of Reynolds number $Re = \frac{UL}{\nu}$, i.e. the

dimensionless number that represents the ratio between inertial and viscous forces for given flow conditions, the applications of interest for the present work are characterised by values of $O(10^6 - 10^8)$, thus turbulence needs to be addressed. 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, such as Detached Eddy Simulation (DES). In DNS, all the scales of motions are resolved for one realization 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 Re: therefore, this approach can be applied to flows characterized 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 (Pope (2011); Sagaut (2006)) and hybrid models (Fröhlich and von Terzi (2008)), have started to be employed in engineering applications. Nevertheless, the solution of RANS equations is still the most common approach in industry, especially in early stages of design or during aerodynamic optimization, when several simulations are required. The general idea behind the RANS approach is to decompose both $\mathbf{u}$ and p into ensemble-averaged and fluctuating components (Reynolds decomposition)

$$\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}', \quad p = \bar{p} + p', \tag{3.3}$$

in order to obtaining approximate solutions to the NSE. By substituting (3.3) into Equations (3.1-3.2) and by applying the average operator, it is possible to derive the RANS formulation for incompressible flows:

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) + \nabla \cdot \left(\overline{\mathbf{u}'\mathbf{u}'}\right) = -\frac{1}{\rho}\nabla \bar{p} + \nabla \cdot (\nu \nabla \bar{\mathbf{u}}), \tag{3.4}$$

$$\nabla \cdot \bar{\mathbf{u}} = 0. \tag{3.5}$$

The averaging operator introduces an additional unknown to the problem, i.e. the Reynolds stress tensor, whose components are defined as $\tau_{ij}^t = -\rho\overline{u_j' u_i'}$. This new term accounts for the fluctuations contribution (Pope (2011)), adding to the momentum equation the macroscopic momentum transport due to turbulence. In order to provide mathematical closure to the system of equations, a turbulence model is required to determine such contribution. Although many different models are available in literature, in this work we will address only linear-eddy viscosity models based on one or two

differential equations, since they represent the most common methods for the problems of interest. Some insight regarding these models is provided in Section 3.1.1

In the following, we will refer to the RANS equations as the high-fidelity/full-order model when addressing the general Problem (1.2) described in Section 1.1. In order to simplify the notation, we will omit the bar over the averaged fields.

3.1.1 Turbulence modelling

The closure of RANS equations is a well known problem in fluid dynamics and various models have been proposed in order to describe the Reynolds stress tensor. These models belong to two main categories:

- **Eddy viscosity models**, where the Reynolds stresses are evaluated though a constitutive relationship based on the mean velocity field. Depending on the relationship type, a further distinction between linear and nonlinear models is generally introduced. Linear models are the most commonly used. They rely on the so-called Boussinesq hypothesis:

$$\tau_{ij}^t = \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij} \tag{3.6}$$

  where k is the turbulent kinetic energy, $k = \frac{1}{2}\overline{u_i' u_i'}$. Equation (3.6) introduces a direct proportionality between the deviatoric Reynolds stress, $\tau_{ij}^t + \frac{2}{3}\rho k \delta_{ij}$, and the mean rate of strain, $\bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$, through a positive factor equal to $2\nu_t$, with ν_t referred to as *turbulent* or *eddy* viscosity. The effect of turbulence translates into an increased *effective* viscosity, given by $\mu_{\text{eff}} = \mu + \mu_t$. Within the class of linear models, early developed one-equation models are incomplete, i.e. their constituent equations include flow-dependent specification, whereas more recent ones and two-equations models are complete.

  Although the eddy-viscosity hypothesis is not entirely correct, it usually leads to reasonably good results.

- **Reynolds stress equation models (RSM)**, where the eddy-viscosity hypothesis is avoided and Reynolds stresses are computed individually, using exact transport equations, in order to account for directional effects and complex interactions that cannot be modelled otherwise.

The suitability of a turbulence model for a given problem is determined by a lot of factors, like the level of description required, completeness, accuracy and range of applicability, but also usability criteria, such as cost and ease to use: the difficulty of performing turbulent calculations, in fact, depends on both the model and the flow features. Generally speaking, the results show a strong dependency on the chosen RANS model. For these reasons, given a certain class of turbulent problems, it does not exist an optimal turbulence model, but rather a set of suitable models that can be applied. A comprehensive review of turbulence flows and modelling can be found for instance in Pope (2011) and Wilcox (2006).

In this work, the following complete models are employed.

Spalart-Allmaras

The Spalart-Allmaras model, introduced by Spalart and Allmaras (1992) for aerodynamic applications, is a one-equation model which introduces an additional transport equation for the turbulent viscosity ν_t, in the form:

$$\frac{\partial \nu_t}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \nu_t = \nabla \cdot \left(\frac{\nu_t}{\sigma_\nu} \nabla \nu_t \right) + S_\nu$$

where σ_ν is a model constant and S_ν is a source term depending on the laminar and turbulent viscosities, ν and ν_t respectively, the rate of rotation, the wall distance and the gradient of ν_t, as well as on several semi-empirical constants.

Realisable $k - \epsilon$

The realisable $k - \epsilon$ model is a modification of the standard $k - \epsilon$ model introduced by Jones and Launder (1972), one of the most commonly used turbulence models. The $k - \epsilon$ model belongs to the class of two-equations models and introduces two extra transport equations for the turbulent kinetic energy and for the turbulent energy dissipation rate, i.e. k and ϵ. The transport equation for k is written as:

$$\frac{\partial k}{\partial t} + \bar{\mathbf{u}} \cdot \nabla k = \nabla \cdot \left(\frac{\nu_t}{\sigma_k} \nabla k \right) + \mathcal{P} - \epsilon$$

where $\nu_t = \rho C_\mu k^2/\epsilon$ is a function of the kinetic energy and the wall distance and $\mathcal{P}$ is the k production term, while the transport of ϵ is expressed in the empirical form:

$$\frac{\partial \epsilon}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \epsilon = \nabla \cdot \left(\frac{\nu_t}{\sigma_\epsilon} \nabla \epsilon \right) + C_{\epsilon 1} \frac{\mathcal{P} \epsilon}{k} - C_{\epsilon 2} \frac{\epsilon^2}{k} \tag{3.7}$$

with the standard values of the constants determined by Launder and Sharma (1974). The empirical nature of the ϵ equation is one of the main source of error of the standard model, which is usually quite inaccurate for complex flows, e.g. round jets or boundary layers under strong adverse pressure gradients. The *realisable* $k - \epsilon$ model (Shih et al. (1995)) tries to mitigate this effect by introducing a variable C_μ (constant in the standard model) and a new definition of the ϵ transport equation. More specifically, the new equation is derived from an exact dynamic equation for the mean-square vorticity fluctuation at high Reynolds numbers:

$$\frac{\partial \epsilon}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \epsilon = \nabla \cdot \left(\frac{\nu_t}{\sigma_\epsilon} \nabla \epsilon \right) + C_1 \epsilon + C_{\epsilon 1} \frac{\mathcal{P}_b \epsilon}{k} - C_2 \frac{\epsilon^2}{k + \sqrt{\nu \epsilon}} \tag{3.8}$$

The noteworthy feature is that the production term of Equation (3.8) does not involve the production of k due to the mean velocity gradients, as in Equation (3.7), but only the buoyancy contribution $\mathcal{P}_b$, providing a better representation of the spectral energy transfer. Moreover, the derived model satisfies certain *realisability* constraints on the Reynolds stresses, assuring consistency with the physics of turbulent flows, differently from the standard model. From the Boussinesq relationship (Equation (3.6)) and the ν_t definition, it follows that the normal stresses

$$\overline{u_i^2} = 2k \left(\frac{1}{3} - C_\mu \frac{k}{\epsilon} \frac{\partial \bar{u}_i}{\partial x_i} \right)$$

becomes negative when the following condition holds:

$$\frac{k}{\epsilon} \frac{\partial \bar{u}_i}{\partial x_i} > \frac{1}{3 C_\mu}.$$

The same can be shown also for shear stresses. In other words, the solution is *not-realisable* when the strain is sufficiently large, unless a variable C_μ is introduced.

Further information about the numerical implementation of the models can be found in Section 3.2.5.

3.2 Numerical tools

Numerical simulations are performed through the OpenFOAM® [1] C++ library developed by OpenCFD Ltd and distributed by the OpenFOAM Foundation. It is an open-source software for computational fluid dynamics which in the past decade has become a reference point for both academic research and industrial users. As underlined in the works of Weller et al. (1998) and Jasak et al. (2007), the main feature of OpenFOAM® is its object-oriented and expressive syntax, that allows the users to easily customize and implement complex physical models.

For the solution of the steady-state RANS equations, we use the `simpleFoam` solver, which implements the SIMPLE pressure-velocity coupling proposed by Patankar (1980).

In the following, a brief description of the main ingredients of the numerical discretization is provided.

3.2.1 Spatial discretization

The computational domain Ω is divided into a finite number of polyhedral control volumes, i.e. the mesh cells, on which the governing equations are solved. Cells do not overlap and their union covers the whole computational domain. Each control volume is delimited by a set of flat faces and may have generic shape and a variable number of neighbours, with no restriction on face alignment, resulting in an *arbitrarily unstructured* mesh that allows handling complex geometries.

An example of control volume is illustrated in Figure 3.1: with reference to the face f, it is possible to identify its *owner* and *neighbour* cell, whose centroids are denoted by P and N respectively. The face area vector $\mathbf{s}_\mathrm{f}$, instead, represents the outward-pointing normal vector whose magnitude is given by the face area. As hinted before, such vector $\mathbf{s}_\mathrm{f}$ and the vector pointing from the cell centre P to N, i.e. $\mathbf{d}$, may be non-aligned: therefore, mesh non-orthogonality must be addressed in the equations discretization. All the unknown variables are computed at P and stored at one single location, according to the co-located variable arrangement and following the typical implementation of the cell-centred finite volume method (see for instance Versteeg and Malalasekera (2007) for a comprehensive description of the finite volume method in fluid dynamics problems).

[1] https://openfoam.org/

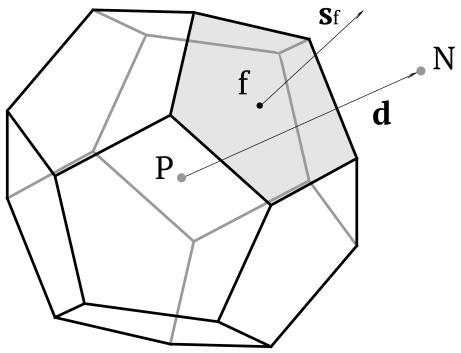


Fig. 3.1 Typical control volume and discretization parameters.

The mesh is described by its points, faces, cells and boundary patches. Each cell is defined by the list of faces bounding its volume, each face as an ordered list of point labels and each point by its spatial coordinates. Boundary faces, i.e. those sitting on the boundary $\partial\Omega$ of the domain, are grouped into patches, in order to easily subdivide the boundary into regions subject to different types of boundary conditions.

3.2.2 Equation discretization

Starting from Equations (3.5) and (3.4), it is possible to derive the finite volume discretization of the full-order model by linearisation and integration of each spatial term over the control volumes, in order to transform the PDEs system into a set of algebraic equations.

The continuity equation can be easily discretized by explicitly evaluating the divergence term over the cells as

$$\int_V \nabla \cdot \mathbf{u}\, dV = \int_S d\mathbf{s} \cdot \mathbf{u} = \sum_f \mathbf{s}_f \cdot \mathbf{u}_f \tag{3.9}$$

where the volume integral is converted into a surface integral thanks to Gauss' theorem.

The discretization of the momentum equation, instead, requires particular attention due to the presence of the non-linear convection term. The linearisation of such term is achieved in the following way:

$$\int_V \nabla \cdot (\mathbf{uu})\ dV = \int_S d\mathbf{s} \cdot (\mathbf{uu}) = \sum_f \mathbf{s}_f \cdot \mathbf{u}_f \mathbf{u}_f \simeq a_P \mathbf{u}_P + \sum_N a_N \mathbf{u}_N \tag{3.10}$$

where a_P and a_N depend on the velocity field and are computed using the velocity fluxes from a previous iteration. This lagging has no relevant effect in steady-state calculation, whereas needs to be properly addressed in transient problems. The Laplacian term and the divergence of the Reynolds stress tensor are also computed using lagged turbulent quantities, i.e. the eddy viscosity ν_t. In particular, the Laplacian is discretized as follows:

$$\int_V \nabla \cdot (\nu_t \nabla \mathbf{u})\, dV = \int_S d\mathbf{s} \cdot (\nu_t \nabla \mathbf{u}) = \sum_f (\nu_t)_f\, \mathbf{s}_f \cdot (\nabla \mathbf{u})_f$$

where the product $\mathbf{s}_f \cdot (\nabla u)_f$ is given by the sum of the orthogonal contribution, i.e. $|\mathbf{s}_f|(\mathbf{u}_N - \mathbf{u}_P)/|\mathbf{d}|$, and a correction for the non-orthogonality (see for instance Jasak (1996)). The pressure gradient is evaluated explicitly by Gaussian integration.

3.2.3 SIMPLE algorithm

As shown in Jasak (1996), the method relies on the derivation of an explicit equation for the pressure. We start from a semi-discretized form of the momentum equation:

$$a_P \mathbf{u}_P + \sum_N a_N \mathbf{u}_N = \mathbf{b}_P - (\nabla p)_P\,. \tag{3.11}$$

For the sake of simplification, the neighbours contributions and the source terms apart from the pressure gradient, whose discretization is not made explicit at this stage, can be grouped into $\mathbf{H}_P(\mathbf{u})$. Equation (3.11) is then rewritten as:

$$a_P \mathbf{u}_P = \mathbf{H}_P(\mathbf{u}) - (\nabla p)_P\,. \tag{3.12}$$

Thus, for the cell velocities we have:

$$\mathbf{u}_P = \frac{\mathbf{H}_P(\mathbf{u})}{a_p} - \frac{(\nabla p)_P}{a_p}. \tag{3.13}$$

In addition to the momentum equation, the velocity field is also subject to the continuity constraint, that according to the discretization procedure of Section 3.2.2, can be expressed in terms of the face fluxes as:

$$\sum_f \mathbf{s}_f \cdot \mathbf{u}_f = 0, \tag{3.14}$$

where $\mathbf{s}_f$ is the face area vector and $\mathbf{u}_f$ represents the velocity interpolated on the face, which is obtain from Equation (3.13):

$$\mathbf{u}_f = \left(\frac{\mathbf{H}_P(\mathbf{u})}{a_P}\right)_f - \left(\frac{\nabla p}{a_P}\right)_f . \tag{3.15}$$

By substituting Equation (3.15) into Equation (3.14), we obtain the final form of the discretised pressure equation:

$$\sum_f \mathbf{s}_f \cdot \left(\frac{\nabla p}{a_P}\right)_f = \sum_f \mathbf{s}_f \cdot \left(\frac{\mathbf{H}_P(\mathbf{u})}{a_p}\right)_f . \tag{3.16}$$

Equations (3.11) and (3.16) are solved iteratively through under-relaxation, until convergence is attained. Starting from an initial guess for the unknown variables and the setup of suitable boundary conditions, the solution procedure consists of the steps summarised in Algorithm 1. First, the momentum equation is under-relaxed implicitly and solved in order to obtain the predicted velocity field $\mathbf{u}^\star$ and the corresponding face fluxes, using the pressure field from the previous iteration. $\mathbf{u}^\star$ is then used in Equation (3.16) to evaluate the new pressure distribution. Once the new p is obtained, the face fluxes are corrected and the new pressure field for the momentum predictor step is computed by under-relaxing p. Finally, the velocity is explicitly corrected and the values on the boundary are updated in order to satisfy the prescribed boundary conditions. For turbulent flows, the conservative fluxes are then used to solve the additional turbulence equations, in order to calculate the effective viscosity to be imposed at the next solver iteration.

Algorithm 1 SIMPLE algorithm

1: set the boundary conditions
2: initialize p and $\mathbf{u}$
3: **while** *convergence* = false **do**
4: under-relax $\frac{a_P}{\alpha_\mathbf{u}}\mathbf{u}_P^{\text{new}} + \sum_N a_N \mathbf{u}_N^{\text{new}} = \mathbf{b}_P - (\nabla p)_P + \frac{1-\alpha_\mathbf{u}}{\alpha_\mathbf{u}} a_P \mathbf{u}^{\text{old}}$
5: $\mathbf{u}^\star \leftarrow a_P \mathbf{u}_P = \mathbf{H}_P(\mathbf{u}) - (\nabla p)_P$
6: $p' \leftarrow \sum_f \mathbf{s}_f \cdot \left[\left(\frac{\nabla p}{a_P}\right)_f\right] = \sum_f \mathbf{s}_f \cdot \left(\frac{\mathbf{H}_P(\mathbf{u}^\star)}{a_p}\right)_f$
7: compute the conservative face fluxes $\mathbf{s}_f \cdot \mathbf{u}_f = -\mathbf{s}_f \cdot \left(\frac{\nabla p'}{a_P}\right)_f + \mathbf{s}_f \cdot \left(\frac{\mathbf{H}_P(\mathbf{u}^\star)}{a_p}\right)_f$
8: under-relax $p^{\text{new}} = p^{\text{old}} + \alpha_p(p' - p^{\text{old}})$, with $0 < \alpha_p \leq 1$
9: correct the cell velocities $\mathbf{u}_p = \frac{\mathbf{H}_P(\mathbf{u}^\star)}{a_p} - \frac{(\nabla p^{\text{new}})_P}{a_p}$
10: update the boundary conditions
11: check for *convergence*
12: **end while**

Further details can be found in Ferziger and Peric (2002) and Versteeg and Malalasekera (2007).

The SIMPLE algorithm loop is implemented in the C++ code as follows:

```
while (simple.loop())
{
    // --- Pressure-velocity SIMPLE corrector
    {
        #include "UEqn.H"
        #include "pEqn.H"
    }

    turbulence->correct();
}
```

addressing first the momentum predictor step (`UEqn.H`), and then the pressure correction step (`pEqn.H`): when required the pressure equation may be solved more than once in order to introduce the correction for the non-orthogonality.

3.2.4 Numerical schemes

The equations are discretised with Gaussian integration, using different schemes for each term. For the gradient terms, we use a second-order accurate Gauss linear scheme for p and $\mathbf{u}$ and the cell-limited version for turbulent quantities, which limits the gradient in order to guarantee boundedness. Such schemes are defined in the OpenFOAM® dictionaries by `Gauss linear` and `cellLimited Gauss linear` ψ, with $\psi = 1$. In both schemes, the face value of the variable y is obtained from the cell centre values as:

$$y_f = w y_P + (1 - w)\, y_N,$$

where the interpolation factor is defined in terms of the cell centre-face centre distance vector, $\mathbf{d}_f$, computed with respect to N, and the distance vector between the cell centres P and N, i.e. $\mathbf{d}$:

$$w = \frac{|\mathbf{d}_f|}{|\mathbf{d}|}. \tag{3.17}$$

When dealing with turbulence, the cell gradient is limited to ensure that the extrapolated face values are bounded by the minimum and maximum values in the neighbouring cells,

i.e.:

$$\min_N(y_N) \leq y_P + \mathbf{d} \cdot (\nabla y)_P \leq \max_N(y_N).$$

In order to prevent local over or under-shoots in the reconstructed field, a gradient limiter is applied. When using $\psi = 1$, if for example the extrapolated face value exceeds y_N, the gradient is scaled in order to reach exactly y_N.

The surface normal gradient scheme is a central-difference scheme with limited non-orthogonal correction. The correction is controlled by a coefficient ψ ranging between 0 and 1: when $\psi = 0$ the correction is off, when set to 1 the scheme is fully corrected, whereas when set to intermediate values the limiter is calculated such that the non-orthogonal correction does not exceed ψ times the orthogonal contribution. With reference to the nomenclature introduced in Figure 3.1, the uncorrected gradient scheme of a property y at a face can be evaluated as:

$$\nabla_f^{\perp} y = \alpha \frac{y_p - y_N}{|\mathbf{d}|} \tag{3.18}$$

with

$$\alpha = \frac{1}{cos(\theta)},$$

being θ the angle between the vector $\mathbf{d}$, that joins the two cell centres, and the surface normal $\mathbf{n}$. The full corrected scheme, instead, is evaluated by adding an explicit correction term to Equation 3.18:

$$\nabla_f^{\perp} y = \alpha \frac{y_P - y_N}{|\mathbf{d}|} + \left(\mathbf{n} - \alpha \frac{\mathbf{d}}{|\mathbf{d}|} \right) \cdot (\nabla y)_f ,$$

where $(\nabla y)_f$ is cell-based gradient interpolated on the face. In order to improve stability even for highly non-orthogonal meshes, we set $\psi = 0.333$, by specifying `limited 0.333`.

The Laplacian terms, $\nabla \cdot (\gamma \nabla y)$ are approximated by a blending of a bounded, first order, non-conservative scheme and an unbounded, second order, conservative scheme, More specifically, the scheme applies the Gauss theorem with linear interpolation, required to transform the diffusion coefficient cell values into face values, through central differencing:

$$\gamma_f = w\gamma_P + (1 - w)\, \gamma_N,$$

where w is defined as in Equation 3.17.

The surface normal gradient is then computed by employing the same limited scheme discussed above, with a 0.333 non-orthogonal correction. The Laplacian scheme is specified by `Gauss linear limited 0.333`: once again, such scheme is less accurate

with respect to the fully corrected counterpart, but it performs better on non-orthogonal meshes.

The convective terms in the turbulence equations are discretised using a Total Variation Diminishing (TVD) bounded scheme, i.e. limited linear differencing. Generally speaking, TVD schemes introduce a blend between a low-order and a higher-order scheme, according to the calculation of a limiter (see Jasak (1996)):

$$y_f = y_{\mathrm{UD}} + \psi(r)\left(y_{\mathrm{HO}} - y_{\mathrm{UD}}\right),$$

where $\psi(r)$ represent the TVD flux limiter, and y_{UD} and y_{HO} are the face value of y for the first-order upwind differencing scheme and the higher-order scheme, that is central differencing in our case. $\psi(r)$ is evaluated through a compact stencil, as shown in Jasak et al. (1999), and defined as a function of the variable r:

$$r = 2\frac{\mathbf{d}\cdot(\nabla y)_P}{y_N - y_P} - 1,$$

where the gradient at cell P, $(\nabla y)_P$ is computed through second order, Gaussian integration. The selected scheme is `bounded Gauss limitedLinear 1`, where the value 1 set the limiter $\psi(r) = 2r$, that is TVD conforming. This guarantees the best performance of the scheme in terms of convergence.

As for the momentum equation, for the convective term we employed a linear upwind differencing scheme, specified through `bounded Gauss linearUpwind grad(U)`. The scheme is second order accurate and employs upwind interpolation with an explicit correction based on the local cell gradient, following Warming and Beam (1976). With reference to Figure 3.2, where the nodes U, C and D are chosen according to the direction of the flux across f, the face value is discretized as:

$$y_f = y_{\mathrm{C}} + \mathbf{d}_f \cdot (\nabla y)_{\mathrm{C}},$$

where $\mathbf{d}_f$ represents the vector directed from the C cell centre to the face centre and the variable gradient on C is computed through the corresponding gradient scheme, i.e. `Gauss linear`.

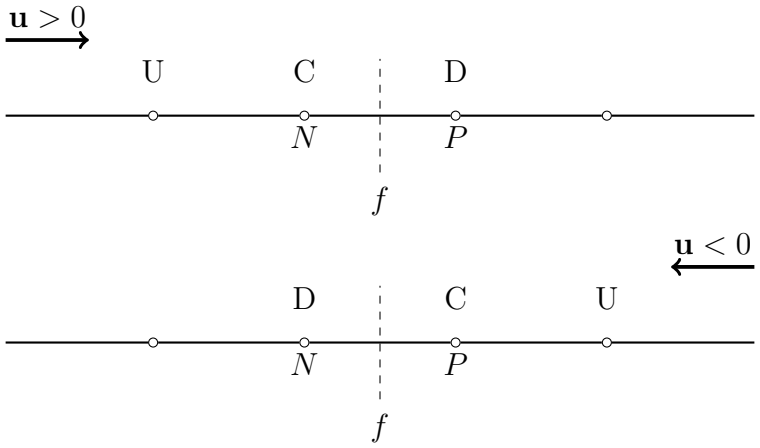


Fig. 3.2 Cells stencils used in the construction of y_f for convection schemes: the subscripts U, C and D denote the upstream, central and downstream nodes, coinciding with the cell centres.

3.2.5 Turbulence

Spalart-Allmaras

The `SpalartAllmaras` model implemented in OpenFOAM® references Spalart and Allmaras (1994) with some minor changes. The transport equation is written for the viscosity-like variable $\tilde{\nu}$:

$$\frac{\partial \tilde{\nu}}{\partial t} + \bar{u}_i \frac{\partial \tilde{\nu}}{\partial x_i} = C_{b1}\tilde{S}\tilde{\nu} - C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left\{\frac{\partial}{\partial x_j}\left[(\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\right] + C_{b2}\frac{\partial \tilde{\nu}}{\partial x_j}\frac{\partial \tilde{\nu}}{\partial x_j}\right\}$$

from which the turbulent viscosity is obtained as

$$\nu_t = \tilde{\nu} f_{v1}$$

with

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3},$$
$$\chi = \frac{\tilde{\nu}}{\nu}.$$

Contrary to the standard SA model, this implementation expresses the quantity $\tilde{S}$ in the production term as follows:

$$\tilde{S} = \max\left(Q + \frac{\tilde{\nu}}{k^2 d^2} f_{v2}, C_s Q\right),$$

where Q is the vorticity magnitude, $Q = \sqrt{2Q_{ij}Q_{ij}}$ with $Q_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i}\right)$, and the term f_{v2} is:

$$f_{v2} = 1 - \frac{\chi}{(1 + \chi f_{v1})}.$$

Additional definitions of the model are:

$$f_w = g\left[\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\right]^{\frac{1}{6}}, \qquad g = r + C_{w2}\left(r^6 - r\right),$$
$$r = \min\left(\frac{\tilde{\nu}}{\tilde{S}k^2d^2}, 10.0\right),$$

with coefficients:

$$\begin{aligned} C_{b1} &= 0.1355, & C_{b2} &= 0.622, \\ C_{w2} &= 0.2, & C_{w3} &= 2.0, \\ C_{v1} &= 7.1, & C_s &= 0.3, \\ \sigma &= 0.66666, & k &= 0.41, \end{aligned}$$
$$C_{w1} = \frac{C_{b1}}{k^2} + \frac{1 + C_{b2}}{\sigma}.$$

Realisable $k - \epsilon$

OpenFOAM® includes the `realizableKE` model developed by Shih et al. (1995). The kinetic energy and dissipation equations are written in the form:

$$\frac{\partial k}{\partial t} + \frac{\partial (k u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G - \epsilon S_k \ ,$$
$$\frac{\partial \epsilon}{\partial t} + \frac{\partial (\epsilon u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] + C_1 S\epsilon - C_2\frac{\epsilon^2}{k + \sqrt{\nu\epsilon}} + S_\epsilon \ ,$$

where:

$$C_1 = \max\left(\frac{\eta}{\eta + 5}, 0.43\right), \qquad \eta = S\frac{k}{\epsilon},$$
$$G = \nu_t S^2, \qquad S = \sqrt{2\bar{s}_{ij}\bar{s}_{ij}} \ , \qquad \bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right).$$

As in standard $k - \epsilon$ models, the turbulent viscosity is computed as

$$\nu_t = C_\mu \frac{k^2}{\epsilon}$$

with C_μ varying through the relation

$$C_\mu = \frac{1}{A_0 + A_s \frac{kU_s}{\epsilon}}.$$

Additional definitions of the model are:

$$C_1 = \max\left(\frac{\eta}{\eta+5}, 0.43\right),$$
$$U_s = \sqrt{\bar{s}_{ij}\bar{s}_{ij} + \tilde{R}_{ij}\tilde{R}_{ij}}\ , \qquad \tilde{R}_{ij} = R_{ij} - 2\epsilon_{ijk}\omega_k\ , \qquad R_{ij} = \bar{R}_{ij} - \epsilon_{ijk}\omega_k\ ,$$

where $\bar{R}_{ij}$ represents the mean rate-of-rotation expressed in a reference frame rotating with ω_k angular velocity. The model is closed by the following constants and relationships:

$$A_0 = 4.0, \qquad A_s = \sqrt{6}\cos\phi_s\ ,$$
$$\phi_s = \frac{1}{3}\cos^{-1}(\sqrt{6}W)\ ,$$
$$W = \frac{\bar{s}_{ij}\bar{s}_{jk}\bar{s}_{ki}}{\sqrt{\bar{s}_{ij}\bar{s}_{ij}}^3}\ ,$$
$$C_2 = 1.9, \qquad \sigma_k = 1.0, \qquad \sigma_\epsilon = 1.2.$$

3.2.6 Boundary conditions

To better understand the implementation of the boundary conditions for the hybrid full-order/reduced-order approach described in Sections 5.3 and 5.3.2, it is worth noting how OpenFOAM® handles the prescription of Dirichlet and Neumann conditions on the boundary.

Dirichlet boundary conditions are equivalent to fix the value on the boundary for the given variable y. For convective terms, the prescribed value on the boundary face, namely y_b, is directly substituted into the discretization; alternatively, the face value is used together with the cell centre value in the computation of the face gradient $(\nabla y)_b$, through the relation

$$\mathbf{s}_f \cdot (\nabla y)_b = \mathbf{s}_f \frac{y_b - y_P}{|\mathbf{d}_n|}\ ,$$

where $\mathbf{d}_n$ is the outward pointing normal vector between the cell centre P and the boundary face, as shown in Figure 3.3 for a generic control volume. Such vector can be

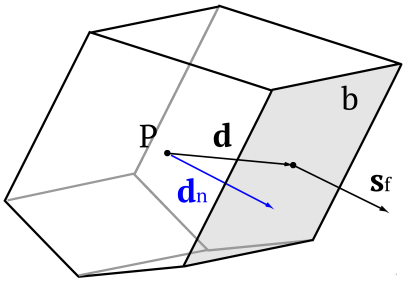


Fig. 3.3 Discretization parameters for a control volume of centre P with a boundary face b. The $\mathbf{d}_n$ vector is orthogonal to b.

easily computed as

$$\mathbf{d}_n = \frac{\mathbf{s}_f}{|\mathbf{s}_f|}\frac{\mathbf{d}\cdot\mathbf{s}_f}{|\mathbf{s}_f|}$$

and its definition is required in order to take into account the non-orthogonality of the mesh.

Neumann boundary conditions, instead, require to specify the component of the gradient normal to the boundary face, i.e.

$$\left(\frac{\mathbf{s}_f}{|\mathbf{s}_f|}\cdot\nabla y\right)_b = g_b.$$

For diffusion terms, g_b can be directly prescribed through the relation $\mathbf{s}_f\cdot(\nabla y)_b = |\mathbf{s}_f|g_b$; whereas for convection terms the face value y_b is derived from the prescribed gradient as $y_b = y_P + |\mathbf{d}_n|g_b$.

From these two main types of numerical boundary conditions, it is possible to straightforwardly derive different types of physical boundary conditions, such as inlet, outlet and no-slip impermeable wall.

For incompressible flows, due to numerical reasons, the inflow is usually modelled by imposing a fixed velocity and a gradient condition for the pressure, whereas the opposite is recommended for the outflow. At turbulent inlets, turbulent quantities are defined through the specification of the turbulent length scale l and intensity I:

$$k = \frac{3}{2}(U_{\text{ref}}I), \qquad \epsilon = \frac{C_\mu^{\frac{3}{4}}k^{\frac{3}{2}}}{l}, \qquad \tilde{\nu} = C_\mu^{\frac{1}{4}}\sqrt{\frac{3}{2}}(U_{\text{ref}}lI).$$

Both slip and non-slip boundary conditions for solid walls are obtained imposing a null pressure normal gradient and the desired wall velocity. In terms of turbulence,

the realisable $k-\epsilon$ model is a high-Reynolds model and requires near-wall treatment, while the Spalart-Allmaras can be used as a low-Reynolds model to solve the boundary layer up to the viscous sublayer, introducing a first cell such that $y^+ \approx 1$ and imposing $\tilde{\nu} = \nu_t = 0$ at the wall. However, a uniform boundary layer mesh respecting this condition is usually difficult to attain for complex industrial geometries. If this this requirement is not satisfied, the resulting wall shear-stress will not be accurate.

In order to restore a physically correct solution, wall functions are usually employed for both methods to set an adequate value of the eddy viscosity at the wall, valid when the first cell centre is in the logarithmic area. For the Spalart-Allmaras model, we use the `nutUSpaldinfWallFuntion` available in OpenFOAM®, that implements the Spalding's blended law (Spalding (1961)):

$$y^+ = u^+ + \frac{1}{E}\left[\exp\left(ku^+\right) - 1 - ku^+ - \frac{1}{2}\left(ku^+\right)^2 - \frac{1}{6}\left(ku^+\right)^3\right]. \tag{3.19}$$

By definition, the curve fits $u^+ = y^+$ in the viscous layer and $u^+ = Ey^+/k$ in the logarithmic area, but its slope is always continuous even in the buffer layer ($5 < y^+ < 30$). As a consequence, its usage is recommended when the position of the first cell centre is not known a priori. As shown in Liu (2016), the corrected turbulent viscosity value is recovered by

$$\nu_t = \frac{(u_\tau)^2}{|\frac{\partial \mathbf{u}_{\text{wall}}}{\partial n}|} - \nu$$

where the friction velocity u_τ is computed through an iterative Newton-Raphton method. After substituting $y^+ = yu_\tau/\nu$ and $u^+ = u/u_\tau$ in Equation (3.19), the Spalding's law becomes:

$$-\frac{yu_\tau}{\nu} + \frac{u}{u_\tau} + +\frac{1}{E}\left[\exp\left(k\frac{u}{u_\tau}\right) - 1 - k\frac{u}{u_\tau} - \frac{1}{2}\left(\frac{u}{u_\tau}\right)^2 - \frac{1}{6}\left(\frac{u}{u_\tau}\right)^3\right] = 0,$$

which is solved in each near-wall cell.

The realisable $k-\epsilon$ model, instead, requires the introduction of wall functions also for the turbulent kinetic energy and dissipation rate. For k, we impose the `kqRWallFunction`, which provides a pure zero-gradient boundary condition, whereas for ϵ we use the `epsilonWallFunction`, that calculates the cell centre value as a weighted sum of the face values:

$$\epsilon = \frac{1}{W}\sum_f W_f \left(\frac{C_\mu^{3/4} k^{3/2}}{k y_f}\right).$$

For turbulent viscosity, a condition based on the turbulent kinetic energy, namely `nutkWallFunction`, is recommended. This condition operates in the following way:

$$\begin{cases} \nu_t = \nu \left(\dfrac{k y^+}{\ln(E y^+)} - 1 \right) & \text{if } y^+ > y^+_{\text{lam}} \ , \\ \nu_t = 0 & \text{if } y^+ \leq y^+_{\text{lam}} \ , \end{cases}$$

with y^+_{lam} the limit between the linear laminar part and the logarithmic layer.

Chapter 4

Benchmark Case

In this chapter we introduce the simplified test case that will be employed later on to assess the numerical properties of the proposed approaches.

4.1 2DCAR: flow past a 2D car profile

This test case involves the study of the incompressible flow around a 2-dimensional car profile, given by the mid-section of the DrivAer[1] car model with fastback body style (see Section 8.1 for further details), referred to as the *2DCAR benchmark* in the following.

4.1.1 Problem specification

The geometry of the problem is shown in Figure 4.1. The solution domain is a $L \times H$ box with $L = 50$ m and $H = 12$ m. The car is moving at a reference speed $U_\infty = 16$ m/s and the reference system is assumed vehicle-fixed: therefore, we have an inflow velocity condition of $\mathbf{u} = (U_\infty, 0)$ and the ground moves in the x-direction with the same reference speed. In terms of physical properties, such as air density and viscosity, we refer to the standard atmosphere model at 15 °C. With this setup, the Re number of the simulations, defined in terms of the free-stream velocity U_∞ and the car profile length $l_{\text{ref}} = 4.61$ m, is set equal to $4.87 \cdot 10^6$, that is a realistic value for automotive applications with turbulent flows. For the definition of turbulent quantities, we assume that the inlet turbulence is

[1] http://www.drivaer.com

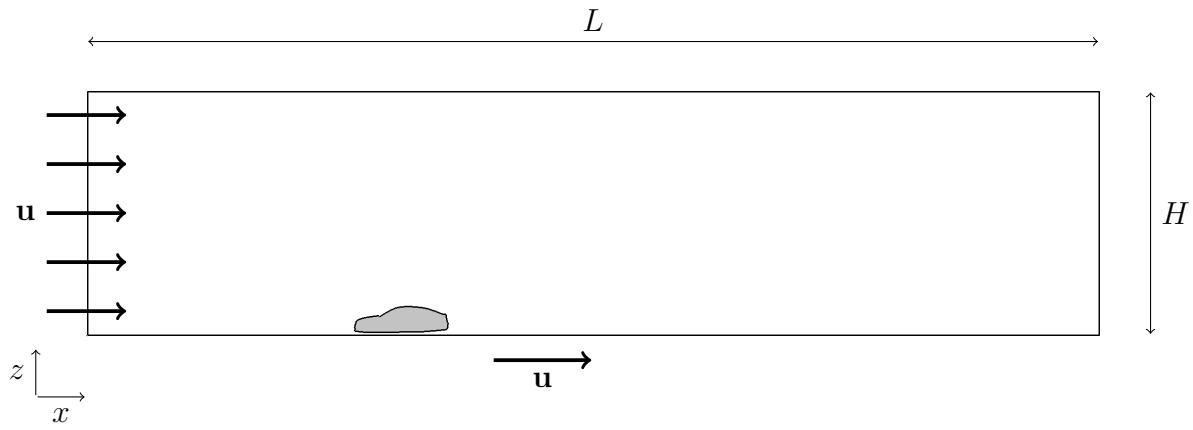


Fig. 4.1 Geometry of the 2DCAR benchmark.

isotropic and characterized by fluctuations of $0.005U_\infty$, with a turbulent length scale of 10 cm.

In terms of boundary conditions, with reference to Figure 4.1, we have:

- inflow conditions with fixed velocity and turbulence in inlet;
- no-slip walls on the car;
- slip wall on the ground;
- outflow with fixed pressure in outlet;
- symmetry on the roof.

The solution domain is discretized by a hex-dominant polyhedral mesh of 93388 cell, generated through the `snappyHexMesh` utility supplied with OpenFOAM®. A prismatic layer is created starting from the solid walls of both the car model and the ground, in order to assure a good resolution of the physical boundary layer, with a target y^+ of 40.

The solution is obtained through the resolution of the steady RANS equations on the computational domain, employing the Spalart-Allmaras turbulence model described in Section 3.2.5, with wall functions for near-wall treatment, as common practice for automotive applications.

In order to assess the accuracy of the proposed methodologies on some output of interest, we introduce the definition of the aerodynamic drag and lift coefficients

$$C_{\mathrm{x}} = \frac{\int_S (-p\mathbf{n} + \boldsymbol{\tau}) \cdot \mathbf{e}_x \, \partial S}{\frac{1}{2}\rho U_\infty^2 l_{\mathrm{f}}}, \qquad C_{\mathrm{z}} = \frac{\int_S (-p\mathbf{n} + \boldsymbol{\tau}) \cdot \mathbf{e}_z \, \partial S}{\frac{1}{2}\rho U_\infty^2 l_{\mathrm{f}}}$$

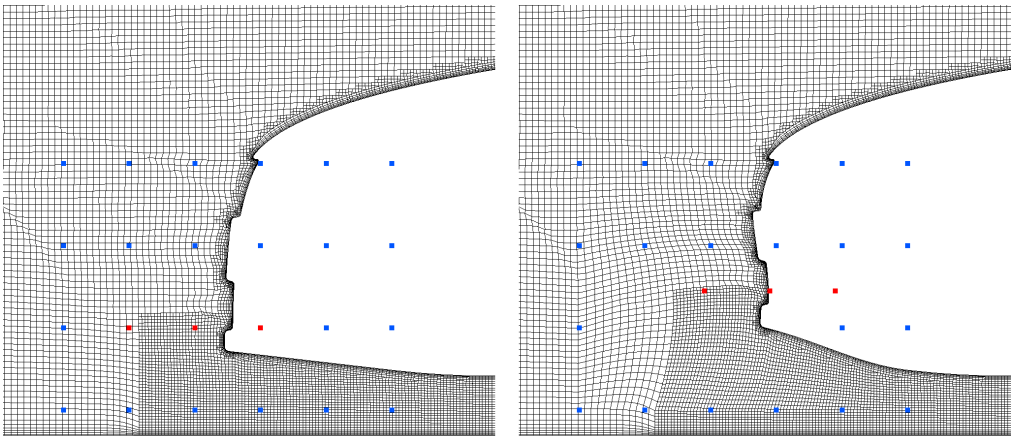


Fig. 4.2 Mesh morphing of the 2DCAR model: original (left) and deformed (right) configurations with corresponding lattices of control points. The red points are the ones allowed to move.

	x_{cp}	z_{cp}
lower bound	-0.18	-0.3
upper bound	0.18	0.3

Table 4.1 Limits of the parameter space.

resulting from the integration of the pressure and shear-stresses over the car profile. $\mathbf{n}$ is the outward-pointing unit vector normal to the profile, whereas $\mathbf{e}_x$ and $\mathbf{e}_z$ denote the unit vectors in the x and z directions, respectively. Both coefficients are adimensionalised with the reference dynamic pressure, i.e. $\frac{1}{2}\rho U_\infty^2$, and a characteristic length l_{f}, that in this case corresponds to the maximum thickness of the profile, equal to 1.21 m.

4.1.2 Parameterization

In this test case, we consider a simple geometry parameterization of the car front bumper, obtained through FFD-based mesh morphing technique and realised with the PyGem library. The deformation is controlled by a lattice of 6×4 control points, wrapped around the frontal region of the model. In order to guarantee a smooth transition between the deformable and fixed regions of the mesh, most of the points are fixed, apart from the three points highlighted in Figure 4.2, that are allowed to move in the x-z plane, resulting in the extension and compression of the bumper in the same directions. The same displacement is applied to the three control points: therefore, we have a 2-dimensional vector of parameters $\boldsymbol{\mu} = (x_{\text{cp}}, z_{\text{cp}})$, where x_{cp} denotes the displacement in the $x-$direction and z_{cp} the one along z. The limits of the parameter space, normalised with respect to the x and z extensions of the lattice, are listed in Table 4.1.

Chapter 5

Reduced-Order Modelling

Part of the work described in this chapter has been previously submitted for publication in Salmoiraghi et al. (2018) and Bergmann et al. (2018).

In this chapter we present the main techniques employed to derive a robust and efficient reduced-order model to be used as surrogate model during the optimization. In Section 5.1 the POD method is introduced, as well as its applications to model order reduction. Particular stress is placed on the state-of-the-art of reduced-order models for turbulent, large-scale, aerodynamic problems. Then, in Section 5.3 we described a hybrid full-order/reduced-order model based on domain-decomposition.

5.1 Proper Orthogonal Decomposition

The POD is a method that allows the computation of a low-dimensional representation of a high-dimensional state. It provides an optimally ordered, orthonormal basis starting from a set of empirical observations, e.g. experimental or computational data. It was first introduced by Lumley (1967) in the framework of fluid mechanics, as a method to identify coherent structures in turbulent flows, loosely defined as those regions of concentrated vorticity, characterised by a specific organization, scale and appreciable lifetime.

The same technique was developed independently in other fields, such as stochastic process modelling and statistical analysis, and it is also known as Karhunen-Loève expansions or decomposition (Kosambi (1943), Karhunen (1946), Loéve (1955)), Principal Component Analysis (Jolliffe (1986)) or Hotelling Analysis (Hotelling (1933)) and Singular

Value Decomposition (see for instance Golub and Van Loan (2013)). Beside from fluid mechanics, the method has been used in different applications, such as data compression, image processing, signal analysis and optimal control. A comprehensive description and review of the POD can be found for example in Aubry (1991), Berkooz et al. (1993), Holmes et al. (1996), Cordier and Bergmann (2003), Benner et al. (2015) and Chinesta et al. (2017).

As already stated, the fundamental idea behind the method is rather simple. Following the general context of approximation theory, we introduce the POD method directly in terms of its relationship with the SVD, focusing on its application to discretized flow solutions: analogous considerations, however, hold also for time-dependent problems. Let us consider a set of N snapshots $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ where each $\mathbf{y}_j = \mathbf{y}(\boldsymbol{\mu}_j) \in \mathbb{R}^n$ represents a steady flow solution computed for a given parameter value. Such snapshots can be arranged into a rectangular, real-valued matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$ with rank $r \leq \min\{n, N\}$, containing the $\mathbf{y}_j$ snapshot as its j^{th} column.

The SVD of $\mathbf{Y}$ can then be written as:

$$\mathbf{Y} = \boldsymbol{\Psi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \tag{5.1}$$

where the columns of the matrices $\boldsymbol{\Psi} \in \mathbb{R}^{n \times N}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N}$, namely $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_i$, are respectively the left and right singular vectors of $\mathbf{Y}$, whereas $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ is the $N \times N$ diagonal matrix containing the singular values of $\mathbf{Y}$ in decreasing order, also named POD singular values, i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. The vectors $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_i$ satisfy $\boldsymbol{Y}^\top \boldsymbol{\psi}_i = \sigma_i \boldsymbol{\phi}_i$ and $\boldsymbol{Y} \boldsymbol{\phi}_i = \sigma_i \boldsymbol{\psi}_i$ for $i = 1, ..., r$, and represent the eigenvectors of $\boldsymbol{Y}\boldsymbol{Y}^\top$ and $\boldsymbol{Y}^\top \boldsymbol{Y}$, respectively, with positive eigenvalues $\lambda_i = \sigma_i^2$.

Equation (5.1) can be rewritten in the form

$$\mathbf{Y} = \boldsymbol{\Psi}_r \mathbf{B}_r \quad \text{with} \quad \mathbf{B}_r = \boldsymbol{\Sigma}_r \boldsymbol{\Phi}_r^\top \tag{5.2}$$

where the subscript r means that the columns corresponding to zero singular values are discarded from the matrices. Thus, it is possible to express the columns $\mathbf{y}_j$ in terms of the r linearly independent columns $\boldsymbol{\psi}_i$ of $\boldsymbol{\Psi}_r$, with the coefficients of the linear expansion given by the columns of $\mathbf{B}_r$. Exploiting $\boldsymbol{\Psi}$ orthogonality, it follows that (see for instance Volkwein (2013)):

$$\mathbf{y}_j = \sum_{i=1}^{r} \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \boldsymbol{\psi}_i \quad \text{for} \quad j = 1, \ldots, N$$

where $\langle \cdot, \cdot \rangle_{\mathbb{R}^n}$ denotes the Euclidean inner product in $\mathbb{R}^n$.

Starting from this first remark, we now discuss the relationship between POD and SVD, referring to Volkwein (2013) for the mathematical demonstrations. In terms of POD, one of the main objectives is to provide an efficient representation of the data in $\boldsymbol{Y}$, by expressing the relevant information through a limited number $M_r \leq r$ of basis vectors. We can therefore define the POD basis as the solution, for any $M_r \in \{1, \dots, r\}$, of problem :

$$\max_{\widetilde{\boldsymbol{\psi}}_1, \cdots, \widetilde{\boldsymbol{\psi}}_{M_r} \in \mathbb{R}^n} \sum_{j=1}^{N} \sum_{i=1}^{M_r} \left| \langle \mathbf{y}_j, \widetilde{\boldsymbol{\psi}}_i \rangle_{\mathbb{R}^n} \right|^2 \quad \text{s.t.} \quad \langle \widetilde{\boldsymbol{\psi}}_i, \widetilde{\boldsymbol{\psi}}_j \rangle_{\mathbb{R}^n} = \delta_{ij} \, \text{for} \, 1 \leq i, j \leq M_r \ . \tag{5.3}$$

Problem (5.3) is an equality-constrained optimization problem, that is equivalent to find the stationary points of the Lagrangian

$$\mathcal{L}(\widetilde{\boldsymbol{\psi}}_1, \dots \widetilde{\boldsymbol{\psi}}_{M_r}, \boldsymbol{\Lambda}) = \sum_{j=1}^{N} \sum_{i=1}^{M_r} \left| \langle \mathbf{y}_j, \widetilde{\boldsymbol{\psi}}_i \rangle_{\mathbb{R}^n} \right|^2 + \sum_{i,j=1}^{M_r} \Lambda_{ij} \left(\delta_{ij} - \langle \widetilde{\boldsymbol{\psi}}_i, \widetilde{\boldsymbol{\psi}}_j \rangle_{\mathbb{R}^n} \right) \ ,$$

for $\widetilde{\boldsymbol{\psi}}_1, ..., \widetilde{\boldsymbol{\psi}}_{M-r} \in \mathbb{R}^n$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{M_r \times M_r}$, with elements Λ_{ij}, $1 \leq i, j \leq M_r$. Thus, the first-order necessary conditions for optimality are given by

$$\nabla_{\tilde{\boldsymbol{\psi}}_k} \mathcal{L}(\widetilde{\boldsymbol{\psi}}_1, \dots \widetilde{\boldsymbol{\psi}}_{M_r}, \boldsymbol{\Lambda}) = 0 \ , \qquad 1 \leq k \leq M_r. \tag{5.4}$$

From (5.4) it follows that

$$\sum_{j=1}^{N} \langle \mathbf{y}_j, \widetilde{\boldsymbol{\psi}}_k \rangle_{\mathbb{R}^n} \mathbf{y}_j = \frac{1}{2} \sum_{i=1}^{M_r} (\Lambda_{ik} + \Lambda_{ki}) \widetilde{\boldsymbol{\psi}}_i \ , \qquad 1 \leq k \leq M_r,$$

or, equivalently,

$$\mathbf{Y}\mathbf{Y}^\top \widetilde{\boldsymbol{\psi}}_k = \frac{1}{2} \sum_{i=1}^{M_r} (\Lambda_{ik} + \Lambda_{ki}) \widetilde{\boldsymbol{\psi}}_i \ , \qquad 1 \leq k \leq M_r. \tag{5.5}$$

Starting from $M_r = 1$ and proceeding by induction, it possible to verify that the conditions found in (5.5) yields to the symmetric $n \times n$ eigenvalue problem

$$\mathbf{Y}\mathbf{Y}^\top \widetilde{\boldsymbol{\psi}}_i = \lambda_i \widetilde{\boldsymbol{\psi}}_i \ , \tag{5.6}$$

of which we consider only the solutions for $i = 1, .., M_r$. Note that $\lambda_i = \Lambda_{ii}$, whereas $\Lambda_{ik} = -\Lambda_{ki}$ for $i \neq k$.

From the SVD of $\boldsymbol{Y}$, it follows that the singular vectors $\boldsymbol{\psi}_i$ solve Equation (5.6). Moreover, the terms $\boldsymbol{\psi}_i$ are a solution to Problem (5.3), whose maximum value is given by $\sum_{i=1}^{M_r} \sigma_i^2 = \sum_{i=1}^{M_r} \lambda_i$ (Volkwein (2013)).

Solving the eigenvalue problem associated with the spatial correlation matrix, $\mathbf{Y}\mathbf{Y}^\top$, or evaluating the SVD of $\mathbf{Y}$ is usually very expensive when n is large. To overcome such difficulty, if $N \ll n$, it is more convenient to compute the POD basis through the so-called *Method of snapshots* proposed by Sirovich (1987), by solving the equivalent eigenvalue problem

$$\mathbf{Y}^\top \mathbf{Y} \boldsymbol{\phi}_i = \lambda_i \boldsymbol{\phi}_i \quad \text{for } i = 1, \dots, M_r \tag{5.7}$$

and setting

$$\boldsymbol{\psi}_i = \frac{1}{\sqrt{\lambda_i}} \mathbf{Y} \boldsymbol{\phi}_i. \tag{5.8}$$

The $N \times N$ eigenproblem associated with the parameter correlation matrix, $\mathbf{Y}^\top \mathbf{Y}$, is then easier to deal with. Nevertheless, it is worth noting that for badly conditioned $\mathbf{Y}$ matrices, the correlation matrix would be worse conditioned than the original: in this eventuality, the SVD approach is more reliable to compute the information associated with small eigenvalues (Demmel (1997)).

To sum up, the following technical result holds.

Theorem 5.1. *Given the matrix* $\mathbf{Y} = [\mathbf{y}_1, \dots \mathbf{y}_N] \in \mathbb{R}^{n \times N}$ *with rank* $d \leq \min\{n, N\}$*, and let* $\mathbf{Y} = \boldsymbol{\Psi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top$ *being its SVD decomposition, then, for any* $M_r \in \{1, \dots, d\}$*, the singular vectors* ψ_i *are solutions to Problem (5.3), whose* arg max *is given by* $\sum_{i=1}^{M_r} \sigma_i^2 = \sum_{i=1}^{M_r} \lambda_i$ *.*

In the following sections, we further discuss some relevant properties of the POD method.

5.1.1 Properties of the POD basis

Assuming that all the hypothesis of Theorem 5.1 are satisfied, it can be shown that the first M_r POD modes capture more energy, in an average sense, than any other M_r-rank approximation of $\mathbf{Y}$. In order to prove such optimality property of the POD basis, we start by demonstrating that the POD coefficients are mutually uncorrelated and their mean square value corresponds to the eigenvalue itself (Volkwein (2013)).

Corollary 5.1 (Uncorrelated POD coefficients)**.** *Let all the hypotheses of Theorem 5.1 hold. Then*

$$\sum_{j=1}^{N} \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \langle \mathbf{y}_j, \boldsymbol{\psi}_k \rangle_{\mathbb{R}^n} = \lambda_i \delta_{ik} \qquad \text{for } 1 \leq i, k \leq M_r \ .$$

Proof. Exploiting the orthonormality of the POD modes, from (5.6) it follows that:

$$\sum_{j=1}^{N} \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \langle \mathbf{y}_j, \boldsymbol{\psi}_k \rangle_{\mathbb{R}^n} = \left\langle \underbrace{\sum_{j=1}^{N} \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \mathbf{y}_j}_{\mathbf{Y}\mathbf{Y}^\top \boldsymbol{\psi}_i}, \boldsymbol{\psi}_k \right\rangle_{\mathbb{R}^n} = \langle \sigma_i^2 \boldsymbol{\psi}_i, \boldsymbol{\psi}_k \rangle = \lambda_j \delta_{jk}$$

for $1 \leq i, k \leq M_r$. □

The notion of optimality of the POD method is then assessed in the following corollary.

Corollary 5.2 (Optimality of the POD basis)**.** *Let all the hypotheses of Theorem 5.1 hold and let*

$$\mathbf{y}_j = \sum_{i=1}^{M_r} \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \boldsymbol{\psi}_i$$

be the decomposition of the j^{th} column of $\mathbf{Y}$ with respect to the POD basis. If

$$\mathbf{y}_j = \sum_{i=1}^{M_r} \langle \mathbf{y}_j, \boldsymbol{\varphi}_i \rangle_{\mathbb{R}^n} \boldsymbol{\varphi}_i$$

denotes the approximation of $\mathbf{y}_j$ with respect to an arbitrary orthonormal basis, then, for every choice of M_r, it holds:

$$\sum_{i=1}^{M_r} \sum_{j=1}^{N} |\langle \mathbf{y}_j, \boldsymbol{\psi}_k \rangle_{\mathbb{R}^n}|^2 = \sum_{i=1}^{M_r} \lambda_i \geq \sum_{i=1}^{M_r} \sum_{j=1}^{N} |\langle \mathbf{y}_j, \boldsymbol{\varphi}_k \rangle_{\mathbb{R}^n}|^2 \tag{5.9}$$

Proof. The equality in (5.9) is easily verified from the proof of Corollary 5.1, whereas the inequality follows from Theorem 5.1. □

It is worth noting that the optimality condition also implies that (see Volkwein (2013))

$$\sum_{i=1}^{N} \left\| \mathbf{y}_j - \sum_{i=k}^{M_r} \langle \mathbf{y}_j, \boldsymbol{\psi}_k \rangle_{\mathbb{R}^n} \boldsymbol{\psi}_k \right\|_{\mathbb{R}^n}^2 \leq \sum_{i=1}^{N} \left\| \mathbf{y}_j - \sum_{i=k}^{M_r} \langle \mathbf{y}_j, \boldsymbol{\varphi}_k \rangle_{\mathbb{R}^n} \boldsymbol{\varphi}_k \right\|_{\mathbb{R}^n}^2 ,$$

i.e. the POD basis minimizes the least-squares error of the snapshot reconstruction. Therefore, it can also be determined by solving the equivalent of Problem (5.3), that is

$$\min_{\tilde{\boldsymbol{\psi}}_1,\cdots,\tilde{\boldsymbol{\psi}}_{M_r}\in\mathbb{R}^n} \sum_{i=1}^{N} \left\| \mathbf{y}_j - \sum_{i=k}^{M_r} \langle \mathbf{y}_j, \tilde{\boldsymbol{\psi}}_k \rangle_{\mathbb{R}^n} \tilde{\boldsymbol{\psi}}_k \right\|_{\mathbb{R}^n}^2 \quad \text{s.t. } \langle \tilde{\boldsymbol{\psi}}_i, \tilde{\boldsymbol{\psi}}_j \rangle_{\mathbb{R}^n} = \delta_{ij} \quad \text{for } 1 \leq i,j \leq M_r \ , \tag{5.10}$$

whose minimum value is given by $\sum_{i=M_r+1}^{N} \sigma_i^2 = \sum_{i=M_r+1}^{N} \lambda_i$.

The POD is then the most efficient among all linear decompositions, in the sense that, given a certain M_r, the projection on the subspace spanned by the M_r leading eigenfunctions will have the greatest possible energy content. In view of model order reduction, this aspect suggests that the number of modes necessary to described the solution space is indeed quite small. Therefore, the choice of M_r represents a critical task in the tuning of any POD-based ROM. A long-established approach (Cordier and Bergmann (2003)) exploits the notion that the λ_i eigenvalues permit to estimate the energy contribution of each mode to the reconstruction of the snapshots. The so-called Relative Information Content (RIC) indicator for a basis of size M_r, is then defined as

$$\mathrm{RIC}(M_r) = \frac{\sum_{i=1}^{M_r} \lambda_i}{\sum_{i=1}^{N} \lambda_i}, \qquad \text{with } 0 < M_r \leq N \ ,$$

whose denominator represents the total information content of the snapshots. Assuming that the eigenvalues are ordered in decreasing values, the POD basis size is chosen by truncating the expansion after the first M_r terms that satisfy the condition $\mathrm{RIC}(M_r) \geq \tau_E$, where τ_E represents a predefined percentage of energy, e.g. 99.9% or greater.

From Equation (5.8) it is also clear that the POD basis space is a linear combination of the solution space spanned by empirical observations, i.e. the snapshots. Thus, the POD modes inherit all the linear properties of the original realizations. For incompressible flows, for example, if we built the basis starting from a set of velocity fields $\mathbf{u}_j$, the resulting eigenmodes will be divergence free, since the velocity field is solenoidal. Moreover, the modes verifies the boundary conditions of the flow realizations.

In this framework, it is often common practice to introduce a forcing field in the POD projection operator

$$\Pi^{\mathrm{POD}}(\mathbf{y}) := \bar{\mathbf{y}} + \sum_{i=1}^{M_r} \alpha_i \boldsymbol{\psi}_i \quad \text{with } \alpha_i = \langle \mathbf{y} - \bar{\mathbf{y}}, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n} \tag{5.11}$$

and set this additional term equal to the average field among the flow realizations, i.e. $\bar{\mathbf{y}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{y}_j$. This is useful for example in external aerodynamic problems, where the average velocity field fixes the far-field velocity: in this way, all the corresponding modes will tend to zero in the far-field. As a consequence the smallest POD eigenvalue is equal to zero and its corresponding mode should be omitted, setting $M_r \leq N-1$. In the applications of interest for this work, all the POD bases will be defined introducing this forcing term.

5.1.2 Weighted inner product

So far, we have described the POD method in the context of the standard L^2 inner product. In fluid dynamics, this is usually the natural choice, since it corresponds to optimality in the sense of the flow kinetic energy. However, for some applications, employing different definitions, such as the H^1 inner product, could be beneficial for the robustness of the ROM, as hinted in Iollo et al. (2000).

Since our problems are often defined on unstructured computational grids, it is natural for us to write the discrete L^2 inner product as a weighted inner product in the form

$$\langle \boldsymbol{\psi}, \tilde{\boldsymbol{\psi}} \rangle_W = \boldsymbol{\psi}^\top \mathbf{W} \tilde{\boldsymbol{\psi}} \qquad \text{for } \boldsymbol{\psi}, \tilde{\boldsymbol{\psi}} \in \mathbb{R}^n,$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a symmetric, positive definite matrix, whose principal diagonal contains the cell volumes. The associated induced norm is then $\|\boldsymbol{\psi}\|_W = \sqrt{\langle \boldsymbol{\psi}, \boldsymbol{\psi} \rangle_W}$. Following the same steps as above, the optimality condition yields the generalised eigenvalue problem

$$(\mathbf{WY})(\mathbf{WY})^\top \boldsymbol{\psi}_i = \lambda_i \mathbf{W}\boldsymbol{\psi}_i \qquad \text{for } i = 1, ..., M_r. \tag{5.12}$$

Given that $\mathbf{W}$ is symmetric and positive definite, we can multiply (5.12) by $\mathbf{W}^{-\frac{1}{2}}$ from the left, obtaining the eigenproblem

$$\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top \hat{\boldsymbol{\psi}}_i = \lambda_i \hat{\boldsymbol{\psi}}_i \qquad \text{for } i = 1, ..., M_r$$

with $\hat{\mathbf{Y}} = \mathbf{W}^{\frac{1}{2}}\mathbf{Y}$ and $\hat{\boldsymbol{\psi}}_i = \mathbf{W}^{\frac{1}{2}}\boldsymbol{\psi}_i$, or equivalently

$$\mathbf{Y}^\top \mathbf{W} \mathbf{Y} \hat{\boldsymbol{\phi}}_i = \lambda_i \hat{\boldsymbol{\phi}}_i \qquad \text{for } i = 1, ..., M_r.$$

Clearly, the resulting eigenmodes $\hat{\boldsymbol{\psi}}_i = \frac{1}{\sqrt{\lambda_i}}\hat{\mathbf{Y}}\hat{\boldsymbol{\phi}}_i$, are now orthonormal with respect to the weighted inner product, whereas

$$\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle_W = \boldsymbol{\psi}_i^\top \mathbf{W} \boldsymbol{\psi}_j = \frac{\lambda_j}{\sqrt{\lambda_i \lambda_j}} \delta_{ij} \qquad \text{for } 1 \leq i, j \leq M_r.$$

5.2 POD-based reduced models

As in classical model reduction strategies, POD-based ROMs usually follow the offline-online paradigm. The general idea is to decompose the method in two phases. During the offline stage, first a database of solution snapshots is generated through expensive full-order simulations, in order to compute the POD basis following the approaches described in Section 5.1. Next, the dimensionality and complexity of the problem are reduced by projecting the full model onto the subspace spanned by the POD modes and introducing other approximations, such as interpolation, to efficiently represent non-linearities and parameter dependence. In the online phase, instead, the ROM is used to obtain rapid responses varying the parameters of interest.

Commonly used POD-based reduction strategies in fluid dynamics include POD-Galerkin and POD with interpolation (PODI). The first method relies on the Galerkin-projection of the NSE onto the POD modes, in order to built a low order approximation of the equations (see Section 5.2.1). The resulting model shows a strong reduction with respect to the original one, and can be used for flow estimation and optimal control. However, the ability of the ROM to perform *predictive* simulations is generally affected by stability and robustness issues, that make the approach quite intrusive and dependent on the application and flow features. Moreover, a very fine sampling of the parameter space is usually required to obtain accurate results. PODI, instead, is based on the simple idea that if the POD expansion coefficients are smooth functions of the parameters, interpolation methods can be used to determine the values corresponding to snapshots not included in the original ensemble (see Section 5.2.2). This simple strategy allows us to significantly reduce the computation time without any concern in terms of stability, but, again, out-of-sample accuracy can be attained only with an adequately large initial database of simulations. Another possibility, mostly used in aeronautic applications, consists in restricting the standard full-order solver to the reduced POD space and achieving model reduction by residual minimization, replacing the fluid dynamics governing equation with a least-squares optimization problem (see Section 5.2.3). This method has a limited impact on canonical CFD solvers and usually leads to more accurate results than PODI,

but at a higher computational cost. However, all these POD-based methods suffer from the fact that the POD basis is a linear combination of the solution space, trying to represent non-linear phenomena.

To overcome some of these difficulties, it is possible to exploit a hybrid full-order/reduced-order method, namely the zonal-POD approach, that will be thoroughly discussed in Section 5.3 and following. The method is based on a Schwarz-type domain decomposition method, and guarantees a moderate reduction with respect to standard ROMs: nevertheless, thanks to the full-order feedback, it has been proven to be stable and robust, with a good prediction accuracy even with a coarse initial sampling of the parameter space.

In order to compare the ROMs outlined in the coming sections, we take into consideration six main criteria, that will be addressed in the following sections:

- **online reduction**, i.e. the computation speed-up observed during the online phase, defined as the ratio between the cost of a FOM evaluation over a ROM evaluation, both expressed in computational time: $\text{cput}_{\text{FOM}}/\text{cput}_{\text{ROM}}$;
- **offline cost**, i.e. the overall cost of the offline phase, evaluated as $N_{\text{FOM}} \times \text{cput}_{\text{FOM}}$, where N_{FOM} indicates the number of FOM evaluations required in order to train the underlying POD model. Such aspect is often not discussed in model reduction literature, but it cannot be overlooked in an industrial context and in particular for optimization problems: if the offline cost is comparable with the cost of a standard optimization loop (without model reduction), it is not clear why we should introduce an additional layer of complexity given by the ROM;
- **range of applicability** of the reduction strategy to problems characterised by different parameterization, flow regimes, properties and conditions, without significant modifications;
- **intrusiveness**, in terms of numerical implementation. Generally speaking, non-intrusive methods do not rely on the discretization of the FOM: hence, they do not require substantial rewriting of the numerical schemes and are easy to implement even when the source code is not available. This aspect is particularly relevant when commercial codes are employed, as frequently happens in industry;
- **stability** of the numerical model, in the sense that approximation errors in the calculation are not magnified, resulting in the simulation to explode;

- **robustness**, or rather the ability of the model to perform predictive simulations even when its tuning is not optimal, without producing extremely different results for small changes in its setup.

5.2.1 POD-Galerkin

The POD-Galerkin method permits to reduce the NSE to a low-dimensional system of ordinary differential equations (ODEs), through Galerkin projection. In general, the model is derived for time-varying problems, by replacing the $\mathbf{u}$ expansion (from Equation 5.11) into the NSE and applying a Galerkin projection of the resulting system onto the subspace spanned by the POD modes. If we consider for instance the incompressible NSE, we obtain the following dynamic system:

$$\begin{aligned}\frac{d\alpha_j(\boldsymbol{\mu},t)}{dt} &= A_j + \sum_{i=1}^{M_r} C_{ij}\alpha_i(\boldsymbol{\mu},t) - \sum_{i=1}^{M_r}\sum_{k=1}^{M_r} B_{ikj}\alpha_i(\boldsymbol{\mu},t)\alpha_k(\boldsymbol{\mu},t) - D_j \ , \\ \alpha_j(\boldsymbol{\mu},0) &= \langle \mathbf{u}(\boldsymbol{\mu},0), \boldsymbol{\varphi}_j\rangle_\Omega\end{aligned}$$

for $j = 1, \dots, M_r$, where

$$\begin{aligned}A_j &= -\langle (\bar{\mathbf{u}}\cdot\nabla)\,\bar{\mathbf{u}}, \boldsymbol{\varphi}_j\rangle_\Omega + \nu\langle \Delta\bar{\mathbf{u}}, \boldsymbol{\varphi}_j\rangle_\Omega \\ B_{ikj} &= \langle (\boldsymbol{\varphi}_i\cdot\nabla)\,\boldsymbol{\varphi}_k, \boldsymbol{\varphi}_j\rangle_\Omega \\ C_{ij} &= -\langle (\bar{\mathbf{u}}\cdot\nabla)\,\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j\rangle_\Omega - \langle (\boldsymbol{\varphi}_i\cdot\nabla)\,\bar{\mathbf{u}}, \boldsymbol{\varphi}_j\rangle_\Omega + \nu\langle \Delta\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j\rangle_\Omega \\ D_j &= \frac{1}{\rho}\langle \nabla p, \boldsymbol{\varphi}_j\rangle_\Omega \ .\end{aligned}$$

Generally speaking, this class of methods is used mainly in time-dependent flow control problems, as in Noack et al. (2011), Weller et al. (2009a,b), Kunisch and Volkwein (2002) and Hinze and Volkwein (2005)) but they can also be employed in PDE-constrained optimization and in steady problems, often coupled with EIM and DEIM techniques in order to reduce the complexity of the evaluation of the non-linear term. Similar models are derived for RANS and LES approaches. For incompressible flows, note that since the solution snapshots satisfy the continuity equation, also the modes, that are a linear combination of said snapshots (as follows from Equation (5.8)), are divergence-free. This implies that in the Galerkin projection of the pressure term,

$$\langle \nabla p, \boldsymbol{\varphi}_j\rangle_\Omega = \int_\Omega \nabla p\cdot\boldsymbol{\varphi}_j dv = -\int_\Omega p\,(\nabla\cdot\boldsymbol{\varphi}_j)\,dv + \int_{\partial\Omega} p\,(\boldsymbol{\varphi}_j\cdot\mathbf{n})\,ds \ ,$$

the only contribution is given by the surface integral. If the velocity field is almost constant at the boundaries, the resulting POD modes are almost null and so the pressure term can be neglected. This is generally true when the computational domain is large enough, or for confined flows. In any other cases, additional terms or a POD basis for the pressure need to be introduced. A review of the problem can be found for instance in Noack et al. (2005).

One of the main issues of the POD-Galerkin method, is that it may give unstable results, especially for flows characterized by high Re, because the stabilization effect due to the truncation of the small scales is lost. A typical remedy to this problem consists in the introduction of artificial dissipation terms, as in Bergmann et al. (2009), Wang et al. (2012), Weller et al. (2010), Buffoni and Willcox (2010) and Östh et al. (2014). Alternatively, methods based on Petrov-Galerkin projections are also employed (Giere et al. (2015), Carlberg et al. (2011)). The main issue with such stabilization methods, however, is that they are usually dependent on the flow features, and cannot be extended to a general case.

The benefits and drawbacks of the method, can be outlined as follows:

- the approach is characterised by a strong online reduction, that allows us to reduce the complexity of the computations by several order of magnitude, passing from $O(10^6 - 10^7)$ degrees of freedom of the FOM discretization, to $O(10 - 10^2)$, i.e. the coefficients of the POD expansion;
- the prediction accuracy strongly depends on the sampling used to train the underlying POD model and a sufficiently high number of snapshots is usually required: as a consequence, the offline phase is extremely expensive.
- it represents an intrusive model reduction technique, since it requires to derive the system of ODE for the application of interest;
- to assure stability, problem-dependent stabilisation terms need to be introduced, further limiting the range of applicability of the method;
- in terms of robustness, it is difficult to perform predictive simulations when the model setup is not optimal.

POD-Galerkin methods usually show reasonable accuracy and are widely studied in literature, even in turbulent flows; nevertheless, the intrusiveness of the approach, coupled with stability and robustness issues, made its application to industrial problems arduous.

5.2.2 PODI

POD with interpolation was first introduce in the work of Bui-Thanh (2003) and it has been recently used in aerodynamic and automotive applications, as in Dolci and Arina (2016) and Salmoiraghi et al. (2018), with good results. As hinted before, the rationale behind the PODI methods regards the evaluation of the POD coefficients for out-of-sample configurations by interpolation. As seen in Section 5.1, each database snapshot $\mathbf{y}_j = \mathbf{y}(\boldsymbol{\mu}_i) \in \mathbb{R}^N$ can be expanded as

$$\mathbf{y}_j = \sum_{i=1}^{M_r} \alpha_{ij} \boldsymbol{\psi_i} \ ,$$

where the parameter dependence is limited to the projection coefficients $\alpha_{ij} = \langle \mathbf{y}_j, \boldsymbol{\psi}_i \rangle_{\mathbb{R}^n}$. Note that the same considerations hold if we consider a forcing field in the POD projection operator. By construction the α_{ij} are discrete functions of the $\boldsymbol{\mu}_j$ points of the parameter space, but they can be transformed into continuous functions α_i in order to perform predictions, assuming a smooth dependence of the coefficients on the parameters. Thus, in any generic parameter point, the corresponding solution can be approximated by

$$\mathbf{y}^\star(\boldsymbol{\mu}) = \sum_{i=1}^{M_r} \alpha_i(\boldsymbol{\mu}) \boldsymbol{\psi_i} \ .$$

Clearly, the approximation error is guaranteed to be null only in the points $\boldsymbol{\mu}_i \in \boldsymbol{\Xi}$, where $\mathbf{y}(\boldsymbol{\mu}_j) = \mathbf{y}^\star(\boldsymbol{\mu}_j)$. The accuracy on the prediction for out-of-sample configuration, instead, is determined by both the distribution and the number of the $\mathbf{y}_j$ solutions, as well as by the interpolant employed, unless a fine sampling is provided. For one-dimensional problems, linear or spline interpolation is generally used (Bui-Thanh et al. (2003)), whereas response surfaces are adopted in the multidimensional case. Rather than regression techniques, that tend to introduce an undesirable smoothing of the data, e.g. in order to filter numerical or experimental noise, multivariate interpolation seems to be preferable, since it guarantees the fitting of the data, and therefore the consistency of the method for in-sample points. A few examples of application, exploiting both least-square regression and radial basis functions are reported in Dolci and Arina (2016), whereas in Salmoiraghi et al. (2018) a natural neighbour interpolation based on the Voronoi tessellation of the database points is used in an industrial framework.

The strengths of the PODI strategy can be summarised in terms of its non-intrusiveness and ease to use, making it suitable for large-scale industrial applications and turbulent flows. More specifically:

- the method does not rely neither on the chosen numerical discretization, nor on the flow features, and it can be straightforwardly employed in a broad class of fluid dynamic problems;
- the associated reduction is remarkable during the online stage ($O(10^6 - 10^7)$), since the interpolation time is negligible and the reconstruction step has a small cost compared to the full-order simulation, enabling the real-time processing of the flow variables;
- convergence or stability are not an issue;
- a parameterization is required, in order to perform the interpolation, but its features are not relevant.

On the other hand, the out-of-sample performance of the method strongly depends on the sampling of the parameter space: a coarse or not significant initial sampling will likely lead to inaccurate results. In order to increase robustness, an extensive offline phase is required to train the POD model.

5.2.3 POD with residual minimization

The idea behind this method was first proposed by LeGresley and Alonso (2000) in the framework of airfoil design. Recent applications for compressible flows include the works of Zimmermann and Görtz (2010), Zimmermann and Görtz (2012) and Amsallem et al. (2013). In this strategy, the coefficients of the POD-expansion are determined by minimizing, in the least-squares sense, the associated CFD residual, which is evaluated through the standard full-order solver. By introducing the POD representation of an out-of-sample solution snapshots, i.e. $\mathbf{y}^\star(\alpha) = \sum_{i=1}^{M_r} \alpha(\boldsymbol{\mu})\boldsymbol{\psi_i}$, the problem reduces to solve

$$\min_{\alpha} \|\mathbf{r}\,(\alpha)\,\|_{L^2}^2 \ ,$$

where $\mathbf{r}$ denotes the residuals.

In terms of robustness, it appears that the performance of the method strongly depends on the definition of such residual, as stressed in Amsallem et al. (2013) for unsteady CFD applications. In addition, also in this approach the snapshots sampling represents a crucial aspect in the ROM construction, that usually requires $O(10 - 10^2)$ full-order solutions, depending on the complexity of the problem.

Among the benefits of the method, it is worth citing the following ones:

- the impact on the full-order solver is very limited, and therefore it represents a non-intrusive tool for model order reduction;
- with respect to PODI, the POD coefficients corresponding to the snapshots used in the model construction are not used during the evaluation of the coefficients for out-of-sample configurations: as a consequence, the method is more suited for extrapolation than interpolation-based approaches;
- the features of the underlying parameterization are not relevant;
- it is potentially suited for a wide range of applications, since it does not depend neither on the numerical discretization nor on the flow characteristics, although particular attention must be placed on the residuals definition.

In terms of reduction of the computational cost, the method shows a moderate gain in the online phase ($O(10 - 10^2)$), but such performance may be improved when coupled with other techniques, such as Missing Point Estimation (MPE) and DEIM. In fact, as in POD-Galerkin methods, the evaluation of the non-linear residual may require the introduction of additional techniques in order to achieve any speed-up. Despite showing promising results for unsteady, compressible, aerodynamic problems, there are no examples in literature of its application to incompressible turbulent flows, to the best of the authors' knowledge.

5.3 Zonal-POD

As mentioned in the previous section, POD-based ROMs generally suffer from the fact that the basis space is a linear combination of the solution space spanned by the snapshots: this means that the model is not capable to represent non-linear phenomenology, unless a sufficient rich database is provided, as happens in standard POD-Galerkin and PODI methods. In order to overcome such difficulties, a hybrid full-order/reduced-order method, also knows as zonal-POD, can be used. The main idea behind this hybrid approach, first proposed by Buffoni et al. (2009), is to split the domain of interest in two subdomains and to use a different approximation method in each region. More specifically, the full-order model is employed where the effects of non-linearities and geometry variations are predominant, whereas linear and weakly non-linear phenomenology is addressed by the reduced-model. The low-dimensional modal representation of the snapshot data is computed offline, using the POD method introduced in Section 5.1. During the online

phase, instead, the two models are coupled through a Schwarz-type domain decomposition method in order to perform predictive simulations for new values of the parameter $\boldsymbol{\mu}$.

A similar notion can be found in Lucia et al. (2003), where POD is coupled with domain decomposition for the first time, in the analysis of 2-dimensional high-speed flow fields with moving shock waves. In such zonal approach, the region interested by the shock wave is described through the FOM or an accurate ROM, whereas in the majority of the domain a less accurate ROM is used, with satisfactory results. In LeGresley and Alonso (2003), the same concept is instead applied to derive an *a posteriori* correction of the original POD approximation.

A detailed description of the method is provided in the following, whereas the algorithm convergence is addressed in Section 5.3.4

5.3.1 Schwarz-POD iterative algorithm

With reference to the Navier-Stokes problem introduced in Section 1.1 (Problem (1.2)), we decompose the initial domain into two regions, $\Omega_1(\boldsymbol{\mu})$ and Ω_2, by exploiting this rationale: the former domain, $\Omega_1(\boldsymbol{\mu})$, is such that its boundary $\partial\Omega_1(\boldsymbol{\mu})$ contains the portion of $\partial\Omega(\boldsymbol{\mu})$ depending on $\boldsymbol{\mu}$, whereas the latter, Ω_2, is not dependent on the parameter. Furthermore, we assume that:

$$\Omega(\boldsymbol{\mu}) = \Omega_1(\boldsymbol{\mu}) \cup \Omega_2, \qquad \Omega_0 := \Omega_1(\boldsymbol{\mu}) \cap \Omega_2 \neq \emptyset$$

where the non-empty set Ω_0 represents the *overlapping region* shared by both subdomains, as sketched in Figure 5.1. Its boundary $\Gamma := \partial\Omega_0$ can be split into

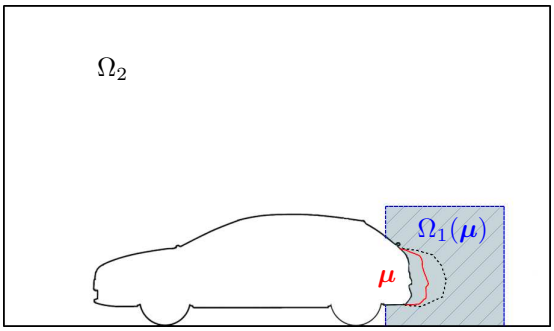
$$\Gamma_1 := \Gamma \cap \Omega_2, \qquad \Gamma_2 := \Gamma \cap \Omega_1(\boldsymbol{\mu}), \qquad \Gamma_0 := \partial\Omega_0 \backslash (\Gamma_1 \cup \Gamma_2) \subset \partial\Omega(\boldsymbol{\mu})$$

where the internal boundary Γ_1 is assumed to be independent on $\boldsymbol{\mu}$. Γ_0, instead, is a subset of the original boundary $\partial\Omega(\boldsymbol{\mu})$.

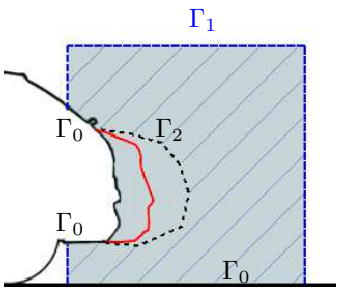
The classical Schwarz' method (see for instance Quarteroni and Valli (1999) or Mathew (2008)) defines a sequence for $k \geq 1$:

$$(\mathbf{u}_1, p_1)^{(1)} \mapsto (\mathbf{u}_2, p_2)^{(1)} \mapsto \cdots \mapsto (\mathbf{u}_1, p_1)^{(k)} \mapsto (\mathbf{u}_2, p_2)^{(k)} \mapsto \cdots \tag{5.13}$$

where each $(\mathbf{u}_i, p_i)^{(k)}$ for $i = 1, 2$ is an approximation of the exact solution restricted to the respective subdomain, i.e. $(\mathbf{u}, p)_{|\Omega_i}$, at the k^{th} iteration of the algorithm.



(a) Ω_1 (light blue), $\Omega_2(\boldsymbol{\mu})$ (white) and Ω_0 (diagonal pattern).



(b) Detail of boundaries.

Fig. 5.1 Example of domain decomposition in the framework of the zonal-POD approach.

In the following we describe how each stage of Sequence (5.13) is realized, in the framework of the Schwarz-POD iterative method.

Stage 1 $(\mathbf{u}_2, p_2) \mapsto (\mathbf{u}_1, p_1)$. Given $(\mathbf{u}_2, p_2)$ in Ω_2, consider its restriction to Γ_1. We define $(\mathbf{u}_1, p_1)$ as the solution of Problem (1.2) restricted to the $\Omega_1(\boldsymbol{\mu})$ domain:

$$\begin{aligned} &NS(\mathbf{u}_1, p_1) = 0 && \text{in } \Omega_1(\boldsymbol{\mu}) \ , \\ &B(\mathbf{u}_1, p_1) = 0 && \text{on } \partial\Omega_1(\boldsymbol{\mu}) \cap \partial\Omega(\boldsymbol{\mu}) \ , \\ &C(\mathbf{u}_1, p_1) = C(\mathbf{u}_1, p_1) && \text{on } \Gamma_1 \ , \end{aligned} \tag{5.14}$$

where $C(\mathbf{u}, p) = g$ denotes an admissible boundary condition on Γ_1 for the NSE, e.g. Dirichlet or Neumann boundary conditions, and the last equation represents a matching condition between the two approximations of the solution, namely $(\mathbf{u}_1, p_1)$ and $(\mathbf{u}_2, p_2)$, across the interface.

Stage 2 $(\mathbf{u}_1, p_1) \mapsto (\mathbf{u}_2, p_2)$. Given $(\mathbf{u}_1, p_1)$ in $\Omega_1(\boldsymbol{\mu})$, we recover $(\mathbf{u}_2, p_2)_{|\Omega_0}$ by solving the least-squares problems

$$\min_{\alpha} \left\| \mathbf{u}_1 - \bar{\mathbf{u}}_{|\Omega_0} - \sum_{i=1}^{M_r^{\psi}} \alpha_i \boldsymbol{\psi}_{i|\Omega_0} \right\|_{\Omega_0}^2$$

$$\min_{\beta} \left\| p_1 - \bar{p}_{|\Omega_0} - \sum_{i=1}^{M_r^{\varphi}} \beta_i \varphi_{i|\Omega_0} \right\|_{\Omega_0}^2$$

that minimize, in the least-square sense, the distance between the full-order solution $(\mathbf{u}_1, p_1)$ and the its POD reconstruction over Ω_0. $(\mathbf{u}_2, p_2)_{|\Omega_0}$ can be easily extended to the whole Ω_2. By construction, in fact, each field is a linear combination of the snapshots

restricted to Ω_0, thus we have:

$$\mathbf{u}_2 = \bar{\mathbf{u}}_{|\Omega_2} + \sum_{i=1}^{M_r^\psi} \alpha_i \boldsymbol{\psi}_{i|\Omega_2} \ , \qquad p_2 = \bar{p}_{|\Omega_2} + \sum_{i=1}^{M_r^\varphi} \beta_i \psi_{i|\Omega_2} \ . \tag{5.15}$$

This extension, however, is not required during the algorithm iterations, since in practice we only need $(\mathbf{u}_2, p_2)$ on Γ_1, which is a subset of $\partial\Omega_0$. If convergence is attained, i.e. if there exist $(\mathbf{u}_1^\star, p_1^\star)$ and $(\mathbf{u}_2^\star, p_2^\star)$ such that

$$\lim_{k\to\infty} (\mathbf{u}_1, p_1)^{(k)} = (\mathbf{u}_1^\star, p_1^\star), \qquad \lim_{k\to\infty} (\mathbf{u}_2, p_2)^{(k)} = (\mathbf{u}_2^\star, p_2^\star),$$

the following matching condition holds:

$$C(\mathbf{u}_1^\star, p_1^\star) = C(\mathbf{u}_2^\star, p_2^\star) \qquad \text{on } \Gamma_1 \ .$$

While $(\mathbf{u}_1^\star, p_1^\star)$ satisfies $NS(\mathbf{u}_1^\star, p_1^\star) = 0$ in $\Omega_1(\boldsymbol{\mu})$, and the original problem boundary conditions $B(\mathbf{u}_1^\star, p_1^\star)$ on $\Omega_1(\boldsymbol{\mu})\backslash\Gamma_1$, $(\mathbf{u}_2^\star, p_2^\star)$ will not be a solution of the Navier-Stokes problem in Ω_2. This is due to the fact that the Navier-Stokes operator is nonlinear, hence the linear combination of the solution snapshots is not, in turn, a solution, except for special circumstances. By construction, however, $(\mathbf{u}_2^\star, p_2^\star)$ is divergence-free in Ω_2. A further investigation of the convergence properties of the Schwarz-POD algorithm is presented in Section 5.3.4.

The iterative method is then integrated within the zonal-POD approach, whose details are summarized in Algorithm 2. In particular, every Schwarz-POD iteration is coupled with a step of the full-order solver, e.g. an iteration of the SIMPLE algorithm.

Algorithm 2 Zonal-POD algorithm

1: set initial boundary conditions for $(\mathbf{u}_1(\boldsymbol{\mu}), p_1(\boldsymbol{\mu}))^{(0)}$ on $\partial\Omega_1$
2: **while** *convergence* = false **do**
3: evaluate $(\mathbf{u}_1, p_1)^{(k)}$ by integrating the governing equations in $\Omega_1(\boldsymbol{\mu})$
4: $\alpha \leftarrow \arg\min_\alpha \left(\| \mathbf{u}_1 - \bar{\mathbf{u}} - \sum_{i=0}^{M_r^\psi} \alpha_i \boldsymbol{\psi}_i \|_{\Omega_0}^2 \right)$
5: $\beta \leftarrow \arg\min_\beta \left(\| p_1 - \bar{p} - \sum_{i=1}^{M_r^\varphi} \alpha_i \varphi_i \|_{\Omega_0}^2 \right)$
6: $(\mathbf{u}_2(\boldsymbol{\mu}), p_2(\boldsymbol{\mu}))^{(k)} \leftarrow \alpha, \beta$
7: evaluate $(\mathbf{u}_2, p_2)^{(k)}_{|\Gamma_1}$
8: update boundary conditions for $(\mathbf{u}_1, p_1)^{(k+1)}$ on Γ_1
9: check for *convergence*
10: **end while**
11: retrieve the solution on Ω_2 (if needed)

With respect to the original work from Buffoni et al. (2009), where the hybrid method is applied to Laplace and Euler equations, we generalize the approach for incompressible Navier-Stokes problems. The proposed method is straightforwardly applied to turbulent flows by substituting the NS operator with the RANS operator: in this case, also the turbulent variables introduced by the closure model will be represented by a separate POD expansion.

Although this thesis is mainly developed in the framework of automotive and naval applications, hence for incompressible flows, it is worth noting that the method may be extended also to compressible flows, introducing the POD approximation of the additional flow variables, i.e. density and internal energy. This generalization, even though easy in principle, implies the definition of a new set of internal boundary conditions on Γ_1, in order to attain a well-posed mathematical problem, and may require extra care for the presence of shock waves, that cannot be accurately described by the ROM.

5.3.2 Numerical implementation

In this section, we report the details of the OpenFOAM® implementation of the hybrid strategy, with particular focus on the boundary conditions treatment on the fluid interface Γ_1.

Implementation of the zonal-POD method

In the `simpleFoam` code, we introduce the following modifications:

```
#include "initialize.H"
#include "createModes.H"
#include "refValues.H"

<...>
while (simple.loop())
{
    // --- Pressure-velocity SIMPLE corrector
    {
    #include "UEqn.H"
    #include "pEqn.H"
    }
```

```
    turbulence->correct();

    #include "calcPODCoeffs.H"
    #include "updateBoundaryField.H"

}
```

Before entering the solution loop, we load the POD modes, initialize the fields with $\bar{\mathbf{y}}$ and pre-compute the minimization matrix $\mathbf{A}$ of the algebraic system $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$, with $A_{jk} = \langle \boldsymbol{\psi}_j, \boldsymbol{\psi}_k \rangle_{\Omega_0}$ and $b_j = \langle \mathbf{y}_1 - \bar{\mathbf{y}}, \boldsymbol{\psi}_j \rangle_{\Omega_0}$, resulting from the least-square problem in Algorithm 2. At every solver iteration, instead, we compute the right-hand side $\mathbf{b}$ of the minimization problem and we solve the linear system in the variables given by the coefficients of the POD expansion (`calcPODCoeffs.H`); once the POD coefficients are known, the boundary conditions on the fluid interface are updated (`updateBoundaryField.H`). Since the size of the minimization problem is small (we employ $O(10)$ POD modes), the cost of the CFD solver iteration is not affected by this modifications, that are negligible with respect to the resolution of the fluid dynamics. Note that the code modifications do not enter in the equations discretization, nor alter the SIMPLE loop.

During the full-order simulation over $\Omega_1(\boldsymbol{\mu})$, there is no need to retrieve the POD solution $\mathbf{y}_2$ over the entire Ω_2 domain, but it is sufficient to evaluate its trace over Γ_1. Therefore, we extend $\mathbf{y}_2$ outside of the reduced domain through a dedicated post-processing utility after the solution on Ω_1 is converged.

Implementation of the boundary conditions

For physical boundaries, i.e. those in common between $\partial\Omega(\boldsymbol{\mu})$ and $\partial\Omega_1(\boldsymbol{\mu})$, when simulating the flow in the reduced domain we impose the same conditions defined for the FOM. The boundary conditions on Γ_1, instead, are updated at every solver iteration solving the minimisation problem described in Section 5.3. Due to the nature of the problem and its numerical discretization, it is not possible to use Dirichlet conditions on all the faces belonging to the boundary Γ_1.

Dirichlet or Neumann conditions are then imposed depending on the sign of the velocity flux through the boundary faces. For inflow boundaries, we use the POD modes to recover the velocity field and the turbulent variables, whereas we assign the pressure

gradient normal to the boundary. For outflows, instead, we prescribe Dirichlet conditions for the pressure field and Neumann conditions for velocity and turbulent quantities.

A similar mechanism is already available in OpenFOAM® , implemented in two classes of derived boundary conditions, namely `inletOutlet` and `outletInlet`. Such classes constitute a wrapper around mixed conditions, i.e. Robin conditions, performing a linear blend between fixed value and gradient condition through the definition of a certain value fraction w:

$$y_f = w y_b + (1 - w)\left(y_c + |\mathbf{d}_n| g_b\right) \ ,$$

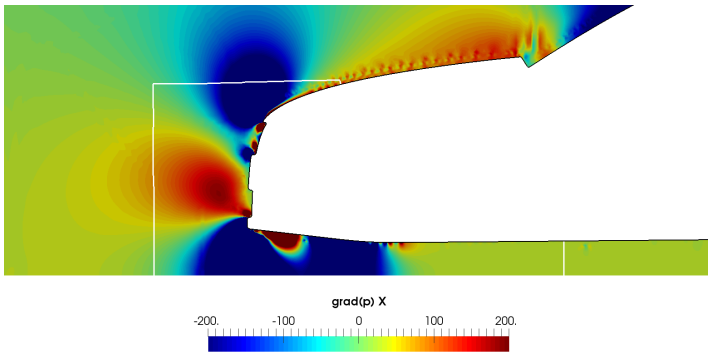
where y_f and y_c represent the face and first cell values respectively, $|\mathbf{d}_n|$ is the face-to-cell distance and y_b and g_b are the prescribed reference values at the boundary. The standard versions of `inletOutlet` and `outletInlet` boundary types, compute locally the value fraction according to the flow flux but enforce a zero gradient condition, that it is not physically accurate for internal interfaces as Γ_1, where there is no evidence that the gradients will be small (see for instance Figure 5.2, where we report the components of ∇p for a case of interest).

For this reason, we implement two new classes of boundary conditions, called `inletOutletFxGrad` and `outletInletFxGrad`, in which the normal gradients on Γ_1 are recovered from the gradients of the solution on Ω_2. Since the expansion coefficients in the POD projection operator (Equation (5.11)) do not depend on the spatial coordinates, the gradient of the $\mathbf{y}_2$ field can be simply expressed as:

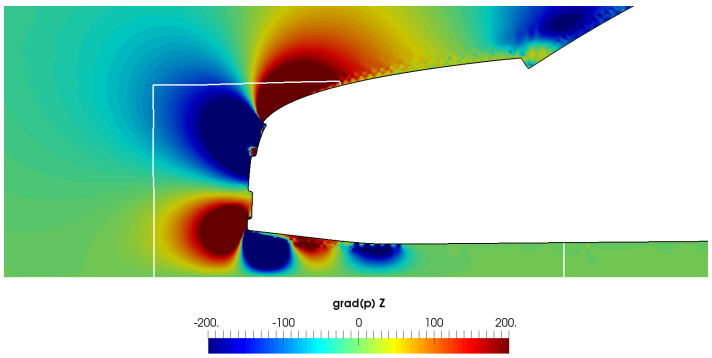
$$\nabla \mathbf{y}_2 = \nabla \bar{\mathbf{y}} + \sum_{i=1}^{M_r} \alpha_i \nabla \boldsymbol{\psi}_i \ ,$$

where the gradients of the average field and POD modes can be conveniently pre-computed and stored at the beginning of the iterative algorithm. Anyway, since we are interested only in the normal gradient on the faces belonging to the interface Γ_1, we need only to compute $\mathbf{y}_2$ on the boundary faces and on the first internal cell and then we can evaluate the reference gradient as $(\mathbf{y}_{2f} - \mathbf{y}_{2c})/|\mathbf{d}_n|$.

In Figure 5.3 we report a comparison between two full-order simulations on the $\Omega_1(\boldsymbol{\mu})$ subdomain for the 2DCAR benchmark, evaluated imposing the different classes of boundary conditions described above. The errors are defined as $|p - p^\star|_{\Omega_1(\boldsymbol{\mu})}/p_{\text{dyn}}$, where p_{dyn} denotes the reference dynamic pressure, defined as $\frac{1}{2}\rho U_{\text{ref}}^2$. In order to discern the error component due only to the numerical implementation of the boundary conditions,

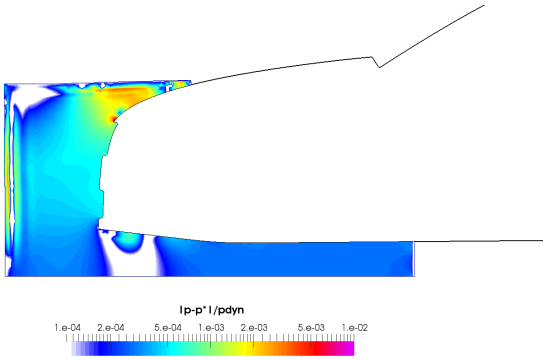


(a) Pressure gradient, x-component.

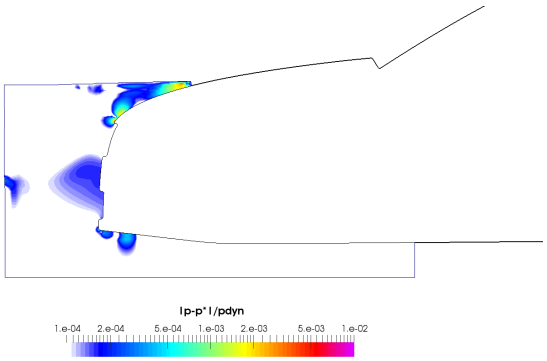


(b) Pressure gradient, z-component.

Fig. 5.2 Detail of the pressure gradient in proximity of the Γ_1 interface (white line) for a FOM snapshot of the 2DCAR benchmark.



(a) Mixed boundary conditions with zero gradients.



(b) Mixed boundary conditions with non-zero gradients.

Fig. 5.3 Error on the pressure field due to the different types of boundary conditions for the 2DCAR benchmark.

the zonal-POD iterative method is disabled in this context: for both simulations, the boundary values are interpolated from the corresponding FOM solution over $\Omega(\boldsymbol{\mu})$ and kept fixed during the iterations of the SIMPLE algorithm. With respect to Figure 5.3a, showing the error obtained with null normal gradients, the solution is more accurate when we recover not only the variables values but also the gradients, as depicted in Figure 5.3b.

5.3.3 Preliminary numerical results

In order to provide some insight about the performance of the zonal-POD method for high Re turbulent flows, we now present some numerical results on the 2DCAR benchmark introduced in Section 4.1.

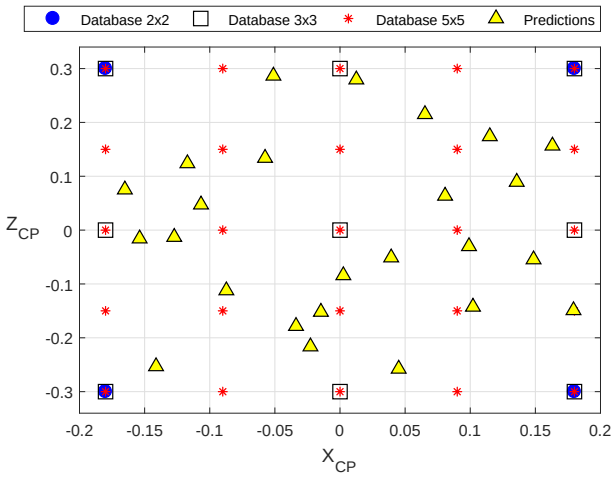


Fig. 5.4 2DCAR benchmark parameter space: database sampling points and prediction points.

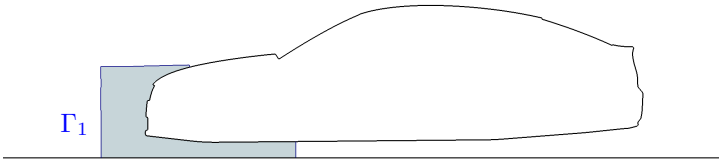


Fig. 5.5 Domain decomposition of the 2DCAR benchmark: $\Omega_1(\boldsymbol{\mu})$ (blue) coincides with Ω_0.

The ability of the hybrid approach to perform predictive simulations is investigated by testing 25 new configurations not included in the database. Such prediction points are generated through a pseudo-random Sobol sequence and their distribution in the parameter space is shown in Figure 5.4, along with the points belonging to the database of snapshots. At this stage, we consider a uniform sampling of the parameter space to generate the snapshots, and in particular three different sets of increasing size are tested, containing $N = 4$, $N = 9$ and $N = 25$ points (see Figure 5.4). Clearly, a uniform distribution is hardly an optimal choice for the parameter space exploration, and more efficient sampling strategies needs to be introduced: this aspect will be later addressed in Chapter 6. In addition, we assume that the domain decomposition is chosen as in Figure 5.5. Since the 2DCAR benchmark parameterization exploits a mesh morphing strategy, it is possible to set $\Omega_0 = \Omega_1(\boldsymbol{\mu})$. The preservation of the mesh topology, in fact, permits to maintain a one-to-one correspondence between the grid elements of the different configurations: this allows us to define the POD basis with respect to the cells ordering, rather than their physical coordinates, without affecting the effectiveness of the approach.

We aim at recovering the full solution in $\Omega(\boldsymbol{\mu})$, by setting $\Omega_2 = \Omega(\boldsymbol{\mu})$ and evaluating the POD on the whole domain. The predicted drag and lift coefficients for the whole car profile are then computed by summing the FOM contribution in $\Omega_1(\boldsymbol{\mu})$ and the one given by the POD reconstruction on $\Omega_2 \backslash \Omega_1(\boldsymbol{\mu})$:

$$C_{\mathrm{x}}^{\star} = C_{\mathrm{x},1}^{\star} + C_{\mathrm{x},2}^{\star} \ ,$$
$$C_{\mathrm{z}}^{\star} = C_{\mathrm{z},1}^{\star} + C_{\mathrm{z},2}^{\star} \ .$$

The terms $C_{\mathrm{x},1}^{\star}$ and $C_{\mathrm{x},2}^{\star}$ are thus obtained through the integrals $\int_{\partial\Omega_1 \backslash \Gamma_1} (-p_1^{\star}\mathbf{n} + \boldsymbol{\tau}_1^{\star}) \cdot \mathbf{e}_x \partial S$ and $\int_{\partial\Omega_2 \backslash \partial\Omega_1} (-p_2^{\star}\mathbf{n} + \boldsymbol{\tau}_2^{\star}) \cdot \mathbf{e}_x \partial S$, respectively, where $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ represent the shear-stresses given by the solution approximation in the corresponding region. $C_{\mathrm{z},1}^{\star}$ and $C_{\mathrm{z},2}^{\star}$ are derived with analogous considerations. The relative importance of the contributions is shown in Table 5.1, for the predictive simulation with $\boldsymbol{\mu}_{\mathrm{test}} = (0.13568, 0.09835)$; the flow solution in $\Omega_1(\boldsymbol{\mu})$ accounts for approximately 31% of C_{x} and 69% of C_{z}. Both regions are then equally relevant for the determination of the aerodynamic performance of the profile. In Table 5.1, we also illustrate the efficacy of the zonal-POD approach. For this purpose, we perform different hybrid simulations with the given decomposition, by varying the size of the database ($N = 4$ and $N = 9$), and without truncating the corresponding POD bases (i.e. $M_r^{\psi} = M_r^{\varphi} = M_r = N - 1$). As expected, the finer database leads to a better approximation of the solution. To evaluate the importance of the boundary conditions updating mechanism, an additional simulation with $M_r = 0$ is performed, whose boundary conditions are derived from the average of $N = 9$ database snapshots. In this case, the contribution of the POD modes is set to zero, and we use only the forcing term of the POD projection operator defined in (5.11), which becomes $\Pi^{\mathrm{POD}}(\mathbf{y}) = \bar{\mathbf{y}}$. As shown in Table 5.1, with the introduction of the POD modes, the relative error on the drag coefficient drops from 8.4% to 0.5%, and from 15.8% to 0.3% for the downforce. This trend is not an artefact of the integration, but can be observed also when we consider spatial-distributed variables, as in the case of Figure 5.6, where we report the distribution of the pressure coefficient, C_p, on the vehicle underbody in both $\Omega_1(\boldsymbol{\mu})$ (solid lines) and $\Omega_2 \backslash \Omega_1(\boldsymbol{\mu})$ (dotted lines). The discrepancy with respect to the FOM is remarkable using $N = 9$ and $M_r = 0$, whereas it becomes imperceptible for $N = 9$ and $M_r = 8$.

The same analysis is carried out for the remaining prediction points. The results are shown in Figure 5.7, where we report the relative error on the total drag coefficient, evaluated as $|C_{\mathrm{x}}(\boldsymbol{\mu}) - C_{\mathrm{x}}^{\star}(\boldsymbol{\mu})|/|C_{\mathrm{x}}(\boldsymbol{\mu})|$. The hybrid approach with $N = 9$ and $M_r = 8$ shows a good behaviour on the parameter space, with an average error over the predictions of 0.93% (maximum error 3.01%), compared to the 7.00% (maximum error 14.83%) with

	C_{x}	$C_{\mathrm{x},1}$	C_{z}	$C_{\mathrm{z},1}$
FOM	0.015075	0.004716	-0.143092	-0.099199
	$C^{\star}_{\mathrm{x}}$	$C^{\star}_{\mathrm{x},1}$	$C^{\star}_{\mathrm{z}}$	$C^{\star}_{\mathrm{z},1}$
zonal-POD w $N = 9$, $M_r = 0$	0.016346	0.006969	-0.120436	-0.096443
zonal-POD w $N = 4$, $M_r = 3$	0.015271	0.004795	-0.146132	-0.100173
zonal-POD w $N = 9$, $M_r = 8$	0.015153	0.004791	-0.142726	-0.098935

Table 5.1 2DCAR benchmark: contribution of the flow solution in $\Omega_1(\boldsymbol{\mu})$ to the global aerodynamic coefficients for a predictive simulation, using the FOM and different zonal-POD approaches.

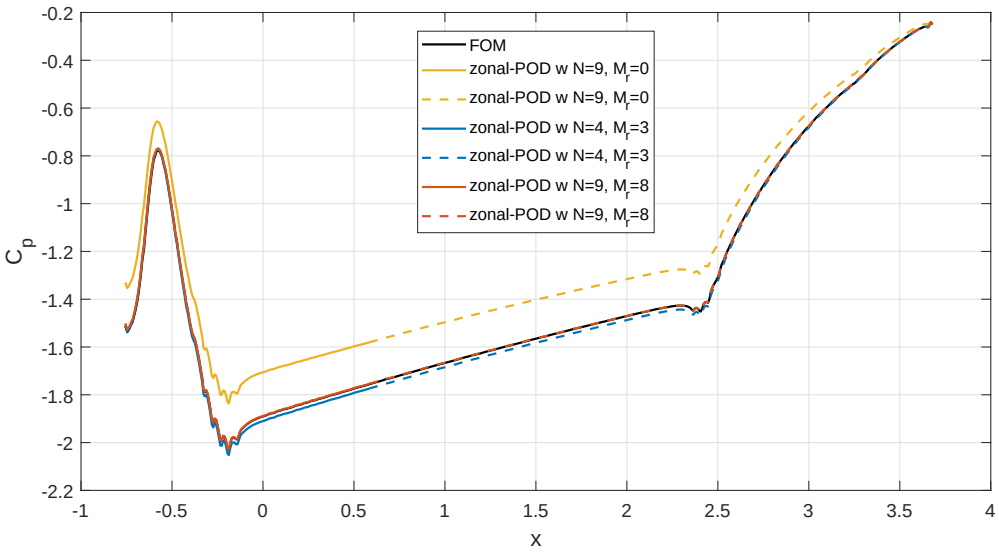


Fig. 5.6 2DCAR benchmark: underbody C_p distribution varying N and M_r.

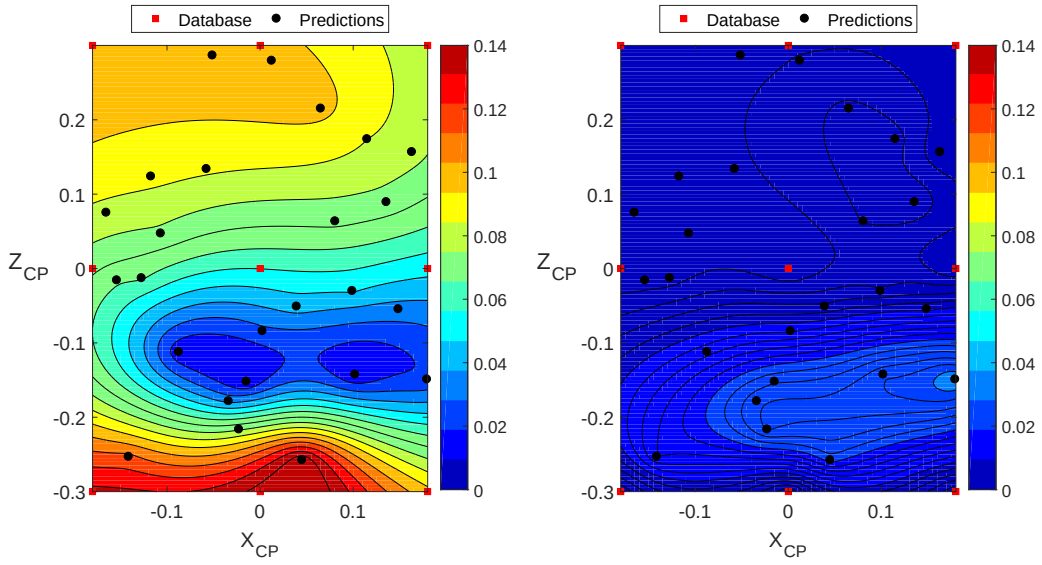


Fig. 5.7 2DCAR relative error on drag coefficient: predictive zonal simulations with $M_r = 0$ (left) and $M_r = 8$ (right), using $N = 9$ snapshots.

no POD reconstruction of the boundary conditions ($M_r = 0$). The convergence of the average prediction error, varying the size of the database, is reported in Table 5.2.

Database	$\bar{e}_{C_x}$
$N = 4$	0.01126
$N = 9$	0.00934
$N = 25$	0.00184

Table 5.2 Convergence of the average error $\bar{e}_{C_x}$ varying the size of the database.

It is also interesting to observe how the error is distributed between the two contributions. As noticeable in Figure 5.8, since we employ the full-order model in $\Omega_1(\boldsymbol{\mu})$, albeit with approximate boundary conditions on Γ_1, the error contribution in this subregion is much smaller than the one outside. This means that the method tends to become particularly accurate when the output of interest is a quantity localized in such domain, as happens for instance when the optimization is focused on a single component of a multi-body configuration.

Finally, in Figure 5.9 we present a comparison of the accuracy between the PODI and zonal-POD approaches, using the same POD basis built over the entire domain. Our method performs significantly better when the database is coarse, whereas the benefits of the hybrid approach tends to vanish when N increases.

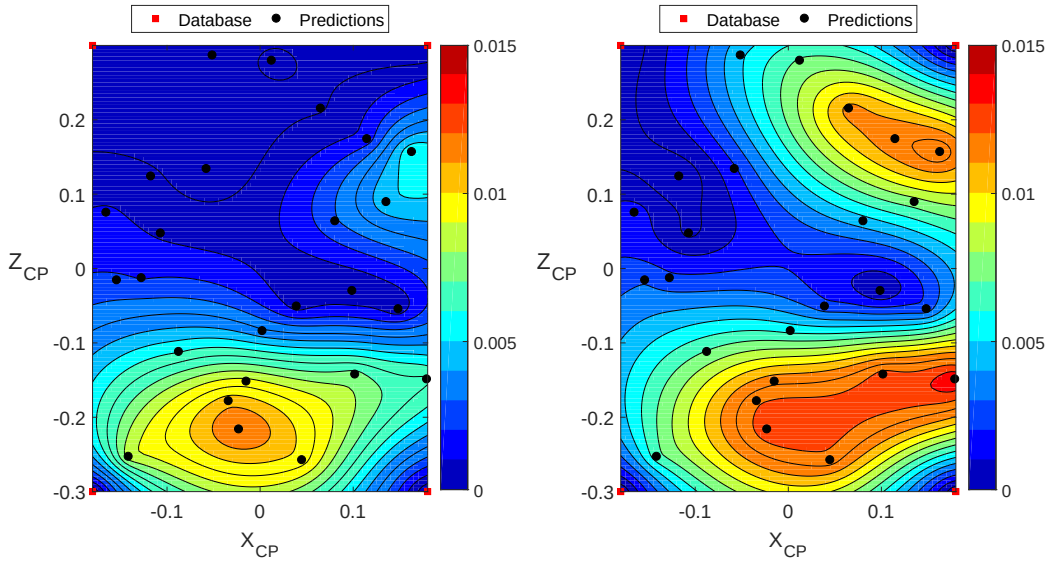
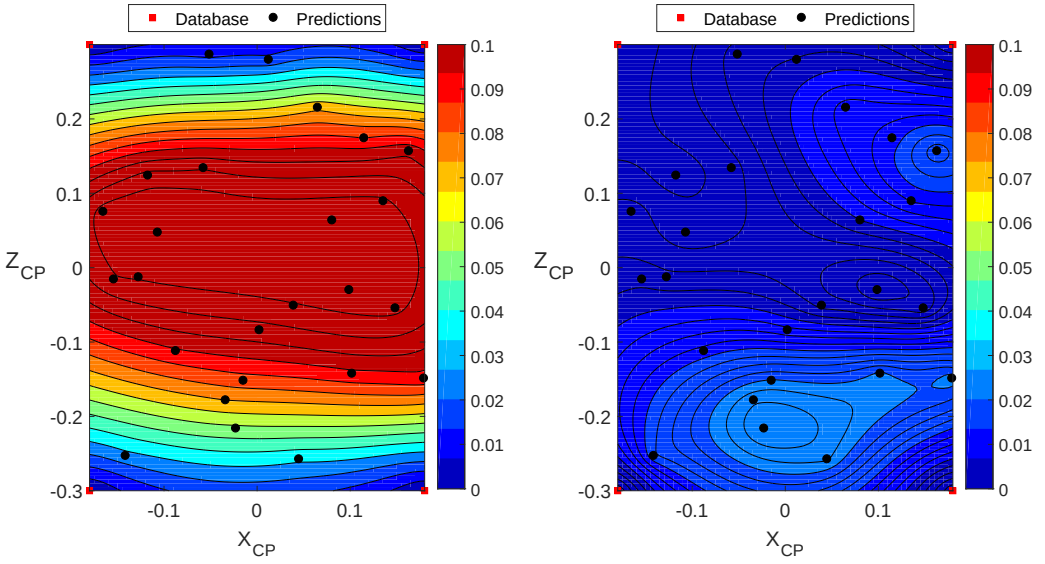


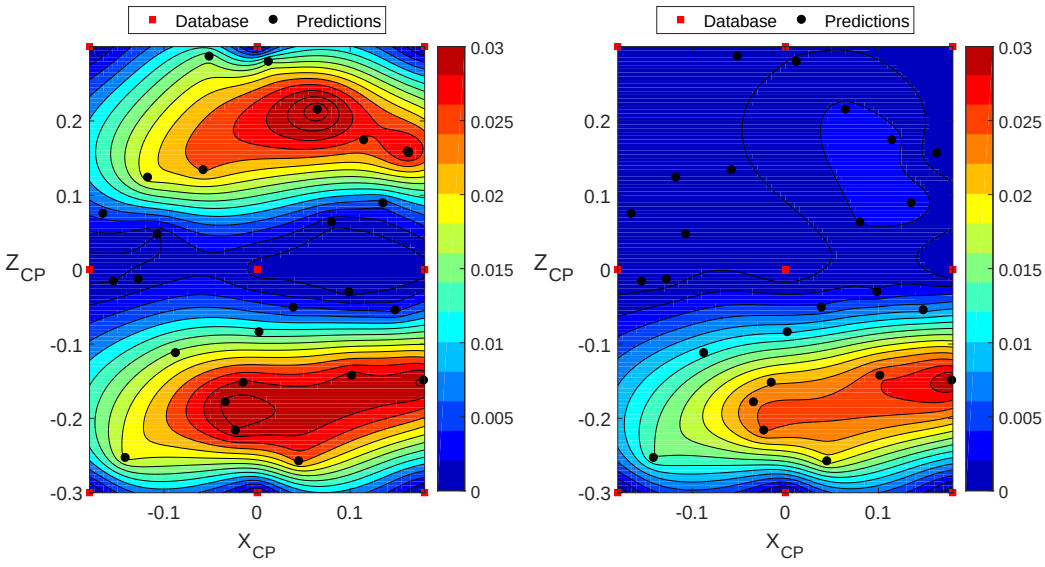
Fig. 5.8 2DCAR error contributions for predictive zonal simulations with $M_r = 3$ using $N = 4$ snapshots: $|C^{\star}_{\mathrm{x},1} - C_{\mathrm{x},1}|/C_{\mathrm{x}}$ (left) and $|C^{\star}_{\mathrm{x},2} - C_{\mathrm{x},2}|/C_{\mathrm{x}}$ (right).

In terms of computational performance, the achievable speed-up is highly application-dependent and may vary a lot from case to case, ranging from $O(10)$ up to $O(100)$. Generally speaking, two main factors contribute to its performance. In the first place, in the zonal-POD approach the FOM is solved on a domain that is a subset of the original one and contains fewer cells; secondly, the convergence of the FOM in Ω_1 is usually faster thanks to a better flow initialization.

As for the geometrical contribution, the critical regions requiring the full-order approximation are usually located near the body (see Chapter 6) where the computational grid tends to be very fine, in order to better represent geometrical details and properly describe the near-wall field. Consequently, for unstructured meshes, even if Ω_1 dimensions are small in the physical space, such subdomain may contain almost as many cells as Ω, with a resulting geometrical speed-up factor close to 1. For automotive and naval applications, representing the main target of this work, this is unlikely to happen thanks to the usage of wall-functions, that permits to reduce the near-wall grid resolution, leading to typical speed-up factors up to $O(10) - O(20)$, depending on the deformation type, as it will be addressed later in Chapter 8. On the other hand, this may be an issue for highly accurate aeronautical applications, requiring for instance a very refined grid in order to fully resolve the boundary layer (i.e. $y^+ < 1$), with deformations that greatly perturb the flow field. Anyway, when the effects of geometry variation are more localized, the performance of the method is expected to improve also in these cases, since Ω_1 will



(a) $N = 4, M_r = 3$.



(b) $N = 9, M_r = 8$.

Fig. 5.9 2DCAR relative error on drag coefficient: PODI with RBF interpolation (left) vs zonal-POD (right).

include only a portion of the geometrical body. For Cartesian grids, instead, the speed-up factors are typically of order $O(50) - O(100)$, as shown for instance in Bergmann et al. (2018).

In terms of flow initialization, the hybrid computations can be initialized using a combination of POD modes, leading to convergence in fewer iterations with respect to runs initialized with free-stream and potential conditions, or other flow solutions. The same rationale, however, can also be applied to initialize the full-order simulations, as shown for instance in Mifsud et al. (2015), where different ROMs over the full domain are used in order to obtain the initial fields for the FOM, accelerating the convergence of the model. A comparison among the zonal-POD approach and the FOM starting from different initial fields is reported in Figure 5.10, for a configuration not included in the snapshots database. We test the FOM convergence performance using two different initialization methods: the standard potential solution, and an approximate solution obtained through POD with RBF interpolation. In agreement with Mifsud et al. (2015), initializing the full-order simulation with the ROM allows us to halve the number of iterations required to converge. Even so, the hybrid method, initialized with the POD forcing term $\bar{\mathbf{y}}$, shows a better performance of all the FOM considered, leading to a speed-up factor of about 4, as can be observed for instance in the trend of both the residual (Figure 5.10 left) and the drag coefficient (Figure 5.10 right). For the zonal-POD simulations, it should be noted that both quantities are computed within Ω_1 and so all the C_x values are normalized with the respective converged value, in order to ease the comparison. Increasing the size of the database, instead, produces only a slight acceleration for both the FOM with PODI initialization and the zonal-POD approach.

5.3.4 Convergence analysis

The Schwarz method and its variations are robustly convergent for a wide class of equations (see for instance Quarteroni and Valli (1999)), with a convergence speed depending on the overlap between the subdomains. Since the Schwarz-POD algorithm deviates from the classical method, a further investigation of its convergence properties is necessary. For the sake of simplicity, let us consider a simplified linear problem, where we focus on a single domain $\Omega = \Omega(\hat{\boldsymbol{\mu}})$ in the given family, so that the $\boldsymbol{\mu}$ dependence is no longer explicit and the decomposition simplifies as $\Omega = \Omega_1 \cup \Omega_2$, $\Omega_0 := \Omega_1 \cap \Omega_2$. The boundary $\partial\Omega$ of the domain is assumed to be sufficiently smooth, e.g. Lipschitz continuous.

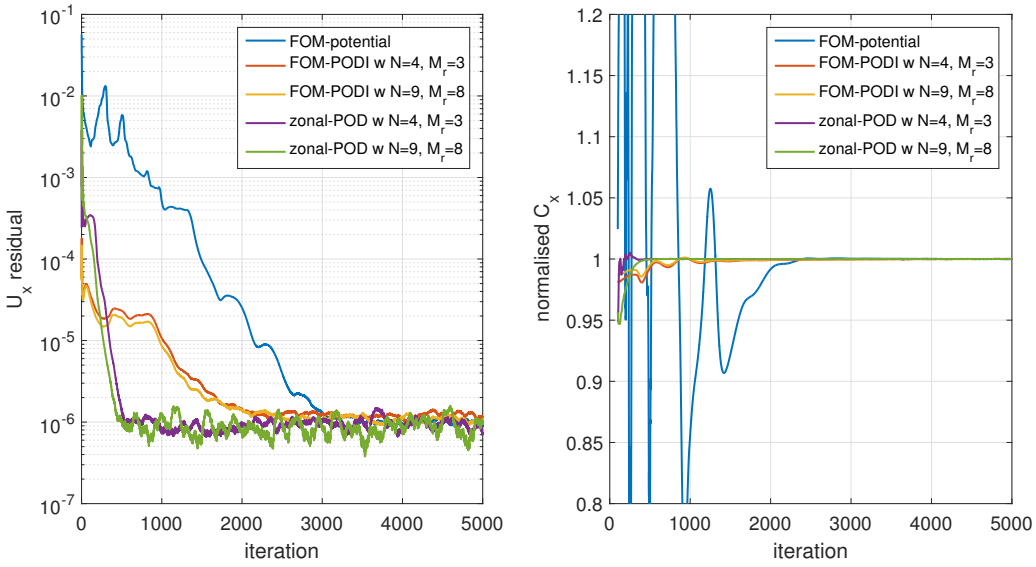


Fig. 5.10 Convergence performance of the zonal-POD approach w.r.t. full-order simulations initialized with different methods for an out-of-sample configuration: u_x residuals (left) and normalised drag coefficient C_x (right).

The NS problem, is then replaced by the following scalar Dirichlet problem:

$$\begin{aligned} Lu &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega, \end{aligned} \tag{5.16}$$

where the convection-diffusion operator $Lu := \nabla\cdot(\nu\nabla u + \mathbf{a}u)$, with f, g, $\nu > 0$ sufficiently smooth and $\mathbf{a}$ satisfying $\nabla\cdot\mathbf{a} = 0$ in Ω, in order to assure that the problem is well posed in $H^1(\Omega)$.

The solutions snapshots $u_i,\ i = 1, \ldots, N$, used to train the POD model, are generated by solving Problem (5.16) for different values of the parameter $\boldsymbol{\mu}$. Thus, the snapshots are supposed to solve the equation $Lu_i = f$ in Ω_2, common among them all, as well as the boundary condition $u_i = g$ on $\partial\Omega_2 \cap \partial\Omega$. Given the linearity of the problem, also their average $\bar{u}$ satisfies $L\bar{u} = f$ in $\Omega_2 \cup \Gamma_2$, $\bar{u} = g$ on $\partial\Omega_2 \cap \partial\Omega$, whereas the natural extensions of each POD basis function to Ω_2, denoted for simplicity with $\varphi_i,\ i = 1, \ldots, M_r$, satisfies $L\varphi_i = 0$ in $\Omega_2 \cup \Gamma_2$, $\varphi_i = g$ on $\partial\Omega_2 \cap \partial\Omega$. We also assume that $\langle\varphi_i, \varphi_j\rangle_{\Omega_0} = \delta_{ij}$ for $1 \le i, j \le M_r$, i.e. the modes are orthonormal in the overlapping region. As seen in Section 5.3, the snapshots generate the following spaces

$$S^0 := \mathrm{span}\{\varphi_i \,:\, i = 1, \ldots, M_r\}, \qquad S := \bar{u} + S^0,$$

where the terms φ_i represent the POD basis functions. The corresponding spaces of traces on Γ_1 are then

$$S^0_{\Gamma_1} := \{v_{|\Gamma_1} \,:\, v \in S^0\}, \qquad S_{\Gamma_1} := \{v_{|\Gamma_1} \,:\, v \in S\} = \bar{u}_{|\Gamma_1} + S^0_{\Gamma_1}.$$

For the problem at hand, the Schwarz-POD algorithm results in the concatenation of the following stages:

Stage 1. Given $u_2 \in S$, set $\lambda := u_{2\,|\Gamma_1} \in S_{\Gamma_1}$ and find $u_1 \in H^1(\Omega_1)$ as a solution of

$$\begin{aligned} Lu_1 &= f \qquad \text{in } \ \Omega_1, \\ u_1 &= g \qquad \text{on } \ \partial\Omega_1 \cap \partial\Omega, \\ u_1 &= \lambda \qquad \text{on } \ \Gamma_1. \end{aligned} \tag{5.17}$$

Stage 2. Given $u_1 \in H^1(\Omega_1)$, compute the POD projection of its restriction to the overlapping region $\tilde{u}_1 := u_{1\,|\Omega_0}$

$$\tilde{u}_2 := \Pi^{\text{POD}}(\tilde{u}_1) \tag{5.18}$$

and get its natural extension to the whole of Ω_2, i.e. $u_2 \in S$.

The sequence can be conveniently seen in terms of the action of an operator on the trace space $S_{|\Gamma_1}$. In fact, if we introduce the mapping $T : S_{|\Gamma_1} \to S_{|\Gamma_1}$ such that

$$T(\lambda) = \mu,$$

with $\mu := \tilde{u}_{2\,|\Gamma_1}$ depending on λ through $\tilde{u}_1$, the Schwarz-POD algorithm can be rewritten as

$$\lambda_{k+1} = T(\lambda_k), \qquad k \geq 1.$$

If the sequence $\{(u_1, u_2)^{(k)}\}$ converges to a pair $(u_1^*, u_2^*) \in H^1(\Omega_1) \times S$, then we have that the trace of the solution on Γ_1, namely $\lambda^\star$, is a fixed point of T, i.e.

$$\lambda^* = T(\lambda^*).$$

Thus, if T is proven to be a contraction with respect to a certain norm in $S_{|\Gamma_1}$, the algorithm converges.

For the sake of clarity, let us express the solution u_1 of (5.17) as a sum of two contributions, i.e. $u_1 = v + w$, so that we have

$$\begin{aligned} Lv &= f \quad && \text{in } \Omega_1, \\ v &= g \quad && \text{on } \partial\Omega_1 \cap \partial\Omega, \\ v &= \bar{\lambda} \quad && \text{on } \Gamma_1, \end{aligned} \tag{5.19}$$

and

$$\begin{aligned} Lw &= 0 \quad && \text{in } \Omega_1, \\ w &= 0 \quad && \text{on } \partial\Omega_1 \cap \partial\Omega, \\ w &= \lambda^0 \quad && \text{on } \Gamma_1. \end{aligned} \tag{5.20}$$

where $\bar{\lambda}$ denotes the restriction $\bar{u}_{|\Gamma_1}$, so that any $\lambda \in S_{|\Gamma_1}$ can be split as $\lambda = \bar{\lambda} + \lambda^0$ for some $\lambda^0 \in S^0_{|\Gamma_1}$. Thus, the POD projection of $\tilde{u}_1$ can be made explicit in the following way:

$$\begin{aligned} \tilde{u}_2 = \Pi^{\mathrm{POD}}(\tilde{u}_1) &= \bar{u} + \sum_{i=1}^{M_r} \langle \tilde{u}_1 - \bar{u}, \varphi_i \rangle_{\Omega_0} \, \varphi_i \\ &= \underbrace{\bar{u} + \sum_{i=1}^{M_r} \langle \tilde{v} - \bar{u}, \varphi_i \rangle_{\Omega_0} \, \varphi_i}_{\tilde{\chi}} + \underbrace{\sum_{i=1}^{M_r} \langle \tilde{w}, \varphi_i \rangle_{\Omega_0} \, \varphi_i}_{\tilde{z}} \end{aligned}$$

where $\tilde{z} = \Pi^{\mathrm{POD},0}(\tilde{w})$ represents the L^2-orthogonal projection of $\tilde{w}$ onto S^0 and $\tilde{\chi}$ is the contribution of the average. Through this distinction, we have that

$$T(\lambda) = \mu = \tilde{\chi}_{|\Gamma_1} + \tilde{z}_{|\Gamma_1} =: \tilde{\chi}_{|\Gamma_1} + T^0 \lambda^0,$$

where $T^0 : S^0 \to S^0$ is the linear operator resulting from the composition of the mappings $\lambda^0 \mapsto w \mapsto \tilde{z} = \Pi^{\mathrm{POD},0}\tilde{w} \mapsto \mu^0 := \tilde{z}_{|\Gamma_1}$. Since the term $\tilde{\chi}_{|\Gamma_1}$ is constant between consequential iterates, we have that $T(\lambda_1) - T(\lambda_2) = T^0(\lambda_1^0 - \lambda_2^0)$, and thus the following result is obtained.

Theorem 5.2. *For a suitable norm $\|\cdot\|$ in $S_{|\Gamma}$, one has*

$$\|T^0\|_{\mathcal{L}(S^0, S^0)} < 1. \tag{5.21}$$

Hence, the operator T is a contraction with respect to this norm, and the Schwarz-POD algorithm converges at a geometric rate towards a limit pair $(u_1^*, u_2^*) \in H^1(\Omega_1) \times S$, which

satisfies the conditions

$$u^*_{1\,|\Gamma_1} = u^*_{2\,|\Gamma_1}, \qquad \tilde{u}^*_2 = \Pi^{\mathrm{POD}}(\tilde{u}^*_1) \quad in \quad \Omega_0.$$

In view of providing a sufficient condition for Theorem 5.2, we recall the following two results, that hold for any norm $\|\cdot\|$ in $S_{|\Gamma_1}$.

Lemma 5.1. *There exists a constant $C_1 > 0$ such that*

$$\|w\|_{L^2(\Omega_0)} \leq C_1 \|\lambda^0\|, \qquad \forall \lambda^0 \in S^0_{|\Gamma_1},$$

where w is the solution of (5.20).

Proof. From the theory of double-layer potentials (see for instance Jerison and Kenig (1995)), we know that w satisfies $\|w\|_{H^{1/2}(\Omega_1)} \leq c\|\lambda^0\|_{L^2(\Gamma_1)}$ since Ω_1 is assumed to be a Lipschitz domain. Since λ^0 belongs to the finite dimensional space $S^0_{|\Gamma_1}$, where all norms are equivalent, the following relation holds

$$\|w\|_{L^2(\Omega_0)} \leq \|w\|_{L^2(\Omega_1)} \leq \|w\|_{H^{1/2}(\Omega_1)} \leq c\|\lambda^0\|_{L^2(\Gamma_1)}.$$

□

Lemma 5.2. *There exists a constant $C_2 > 0$ such that*

$$\|z_{|\Gamma_1}\| \leq C_2 \|z\|_{L^2(\Omega_0)}, \qquad \forall z \in S^0.$$

Proof. Given that any $z \in S^0$ satisfies $Lz = 0$ in Ω_2, we also know that $Lz = 0$ in Ω_0. For a sufficiently smooth test function v, through integration by part we obtain the following relation:

$$0 = \int_{\Omega_0} Lz\, v = \int_{\Omega_0} z\, L^\top v + \int_{\partial\Omega_0} \left(-\frac{\partial z}{\partial n} + \mathbf{a}\cdot\mathbf{n}\, z \right) v + \int_{\partial\Omega_0} z \frac{\partial v}{\partial n}. \tag{5.22}$$

For any $\xi \in H^{1/2}(\partial\Omega_0)$, we consider the solution $v = v_\xi \in H^2(\Omega_0)$ of the fourth-order problem $LL^\top v = 0$ in Ω_0, $v = 0$ and $\frac{\partial v}{\partial n} = \xi$ on $\partial\Omega_0$, for which we have $\|v_\xi\|_{H^2(\Omega_0)} \leq c\|\xi\|_{H^{1/2}(\partial\Omega_0)}$. If we take supremum over ξ in the identity

$$\int_{\partial\Omega_0} z\, v_\xi = -\int_{\Omega_0} z\, L^\top v_\xi$$

we obtain that $\|z_{|\partial\Omega_0}\|_{H^{-1/2}(\partial\Omega_0)} \leq c\|z\|_{L^2(\Omega_0)}$. As in 5.1, the result easily follows by observing that $z_{|\partial\Omega_0}$ ranges in the finite dimensional space S^0 and by invoking again the norms equivalence in finite dimension. □

It is worth noting that Lemma 5.1 and 5.2 only guarantee the existence of the C_1, C_2 constants, whose value and dependence on the dimension of the $S^0_{|\Gamma_1}$ space are related with the choice of an appropriate norm.

Finally, we have the following result.

Theorem 5.3. *Given a suitable norm in S^0, let C_1 and C_2 be constants for which the inequalities in Lemma 5.1 and 5.2 are valid. Then, the operator T^0 satisfies $\|T^0\|_{\mathcal{L}(S^0,S^0)} \leq C_1C_2$. Hence,*

$$C_1C_2 < 1$$

is a sufficient condition for (5.21) to hold.

Proof. We consider an arbitrary $\lambda^0 \in S^0_{|\Gamma_1}$, and let $\mu^0 = T^0\lambda^0$. Using Lemma 5.1 and 5.2 and the property of the L^2-orthogonal projection, we have

$$\|\mu^0\| = \|\tilde{z}_{|\Gamma_1}\| \leq C_2\|\tilde{z}\|_{L^2(\Omega_0)} \leq C_2\|\tilde{w}\|_{L^2(\Omega_0)} \leq C_2C_1\|\lambda^0\|. \tag{5.23}$$

□

In the following section we prove the convergence for a simplified problem, i.e. a linear operator over a Cartesian decomposition, whereas for the NS/RANS problem we limit ourselves to simply verify the convergence condition by explicitly evaluating the C_2C_1 constant through the computation of the $\langle\tilde{w}, \varphi_i\rangle_{\Omega_0}$ terms for the 2DCAR benchmark with $N = 4$. The resulting constant is equal to 0.56917, showing compliance with the condition in Theorem 5.3.

5.3.5 Cartesian domains

We now consider the Cartesian domain $\Omega = (O, A) \times (O, B) \subset \mathbb{R}^2$ of Figure 5.11, that we decompose on the subdomains $\Omega_1 = (O, C) \times (O, B)$ and $\Omega_2 = (D, A) \times (O, B)$. After applying a change of independent variables, we can assume that $\Omega = (0, s) \times (0, \pi)$, and, coherently, $\Omega_2 = (q, s) \times (0, \pi)$ and $\Omega_2 = (q, s) \times (0, \pi)$ for $0 < q < r < s$. As a consequence, $\Omega_0 = (q, r) \times (0, \pi)$ and $\Gamma_1 = \{r\} \times [0, \pi]$.

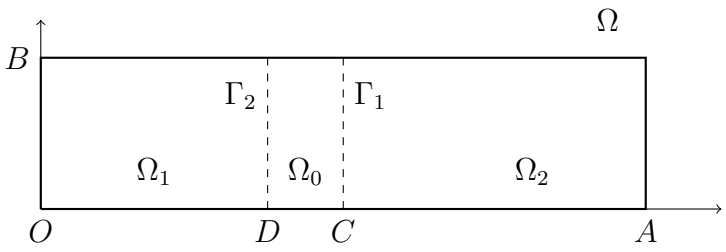


Fig. 5.11 Decomposition of the rectangular domain Ω in two overlapping subdomains.

In order to solve Problem (5.20) by separation of variables, we assume that the L operator has constant coefficients in Ω: after dividing by the diffusion coefficient ν, we obtain that $Lu = -\Delta u + au_x + bu_y$ for a suitable pair of real numbers a, b. At this point, if we look for a solution of the problem in the form $w(x, y) = \eta(x)\Theta(y)$, imposing $Lw = 0$ yields

$$(-\eta_{xx} + a\eta_x)\Theta + \eta(-\Theta_{yy} + b\Theta_y) = 0.$$

Let Θ be a solution of the eigenvalue problem

$$-\Theta_{yy} + b\Theta_y = \lambda\Theta \qquad \text{in } (0, \pi), \qquad \Theta(0) = \Theta(\pi) = 0.$$

If we substitute $\Theta(y) = \mathrm{e}^{by/2}\vartheta(y)$, it is easy to check that ϑ is a solution of the eigenvalue problem

$$-\vartheta_{yy} = (\lambda - \tfrac{b^2}{4})\vartheta \qquad \text{in } (0, \pi), \qquad \vartheta(0) = \vartheta(\pi) = 0,$$

whose solutions are given by

$$\vartheta_k(y) = \sqrt{\tfrac{2}{\pi}} \sin ky, \qquad \text{with } \lambda_k = k^2 + \tfrac{b^2}{4}, \qquad k \geq 1. \tag{5.24}$$

By denoting with $\eta_k(x)$ the solution of the k^{th} boundary-value problem

$$-\eta_{xx} + a\eta_x + \lambda_k\eta = 0 \quad \text{in } (0, r), \qquad \eta(0) = 0, \quad \eta(r) = 1, \tag{5.25}$$

we can express the solution of Problem (5.20) as

$$w(x, y) = \sum_{k\geq 1} \alpha_k \eta_k(x)\Theta_k(y) = \mathrm{e}^{by/2} \sum_{k\geq 1} \alpha_k \eta_k(x)\theta_k(y),$$

where the boundary condition on Γ_1 is enforced by requiring that the α_k coefficient are the sine-Fourier coefficients of the function $h(y) := \mathrm{e}^{-by/2}\lambda^0(y)$, which is surely

square-integrable in $(0, \pi)$. Thus, the matching condition is:

$$w(r, y) = \mathrm{e}^{by/2} \sum_{k \geq 1} \alpha_k \theta_k(y) = \lambda^0(y), \qquad 0 < y < \pi \ ,$$

To asses the convergence of our method for the problem at hand, we need to bound the L^2-norm of w in Ω_0. In order to do so we need to recall the following result.

Lemma 5.3. *The functions η_k, solutions of the boundary-value problem defined in (5.25), are monotone with respect to the index k in the interval $[0, r]$, precisely*

$$\eta_{k+1}(x) \leq \eta_k(x) \qquad \forall x \in [0, r], \quad \forall k \geq 1.$$

Proof. Setting $\sigma_{1k} := \frac{a}{2} + p_k$ and $\sigma_{2k} := \frac{a}{2} - p_k$, with $p_k := \frac{\sqrt{a^2 + 4\lambda_k}}{2}$, allows us to rewrite the $|\eta_k$ functions as

$$\eta_k(x) = \frac{\mathrm{e}^{\sigma_{1k} x} - \mathrm{e}^{\sigma_{2k} x}}{\mathrm{e}^{\sigma_{1k} r} - \mathrm{e}^{\sigma_{2k} r}}.$$

The result follows if

$$\frac{\eta_k(x)}{\eta_{k+1}(x)} = \frac{\psi_k(x)}{\psi_k(r)} \leq 1$$

with

$$\psi_k(x) := \frac{\mathrm{e}^{\sigma_{1k} x} - \mathrm{e}^{\sigma_{2k} x}}{\mathrm{e}^{\sigma_{1,k+1} x} - \mathrm{e}^{\sigma_{2,k+1} x}} = \frac{\mathrm{e}^{p_k x} - \mathrm{e}^{-p_k x}}{\mathrm{e}^{p_{k+1} x} - \mathrm{e}^{-p_{k+1} x}}, \tag{5.26}$$

that it is certainly true if we prove that ψ_k does not increase with respect to $x \geq 0$, i.e. if its derivative

$$\psi_k'(x) = \frac{\gamma_k(\mathrm{e}^{\delta_k x} - \mathrm{e}^{-\delta_k x}) - \delta_k(\mathrm{e}^{\gamma_k x} - \mathrm{e}^{-\gamma_k x})}{(\mathrm{e}^{p_{k+1} x} - \mathrm{e}^{-p_{k+1} x})^2} < 0$$

with $\gamma_k := p_{k+1} + p_k > p_{k+1} - p_k =: \delta_k > 0$. By expanding the exponential functions of the fraction numerator $N_k(x)$ in Taylor series, we get that

$$N_k(x) = 2\delta_k \gamma_k \sum_{n=1}^{\infty} (\delta_k^{2n} - \gamma_k^{2n}) \frac{x^{2n+1}}{(2n+1)!} < 0 \qquad \text{for} \quad x > 0,$$

which proves the result. □

Applying Lemma 5.3 and exploiting the L^2-orthogonality of the eigenfunctions (5.24), we have that

$$\|w\|^2_{L^2(\Omega_0)} = \int_q^r \int_0^\pi \left(\sum_{k\geq 1} \alpha_k \eta_k(x) \theta_k(y) \right)^2 \mathrm{e}^{by} \, \mathrm{d}y \, \mathrm{d}x \tag{5.27}$$

$$\leq C_b \int_q^r \sum_{k\geq 1} (\alpha_k \eta_k(x))^2 \, \mathrm{d}x = C_b \sum_{k\geq 1} \alpha_k^2 \int_q^r \eta_k^2(x) \, \mathrm{d}x \tag{5.28}$$

$$\leq C_b \int_q^r \eta_1^2(x) \, \mathrm{d}x \sum_{k\geq 1} \alpha_k^2, \tag{5.29}$$

with the constant $C_b = \max_{y\in[0,\pi]} \mathrm{e}^{by}$. On the other hand,

$$\sum_{k\geq 1} \alpha_k^2 = \|h\|^2_{L^2(0,\pi)} = \int_0^\pi \mathrm{e}^{-by} (\lambda^0)^2(y) \, \mathrm{d}y \leq \frac{1}{c_b} \|\lambda^0\|^2_{L^2(\Gamma_1)}, \tag{5.30}$$

with $c_b = \min_{y\in[0,\pi]} \mathrm{e}^{by}$. By substituting (5.30) in (5.27), it follows that the solution of Problem (5.20) satisfies the relation

$$\|w\|^2_{L^2(\Omega_0)} \leq \frac{C_b}{c_b} \int_q^r \eta_1^2(x) \, \mathrm{d}x \; \|\lambda^0\|^2_{L^2(\Gamma_1)} \tag{5.31}$$

that provides an expression for the C_1 constant in Lemma 5.1, when the L^2-norm is considered.

Let us now consider any $z \in S^0$, that satisfies $Lz = 0$ in Ω_2 and $z = g$ on $\partial\Omega_2 \setminus \Gamma_2$, where $\Gamma_2 = \{q\} \times [0, \pi]$. As above, we can expressed it as

$$z(x, y) = \sum_{k\geq 1} \beta_k \xi_k(x) \Theta_k(y) = \mathrm{e}^{by/2} \sum_{k\geq 1} \beta_k \xi_k(x) \theta_k(y),$$

where ϑ_k is the same function defined in (5.24), whereas ξ_k represents the solution of the boundary-value problem

$$-\xi_{xx} + a\xi_x + \lambda_k \xi = 0 \quad \text{in} \quad (q, s), \qquad \xi(r) = 1, \quad \xi(s) = 0. \tag{5.32}$$

Therefore, the trace $z_{|\Gamma_1}$ on Γ_1 can be expressed as:

$$z(r, y) = \mathrm{e}^{by/2} \sum_{k\geq 1} \beta_k \theta_k(y), \qquad 0 < y < \pi. \tag{5.33}$$

As in Lemma 5.3, the following result holds.

Lemma 5.4. *The functions ξ_k, solutions of the boundary-value problem defined in (5.32), are monotone with respect to the index k in the interval $[q, r]$, precisely*

$$\xi_{k+1}(x) \geq \xi_k(x) \qquad \forall x \in [q, r], \quad \forall k \geq 1.$$

Proof. Let ψ_k be the same function introduced in the proof of Lemma 5.3 (see (5.26)). Then, it is easily verified that

$$\frac{\xi_k(x)}{\xi_{k+1}(x)} = \frac{\psi_k(s-x)}{\psi_k(s-r)} \leq 1, \qquad \text{if } \; q \leq x \leq r < s,$$

whence the result. □

As above, by applying 5.4, we can derive an expression for the C_2 constant in Lemma 5.2, when the L^2-norm is used on Γ_1. With the same definition of C_b and c_b, we obtain that

$$\|z\|^2_{L^2(\Omega_0)} \geq c_b \sum_{k\geq 1} \beta_k^2 \int_q^r \xi_k^2(x)\,\mathrm{d}x \geq c_b \int_q^r \xi_1^2(x)\,\mathrm{d}x \sum_{k\geq 1} \beta_k^2$$

and

$$\sum_{k\geq 1} \beta_k^2 = \int_0^\pi \mathrm{e}^{-by} z^2_{|\Gamma_1}(y)\,\mathrm{d}y \geq \frac{1}{C_b}\,\|z_{|\Gamma_1}\|^2_{L^2(\Gamma_1)},$$

Therefore, any $z \in S^0$ satisfies

$$\|z_{|\Gamma_1}\|^2_{L^2(\Gamma 1)} \leq \frac{C_b}{c_b} \frac{1}{\int_q^r \xi_1^2(x)\,\mathrm{d}x}\; \|z\|^2_{L^2(\Omega_0)}. \tag{5.34}$$

By combining Theorem 5.2 with relations (5.31) and (5.34), the following result is obtained.

Theorem 5.4. *With respect to the L^2-norm in S^0, it holds*

$$\|T^0\|_{\mathcal{L}(S^0,\,S^0)} \leq \left(\frac{C_b}{c_b}\right)^2 \frac{\int_q^r \eta_1^2(x)\,\mathrm{d}x}{\int_q^r \xi_1^2(x)\,\mathrm{d}x}\,.$$

Hence, if the ratio $\frac{C_b}{c_b} \geq 1$ is sufficiently small, $\|T^0\|_{\mathcal{L}(S^0,\,S^0)} < 1$ and the Schwarz-POD algorithm is convergent.

Proof. The result follows if we prove that

$$\frac{\int_q^r \eta_1^2(x)\,\mathrm{d}x}{\int_q^r \xi_1^2(x)\,\mathrm{d}x} < 1,$$

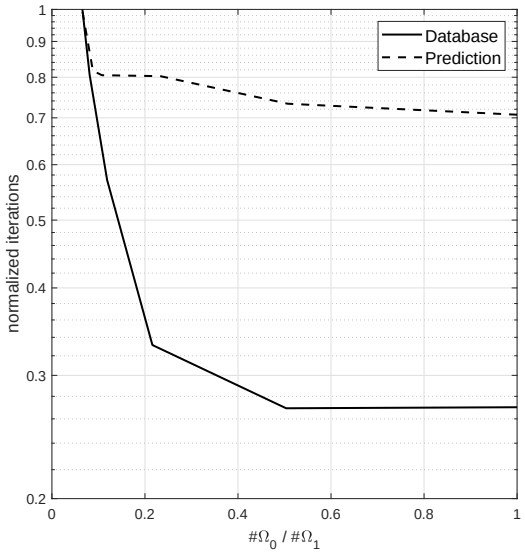


Fig. 5.12 2DCAR convergence velocity (iterations normalised w.r.t. to the maximum number of iterations required to converge) of the zonal-POD algorithm with respect to the number of cells in Ω_0 (normalised w.r.t the number of cells in Ω_1).

that is certainly true as a consequence of the inequalities $\eta_1(x) < \eta_1(r) = 1$ and $\xi_1(x) > \xi_1(r) = 1$, verified for all $x < r$. □

It is worth noting that the ratio $\frac{C_b}{c_b}$ is close to 1 when the absolute value $|b|$ of the convection coefficient in the y-direction is sufficiently small. Anyway, it is always possible to reduce the $\frac{\int_q^r \eta_1^2(x)\,\mathrm{d}x}{\int_q^r \xi_1^2(x)\,\mathrm{d}x}$ ratio by increasing the size of the integration interval $[q, r]$, which is equivalent to select a larger Ω_0. As in classical Schwarz methods, a more extended overlapping region will benefit the convergence performance of the algorithm, as shown for instance in Figure 5.12, where we report the convergence velocity with respect to the size of Ω_0 for the 2DCAR benchmark, both for an in-sample and out-of-sample configuration. The plots are generated by initializing the algorithm with the average fields and maintaining fixed the size of Ω_1. The convergence speed is expressed in terms of the number of iterations required for the SIMPLE algorithm to reach a steady solution in Ω_1, calling a POD-update of the boundary condition at every iteration. As expected, the zonal-POD algorithm converges faster with a larger overlapping region, whereas instabilities may arise when a very small Ω_0 is used ($O(100)$ cells): it appears that in these cases, the CFD feedback is no longer sufficient to guarantee a stable solution. Note that for configurations belonging to the snapshots database, in the limit of no truncation of the POD basis, the converged solution in Ω_1, $\mathbf{y}_1^\star$, is equal to the actual full-order

	POD-Galerkin	PODI	zonal-POD
$\text{cput}_{\text{FOM}}/\text{cput}_{\text{ROM}}$	$O(10^4 - 10^6)$	$O(10^6 - 10^7)$	$O(10 - 10^2)$
offline cost	$O(10^2) \times \text{cput}_{\text{FOM}}$	$O(10^2) \times \text{cput}_{\text{FOM}}$	$O(10) \times \text{cput}_{\text{FOM}}$
range of applicability	narrow	large	large
intrusiveness	high	very low	low
stability	needs stabilizers	-	-
robustness	low	low	medium

Table 5.3 Comparison of POD-based model reduction strategies for the applications of interest.

solution $\mathbf{y}_{|\Omega_1}$, whereas for predictive simulations the converged solution obtained through the zonal-POD approach will be just an approximation.

5.3.6 Conclusions

The performance of the POD-Galerkin, PODI and zonal-POD strategies with respect to the above-mentioned set of criteria is summarised in Table 5.3. Among the advantages of the zonal-POD method, it is worth citing the following ones:

- it is easy to implement straight forwardly in the canonical CFD solver, since it reduces to the imposition of a non-local boundary condition, and therefore its usage has limited impact on existing CFD codes;
- it does not rely on the choice of the numerical discretization, so it can be used coupled to different discretization methods, e.g. finite-element, finite-volume or immersed-boundary, as in Bergmann et al. (2018);
- it does not exploit flow specific properties, meaning that it can be employed in both laminar and turbulent conditions, for incompressible and compressible flows;
- it does not depend on the parameterization or on its properties, since the POD model is based on empirical observations and the parameterization does not intervene in the coupling strategy;
- it presents good stability and robustness properties, thanks to FOM feedback in the region of interest.

The main drawback of the approach, instead, is constituted by its moderate reduction with respect to standard techniques, making it less performing during the online phase. In

addition, the accuracy for out-of-sample configurations depends not only on the database used to train the POD model, as for PODI and POD-Galerkin, but also on the choice of the domain decomposition. This aspect introduces an additional layer of complexity in the tuning of the strategy.

However, thanks to its low intrusiveness and limited offline cost, the zonal-POD method represents a competitive tool for model reduction of large-scale aerodynamic problems, that may be efficiently employed in an industrial framework, as addressed in Chapter 8.

Chapter 6

Accuracy Estimation

Part of the work described in this chapter has been previously submitted for publication in Bergmann et al. (2018) and Salmoiraghi et al. (2018).

In this chapter we focus on the accuracy estimation of the zonal-POD method, and in particular on the choice of the domain decomposition (Section 6.2) and on an efficient initial sampling of the parameter space (Section 6.3).

6.1 Cross-validation

As already addressed, by construction the POD basis gives an optimal representation, in terms of energy, of the solution space. This means that, in the limit of no-compression, the basis is capable to reproduce exactly the snapshots belonging to the solution database. As a consequence, the zonal-POD algorithm converges to the exact solution for in-sample configurations, neglecting numerical errors that may arise from a slightly different discretization, e.g. when surface morphing requires a new computational grid generation for the reduced domain.

Nevertheless, a good ROM should ensure sufficiently robust and accurate results over the entire parameter space. In terms of accuracy, the approximation error for out-of-sample simulations depends on both the choice of the decomposition and the sampling of the parameters space used to create the database. While this analysis can always be done a posteriori, it is usually not clear how to proceed a priori, especially when very limited data is available in early stages of design. Despite guaranteeing significant time savings, it should be remembered that the computational cost of the zonal-POD

simulation is not negligible with respect to the complete model, since it requires to solve the FOM over a subset of the original domain, leading to a less severe on-line reduction compared to standard ROMs. Thus, the model cannot be extensively used to assess accuracy and we must rely only on relevant information already available in the initial dataset, that may be generated randomly or through other criteria.

In order to identify an estimation method characterized by low bias and variance, we employ cross-validation over the initial set of instances $\mathcal{D} = \{\mathbf{z}_i = (\boldsymbol{\mu}_i, \mathbf{y}_i), \quad i = 1, \dots, N\}$, with $\boldsymbol{\mu}_i$ belonging to the input space $\mathcal{M}$, i.e. the parameter space, and $\mathbf{y}_i$ representing the corresponding solution in the output space $\mathcal{Y}$. Generally speaking, cross-validation is used in statistics to estimate the capability of the model to perform on previously unseen data, i.e. on predictions. The rationale behind the so-called k-fold cross-validation, also known as rotation estimation (see Kohavi (1995)), is to divide the dataset $\mathcal{D}$ into k disjoint subsets $\mathcal{D}_i$, i.e. the folds, and to train the resulting k models on $\mathcal{D} \setminus \mathcal{D}_i$ and test them on all the remaining partitions $\mathcal{D}_i$. It is then possible to formally define the cross-validation estimate of accuracy as:

$$R_{\text{CV}} = \frac{1}{N} \sum_{\mathbf{z}_i \in \mathcal{D}} \delta\left(\mathcal{I}(\mathcal{D} \setminus \mathcal{D}_i, \boldsymbol{\mu}_i), \mathbf{y}_i\right) \ , \tag{6.1}$$

where the term $\delta\left(\mathcal{I}(\mathcal{D} \setminus \mathcal{D}_i, \boldsymbol{\mu}_i), \mathbf{y}_i\right)$, sometimes referred to as loss functional, represents the error made when predicting the output $\mathbf{y}_i$ through $\mathcal{I}(\mathcal{D} \setminus \mathcal{D}_i, \boldsymbol{\mu}_i)$, i.e. the model trained over partitions not containing the instance $(\boldsymbol{\mu}_i, \mathbf{y}_i)$. Clearly such empirical estimate depends on the dataset partitioning: complete k-fold cross-validation will require to evaluate all the possible combinations for choosing the N/k sets, but it is usually too expansive to perform. The simplest way to obtain a complete cross-validation, is to adopt the N-fold cross-validation, usually referred to as *leave-one-out method* (Geisser (1993)). In the leave-one-out approach k is chosen equal to the number of samples in the database and each instance is removed from the training set and used to test the model. In terms of cross-validation variance, the following result from Kohavi (1995) holds.

Lemma 6.1 (Variance of k-fold cross-validation)**.** *If the training algorithm for a system is stable with respect to the perturbations caused by deleting the instances during the cross-validation procedure, the variance of the k-fold estimate will be approximately*

$$Var = R_{CV}\frac{(1 - R_{CV})}{N} \tag{6.2}$$

where N represents the number of instances in the dataset.

Since the variance in Lemma 6.1 does not depend on k, it follows that, providing a sufficiently large dataset, the variance of the leave-on-out estimate coincides with the k-fold one. For the leave-one-out estimator, it is intuitive to understand that the effect of removing a single instance from the loss functional tends to vanish increasing the size of the dataset, and then the model will become stable.

Another attractive quality of the leave-on-out estimate is that it provides an almost unbiased estimate of the classifier generalisation ability (see for example Cawley and Talbot (2003)).

Lemma 6.2 (Bias of leave-on-out cross-validation)**.** *The leave-one-out cross-validation gives an almost unbiased estimate in the following sense:*

$$E_{\mathcal{D}}[R_{loo}(\mathbf{y}_{\mathcal{D}})] = E_{\mathcal{D}'}[R_{gen}(\mathbf{y}_{\mathcal{D}})] \tag{6.3}$$

where R_{loo} is the leave-one-out error (from Equation (6.1)) and $R_{gen}(\mathbf{y}) = E_{\mathbf{z}}[\delta(\mathcal{I}(\mathcal{D}, \boldsymbol{\mu}), \mathbf{y})]$, i.e. the generalization error of $\mathbf{y}$ w.r.t. the loss δ, evaluated on the $\mathcal{D}'$ set of dimensions $N - 1$.

This lemma, however, is not valid when the learning algorithm is unstable, in which case the use of the leave-one-out error is not recommended. A comprehensive review of the leave-one-out method strengths and drawbacks, in the framework of machine learning, can be found in Elisseeff and Pontil (2003). For the applications of interest, the amount of training data available is usually very limited, due to the cost of FOM evaluations. As a consequence, small perturbations of the data are likely to result in significant changes in the model. Thus, it makes sense for us to adopt the leave-one-out strategy rather than k-fold cross-validation, since it is less affected by such perturbations.

In terms of the zonal-POD, we perform the leave-one-out cross-validation over the snapshots database in order to assess the goodness of the POD model. For $i = 1, \dots, N$ we iteratively remove the snapshot $(\boldsymbol{\mu}_i, \mathbf{y}_i)$ from the dataset and project it onto the subspace spanned by the POD basis built with all the remaining ones. In this way, it is possible to quantify how much the snapshots in the database are mutually independent: if its associated projection error tends to zero, the snapshot can be expressed as a linear combination of the others, and therefore it adds no significant information to the database, whereas relevant points in the parameter space are characterized by high projection errors.

Such information is used differently in order to either identify the domain decomposition or select a sufficiently representative database of simulations, as explained in detail

in Sections 6.2 and 6.3. Clearly, the two aspects are not uncorrelated: generally speaking, we can reasonably assume that a finer sampling will permit to reduce the extent of the full-order region in the hybrid approach, whereas a poorer exploration of the parameter space will required a more disadvantageous decomposition.

6.2 Domain decomposition

In terms of domain decomposition, two problems need to be addressed:

1. the detection of the crucial regions where the POD basis fails to represent non-linearities, and therefore the position of the boundary Γ_1;
2. the choice of the overlapping region, Ω_0.

6.2.1 Interface detection

Focusing on the first aspect, the leave-one-out out-of-sample estimate allows us to obtain a spatial error map for each snapshot, based on the POD projection error, defined as:

$$e^{(k)}(\mathbf{x}) = |\mathbf{y}(\mathbf{x}, \boldsymbol{\mu}_k) - \mathbf{y}^*(\mathbf{x}, \boldsymbol{\mu}_k)| \ ,$$

where the vector field $\mathbf{x}$ represents the spatial coordinates and $\mathbf{y}^*(\boldsymbol{\mu}_k) := \Pi^{\mathrm{POD}}(\mathbf{y}(\boldsymbol{\mu}_k))$, using the POD projection operator definition introduced in Equation (5.11) (for the sake of simplicity, the $\mathbf{x}$ dependence is omitted). Since the k^{th} snapshot is not employed in the basis construction, the corresponding error distribution represents an out-of-sample estimate of the error associated to the prediction of new configurations.

By repeating this procedure for every snapshot in the database, it is possible to combine all these error fields, in order to evaluate a global error map, defined as the envelope of the maximum errors over the database:

$$e(\mathbf{x}) = \max_{1 \leq k \leq N} e^{(k)}(\mathbf{x}) \ . \tag{6.4}$$

Such error distribution map can be used to identify the most critical domain regions, where the representation of the flow field by the POD basis is strongly affected by a change in the input parameters.

The main steps of the strategy are summarised in Algorithm 3.

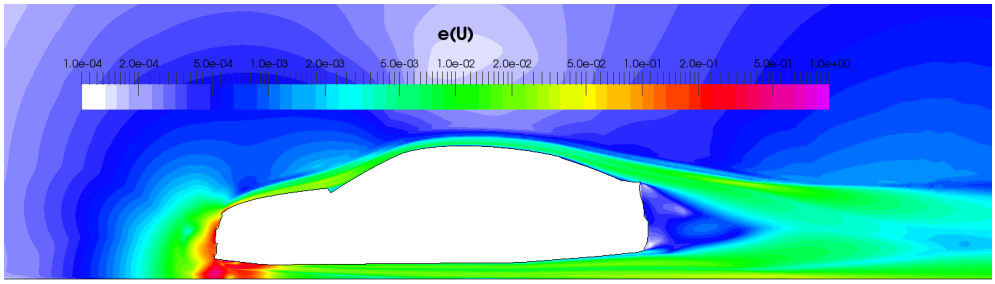
Algorithm 3 Leave-one-out algorithm for Γ_1 detection

1: $\boldsymbol{\Xi} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, set of parametric points
2: $\boldsymbol{\Theta} = [\mathbf{y}(\boldsymbol{\mu}_1), ..., \mathbf{y}(\boldsymbol{\mu}_N)]$, database of N solutions
3: **for all** $\boldsymbol{\mu}_k$ in $\boldsymbol{\Xi}$ **do**
4: $\quad \boldsymbol{\Theta}_k \leftarrow$ remove $\mathbf{y}(\boldsymbol{\mu}_i)$ from $\boldsymbol{\Theta}$
5: $\quad$ POD basis $\leftarrow \boldsymbol{\Theta}_k$
6: $\quad \mathbf{y}^*(\boldsymbol{\mu}_k) \leftarrow$ POD projection
7: $\quad e^{(k)}(\mathbf{x}) = |\mathbf{y}(\boldsymbol{\mu}_k) - \mathbf{y}^*(\boldsymbol{\mu}_k)|$
8: **end for**
9: $e(\mathbf{x}) = \max_{1 \leq k \leq N} e^{(k)}(\mathbf{x})$

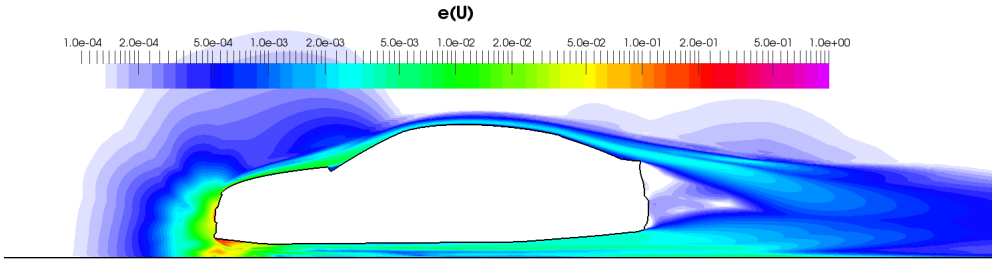
Since we use a different POD basis for each involved physical quantity, in order to have a better representation, the leave-on-out strategy described above can be employed to individually compute the error map indicator for all the unknowns, i.e. the velocity field $\mathbf{u}$, the pressure p and the turbulent quantities. Although it is reasonable to suppose that the larger error values will be localized near the body, since the effects of the flow perturbation introduced by the varying geometry are expected to be weakly-non linear in the far field, there is no evidence that the different error maps will present the same structures. On the contrary, it won't be the case.

Let us consider the 2DCAR benchmark. As expected, the most crucial zones are localised close to the front bumper and in the turbulent wakes, as shown in Figures 6.1 and 6.2, reporting, for different sizes of the initial database, the global error maps for both velocity and pressure, i.e. $e_{\mathbf{u}}$ and e_p. In particular we plot the results for three uniform samplings on the two-dimensional parameter space, corresponding to $N = 4$, $N = 9$ and $N = 25$ snapshots. The errors are normalised with respect to the reference far-field velocity, U_∞, and the reference dynamic pressure $\frac{1}{2}\rho U_\infty^2$, common among all the snapshots. As expected, by enriching the database the maximum error is reduced, as well as the extension of critical regions, i.e. where the POD basis fails to represents the non-linear phenomenology. When such zones cover almost the full Ω domain, this indicates that the database does not contain enough information and should be improved, as we will address in Section 6.3. In this example, the RIC analysis confirms that all the snapshots provide a sensible contribution to the energy, and therefore the POD bases are not truncated.

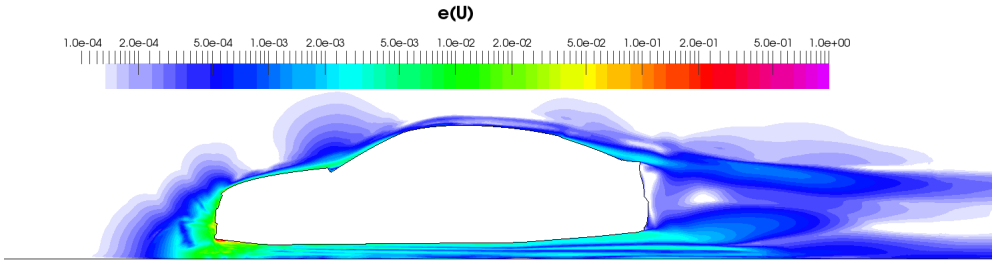
The domain Ω_1, used for the zonal-POD simulations, is then chosen as the quasi-rectangular bounding box surrounding the deformable part of the geometry and containing all the cells characterized by an error on the velocity larger than a certain threshold



(a) $N = 4, M_r = 3$.

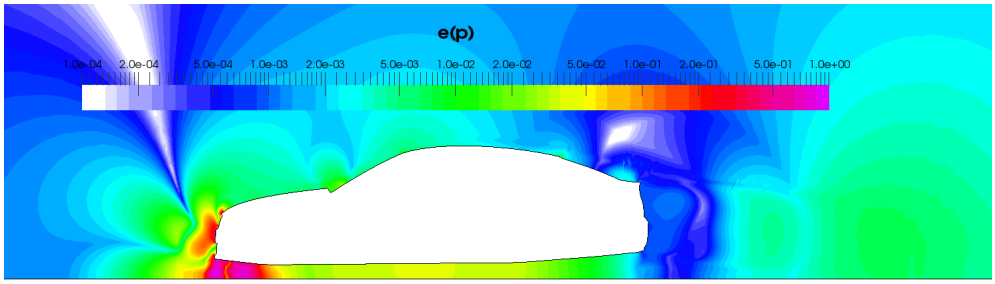


(b) $N = 9, M_r = 8$.

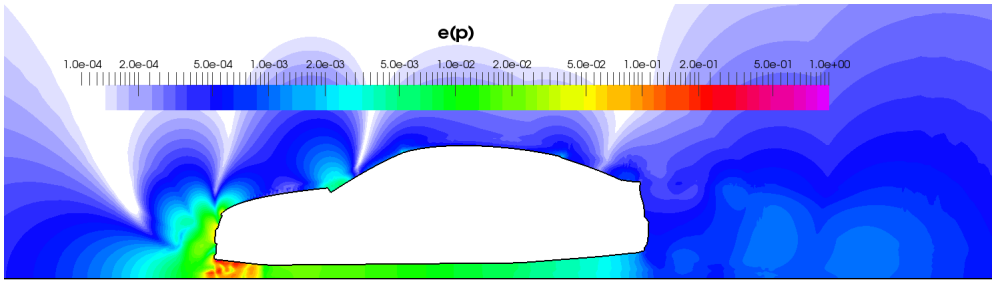


(c) $N = 25, M_r = 24$.

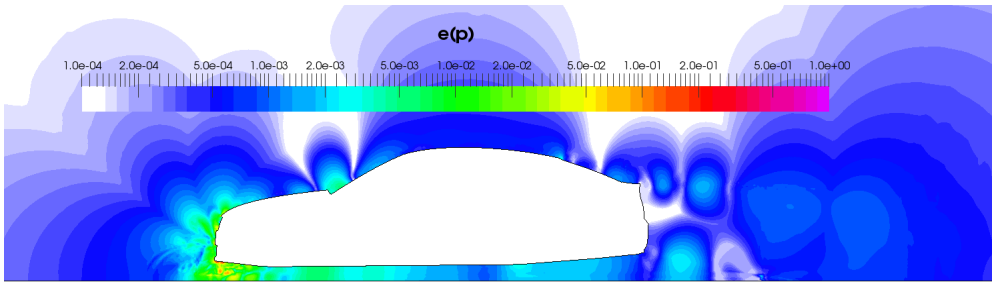
Fig. 6.1 Leave-one-out velocity prediction error for the 2DCAR benchmark, using three different databases.



(a) $N = 4, M_r = 3$.

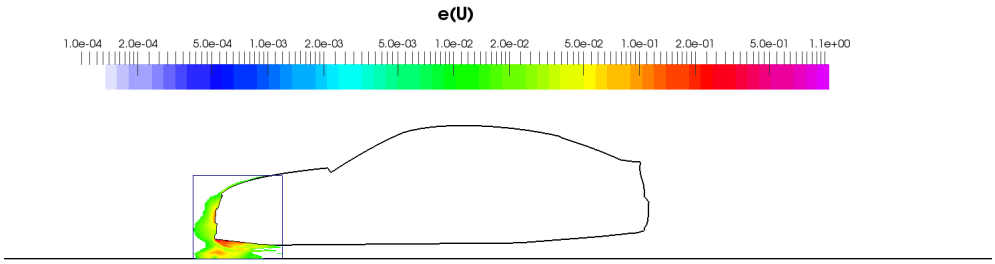


(b) $N = 9, M_r = 8$.

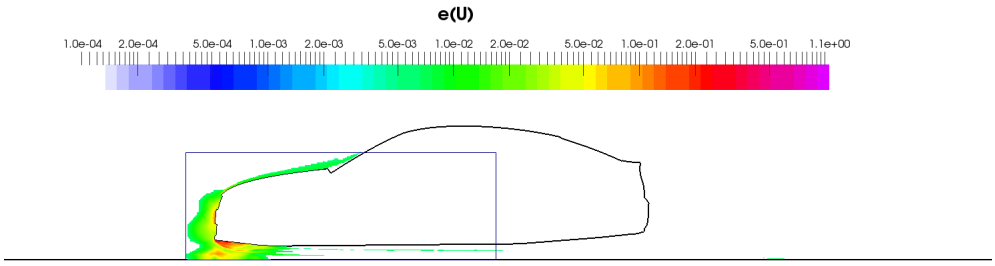


(c) $N = 25, M_r = 24$.

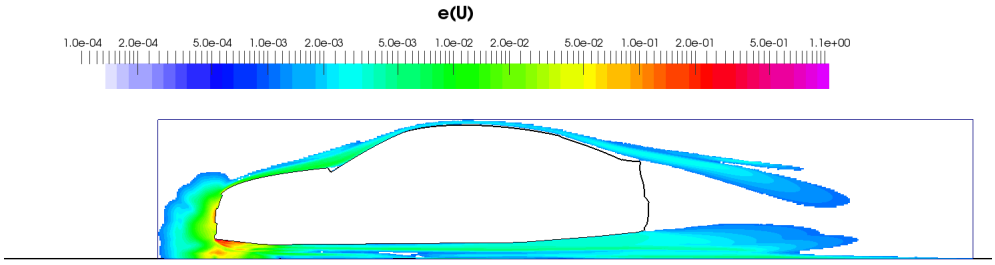
Fig. 6.2 Leave-one-out pressure prediction error for the 2DCAR benchmark, using three different databases.



(a) $\sigma_R = 0.01U_\infty$.



(b) $\sigma_R = 0.005U_\infty$.



(c) $\sigma_R = 0.001U_\infty$.

Fig. 6.3 2DCAR benchmark: Ω_1 domain varying the error threshold σ_R.

value σ_R, i.e. $\Omega_1(\boldsymbol{\mu}) \subset \{\forall \mathbf{x} | e(\mathbf{x}) > \sigma_R\}$. This approach forces to include in Ω_1 also cells that are well represented by the POD model, penalising the achievable reduction. In principle, considering only those cells whose error exceeds the assigned threshold would fix this issue, but it may results in a domain $\Omega_1(\boldsymbol{\mu})$ given by a set of disjoint subdomains, each with a number of cells equal or greater than 1: an occurrence that may originate problems for the CFD solver, as it happens for the numerical tools employed in this work. Since we aim at developing a general and easy-to-implement strategy within the canonical CFD solver, the definition of the reduced domain through the bounding-box of the error, albeit not optimal, leads to a remarkable simplification in the workflow.

Figures 6.3 and 6.4, both obtained with $N = 9$, show how the size of Ω_1 tends to decrease for higher values of the error threshold σ_R. Given a certain database, if the field reconstruction is not sufficiently good, it is always possible to extend the domain

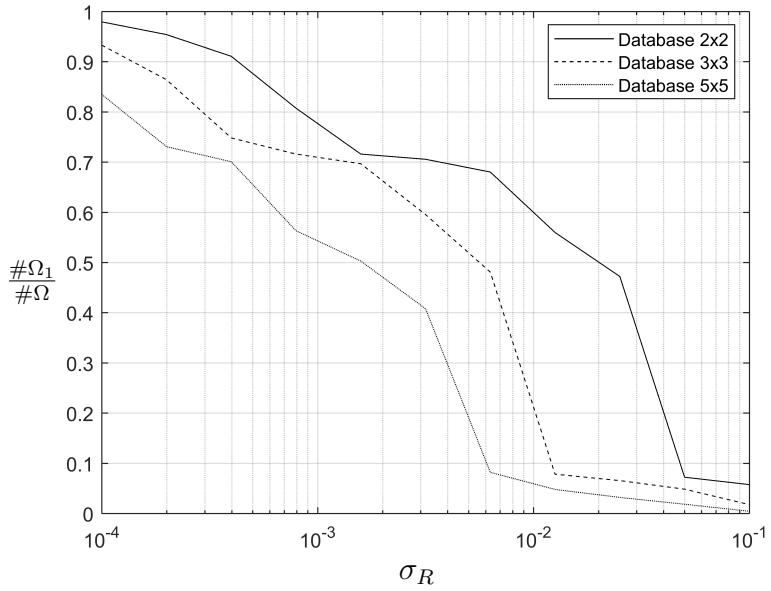


Fig. 6.4 Number of cells in Ω_1 normalised w.r.t. the cells in Ω as a function of the prediction error threshold σ_R.

where we solve the FOM and obtain more accurate results: obviously, the larger Ω_1, the less we gain in terms of computational speed-up. In the limit of σ_R tending to zero, Ω_1 will coincide with the original Ω, and the full-order problem will be solved on the whole domain, vanishing the computational gain of the zonal-POD approach. Therefore, a trade-off between accuracy and speed-up is always required. For example, if we choose to include in Ω_1 all those cells with $\sigma_R > 0.01U_\infty$, the resulting reduction of the problem size will be of a factor 10.

In this example, since mesh morphing is involved, the computational grid in Ω_1 is a subset of the original mesh in Ω, and it maintains the same resolution and topology. When dealing with surface morphing, a new grid needs to be generated in Ω_1, taking care to guarantee the same resolution as in the FOM simulations. Note that this is a general requirement, since the grid resolution determines the spatial scales that are present in the solution snapshots, and, consequently, in the POD model.

Besides being non-intrusive and efficient to compute, this error indicator built on the full-order solutions is also correlated with the predictive performance of the hybrid model, as shown in Figure 6.5, where we report the error on the overall drag coefficient varying the size of the full-order domain. In particular, we perform the predictions using the domain Ω_1 corresponding to two different error thresholds, i.e. $\sigma_R = 0.017U_\infty$ and $\sigma_R = 0.001U_\infty$. The average error in the output of interest drops from 0.9341%

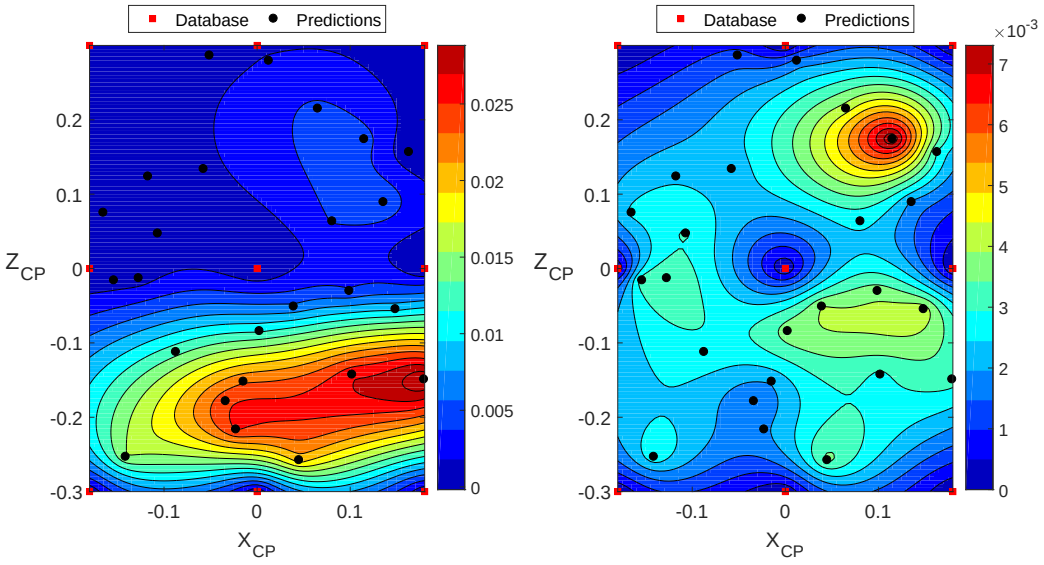


Fig. 6.5 2DCAR prediction error ϵ_{C_x} on the drag coefficient varying the decomposition with $N = 9$, $M_r = 8$: $\sigma_R = 0.017U_\infty$ (left) vs $\sigma_R = 0.001U_\infty$ (right).

to 0.2992%: this accuracy increase, however, is traded with a sensible reduction of the hybrid model geometrical speed-up, that shift from a factor 10 to a factor 2.

6.2.2 Overlapping detection

Regarding the identification of the overlapping region, where the full-order and reduced-order model are coupled, it would seem natural to set $\Omega_0 = \Omega_1(\boldsymbol{\mu})$, considering the benefits in terms of algorithm convergence. Nevertheless, two additional considerations need be made. First, when dealing with surface morphing, the most natural choice is to define the POD modes in physical space, by interpolating for instance all the snapshots on a reference grid. However, the solution domain changes for different configurations, meaning that the spatial coordinates affected by the geometry variation in some configurations belong to the solution domain, whereas in other cases this is no longer true. In order to define the spatial correlation of the snapshots and compute the POD, the region covered by the shape deformation is removed. Consequently, we have that Ω_0 is a certain subset of $\Omega_1(\boldsymbol{\mu})$ and does not depend on the parameter.

Second, we have to recall that the best POD approximation that we can obtain for an out-of-sample configuration in Ω_2, namely $\mathbf{y}_\mathrm{p} = \mathbf{y}(\boldsymbol{\mu}_\mathrm{p})$, is given by the orthogonal POD

projection $\mathbf{y}_\mathrm{p}^\star = \prod^{\mathrm{POD}}(\mathbf{y}_\mathrm{p})$, that corresponds to the solution of the least-squares problem

$$\min_{\alpha} \parallel (\mathbf{y}_\mathrm{p} - \bar{\mathbf{y}}) - \boldsymbol{\Psi}\boldsymbol{\alpha} \parallel^2_{\Omega_2} \quad .$$

However, this is not generally true if we solve the same problem in an arbitrary chosen $\Omega_0 \subset \Omega_2$, meaning that if we need to recover the solution outside Ω_0, the approximation is no longer optimal. In principle, in the zonal-POD approach we simulate through the FOM those regions that are most affected by the perturbations introduced by the shape deformations: thus, it is safe to assume that the modes will be strong in Ω_1, whereas they will tend to vanish at the far-field, where the solution is given mostly by the average field. Therefore, the contribution of the outer region to the POD projection should be minimal. When this assumption is not valid, it could be useful to investigate how the choice of the overlapping region influences the prediction performance of the method. In order to do so, we follow the same rationale of the Missing Point Estimation (MPE) (Astrid et al. (2008)). The MPE aims at reducing the computational cost associated with the solution of reduced-order systems, by evaluating the residual vector only on a subset of the computational points (control volumes) (see for instance Vendl et al. (2014) and Zimmermann and Willcox (2016) for POD-based applications).

Let us consider all the n cells included in Ω_0, and arrange their indices in the vector $\mathbf{C} = (1, ..., n)$. Let $\mathbf{C}_0 = (c_1, ..., c_{n_0})$ denote a subset of Ω_0, where n_0 is the number of selected points and $c_1, ..., c_{n_0}$ their indices. For each flow variable we can define a filtering matrix as

$$\mathbf{P} := [\mathbf{e}_{c_1}, ..., \mathbf{e}_{c_{n_0}}] \in \mathbb{R}^{n \times n_0} \ ,$$

whose columns, $\mathbf{e}_i \in \mathbb{R}^n$, are the canonical unit vectors. We want then to identify an optimal subset of points in Ω_0 such that the solution of the following *masked least-squares* problem

$$\min_{\alpha} \parallel \mathbf{P}^\top(\mathbf{y}_\mathrm{p} - \bar{\mathbf{y}}) - \mathbf{P}^\top\boldsymbol{\Psi}\boldsymbol{\alpha} \parallel^2_{\Omega_0}$$

corresponds to a good approximation of $\mathbf{y}_\mathrm{p}$ in Ω_2. Rather than using fast low-rank approximations, computationally intractable for our applications, we select the subsets based on the flow physics, and in particular according to given leave-one-out error thresholds: then, several hybrid simulations are performed, aimed at identifying the best Ω_0 with respect to a certain goal function (e.g. some aerodynamic coefficient). Despite making the offline phase more demanding, this approach does not show a significant improvement in the 2DCAR benchmark, as illustrated in Figure 6.6. Consequently, we will select the whole Ω_0 in the following.

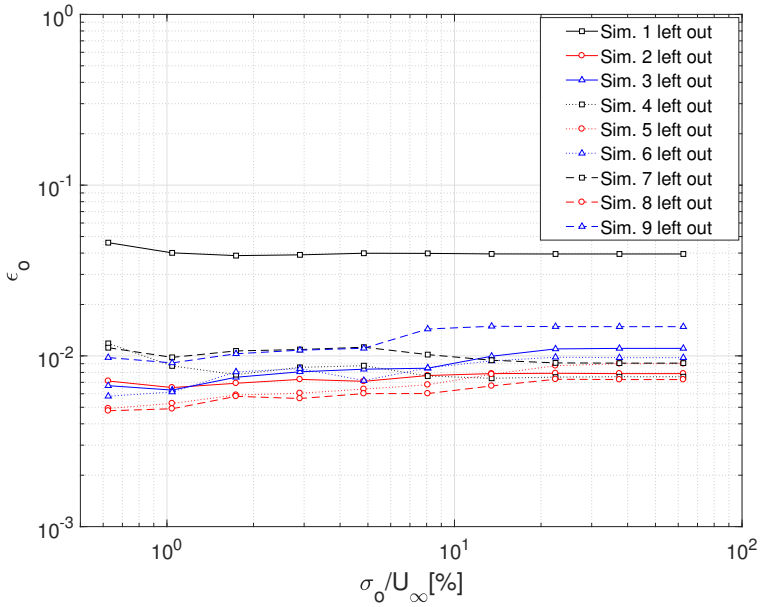


Fig. 6.6 L^2-norm error on the boundary conditions imposed on Γ_1, as a function of the error threshold, using $N = 9$ snapshots.

In other cases, however, such effort seems legitimate, as in the problem of the interaction between a NACA0012 airfoil and a vortex, studied in Bergmann et al. (2018) with the same methodology.

6.3 Snapshots selection

Since the systematic exploration of the parameter space is usually prohibitive for the applications of interest, efficient sampling strategies must be introduced. With respect to standard near-random or uniform sample generation, the *greedy* method (see for instance Hesthaven et al. (2016)) represents a natural choice for optimal space identification. The basic idea is to define a sharp and inexpensive a posteriori error bound, $\Delta_m(\boldsymbol{\mu})$, and to use this information to drive the algorithm, by computing only winning candidate snapshots. The greedy sampling strategy is then an iterative procedure, in which new snapshots are sequentially added in order to improve the overall precision of the POD model. Formally, we have to provide an estimate of the error due to the ROM approximation, that satisfies:

$$\|\mathbf{y}(\boldsymbol{\mu}) - \mathbf{y}^\star(\boldsymbol{\mu})\|_X \leq \Delta_m(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathcal{M}$$

with $X \subseteq \Omega(\boldsymbol{\mu})$. Here we consider the L^2 norm defined over a certain subset of the domain, but alternatively it is possible to use other norms, or directly the measure of the error over the optimization objective function $|f(\boldsymbol{\mu}) - f^\star(\boldsymbol{\mu})|$, for a goal-oriented approach. At the m-iteration of the procedure, the next point to be evaluated through the FOM is selected as:

$$\boldsymbol{\mu}_{m+1} = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \Delta_m(\boldsymbol{\mu})$$

and then we compute the corresponding solution $\mathbf{y}(\boldsymbol{\mu}_{m+1})$, to be added to the snapshots database. Convergence is attained where the maximum estimated error drops below a certain value.

Even if its cost is small, the error bound cannot be computed over the entire parameter space: therefore it is necessary to define a suitable and finite trial set, $\boldsymbol{\Xi}_{\text{trial}}$, for the evaluation of $\Delta_m(\boldsymbol{\mu})$.

In the following section we investigates two possible definitions of such error indicator, coupled with different strategies for the trial set selection.

6.3.1 Error indicator based on the NSE residuals

In the first approach, the error indicator is based on the residuals of the NSE computed using the flow fields predicted by the POD model, following the same idea proposed by Grepl and Patera (2005) and Lombardi et al. (2011) for POD-Galerkin approaches. In our case, we do not have an approximation of the Navier-Stokes operator, so we evaluate the residual corresponding to a given parameter value by substituting the predicted solution into the FOM discretization of the NSE. In this framework, the predictive solution $\mathbf{y}^\star(\boldsymbol{\mu}_t) = (\mathbf{u}^\star(\boldsymbol{\mu}_t), p^\star(\boldsymbol{\mu}_t))$ is not the one resulting from the zonal-POD simulation, but it is evaluated using only the POD model. Without introducing the numerical discretization, the full-order snapshots are exact solutions to Equations (3.1) and (3.2), while this won't be the case for linear combination of such snapshots, given the non-linear nature of the Navier-Stokes operator. In terms of the discrete problem, it seems reasonable to assume that the FOM residuals will be lower than the ones computed with the POD solution.

We propose two different approaches based on the NSE residuals, referred to in the following as *resGA-PODI* and *resGA-L1O*. In the first strategy, summarized in Algorithm 4, we computed the complete POD basis using all the snapshots available and then we employ POD with interpolation, as described in Section 5.2.2, in order to evaluate the approximate $\mathbf{y}^\star$ solutions for different values of the parameter.

Algorithm 4 resGA-PODI
greedy algorithm with Δ_m based on NSE residuals and POD interpolation

$\boldsymbol{\Xi}_0 = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, initial set of parametric points
$\boldsymbol{\Theta}_0 = [\mathbf{y}(\boldsymbol{\mu}_1), ..., \mathbf{y}(\boldsymbol{\mu}_N)]$, initial database of solutions
POD basis $\leftarrow \boldsymbol{\Theta}_0$
select a trial set of new parameters $\boldsymbol{\Xi}_{trial}$
while $\max \Delta_m > tol$ **do**
    for all $\boldsymbol{\mu}_t$ in $\boldsymbol{\Xi}_{trial}$ **do**
        $\alpha(\boldsymbol{\mu}_t)\,, \beta(\boldsymbol{\mu}_t) \leftarrow$ interpolation
        $\mathbf{u}^\star(\boldsymbol{\mu}_t) = \sum_{i=1}^{M_r^\psi} \alpha_i(\boldsymbol{\mu}_t)\boldsymbol{\psi}_i \quad \text{for } i = 1, ..., M_r^\psi$
        $p^\star(\boldsymbol{\mu}_t) = \sum_{i=1}^{M_r^\varphi} \beta_i(\boldsymbol{\mu}_t)\varphi_i \quad \text{for } i = 1, ..., M_r^\varphi$
        $\mathbf{r}^\star(\boldsymbol{\mu}_t) = NS(\mathbf{u}^\star(\boldsymbol{\mu}_t), p^\star(\boldsymbol{\mu}_t))$
        $r^\star(\boldsymbol{\mu}_t) \leftarrow \mathbf{r}^\star(\boldsymbol{\mu}_t)$
        $\Delta_m(\boldsymbol{\mu}_t) = r^\star(\boldsymbol{\mu}_t)$
    end for
    $\boldsymbol{\mu}_{m+1} \leftarrow \arg\max \Delta_m(\boldsymbol{\mu}_t)$
    $\boldsymbol{\Xi}_{m+1} = [\boldsymbol{\Xi}_m, \boldsymbol{\mu}_{m+1}]$
    compute $\mathbf{y}(\boldsymbol{\mu}_{m+1})$
    $\boldsymbol{\Theta}_{m+1} = [\boldsymbol{\Theta}_m, \mathbf{y}(\boldsymbol{\mu}_{m+1})]$
    POD basis $\leftarrow \boldsymbol{\Theta}_{m+1}$
end while

For our applications, the NSE residual is evaluated coherently with the FOM employed. The general approach for the residual computation in the OpenFOAM® solvers is described in the following. Given a matrix system $\mathbf{Mx} = \mathbf{b}$, e.g. the one resulting from the discretization of the momentum equation, the vector of cell residuals is defined as:

$$\mathbf{r} = \mathbf{b} - \mathbf{Mx} \ .$$

The scalar residual r is then computed through a normalization procedure that involves the evaluation of the normalization factor:

$$n = \sum_c \left(|\mathbf{Mx} - \mathbf{M}\bar{\mathbf{x}}| + |\mathbf{b} - \mathbf{M}\bar{\mathbf{x}}| \right) + \varepsilon \ ,$$

where $\bar{\mathbf{x}}$ represents the average over the cells of the solution vector and ε is set to a small value in order to avoid dividing by zero. Finally, r is given by the sum of $\mathbf{r}$ over the cells normalised with respect to the factor n:

$$r = \frac{1}{n} \sum_c |\mathbf{b} - \mathbf{Mx}| \ .$$

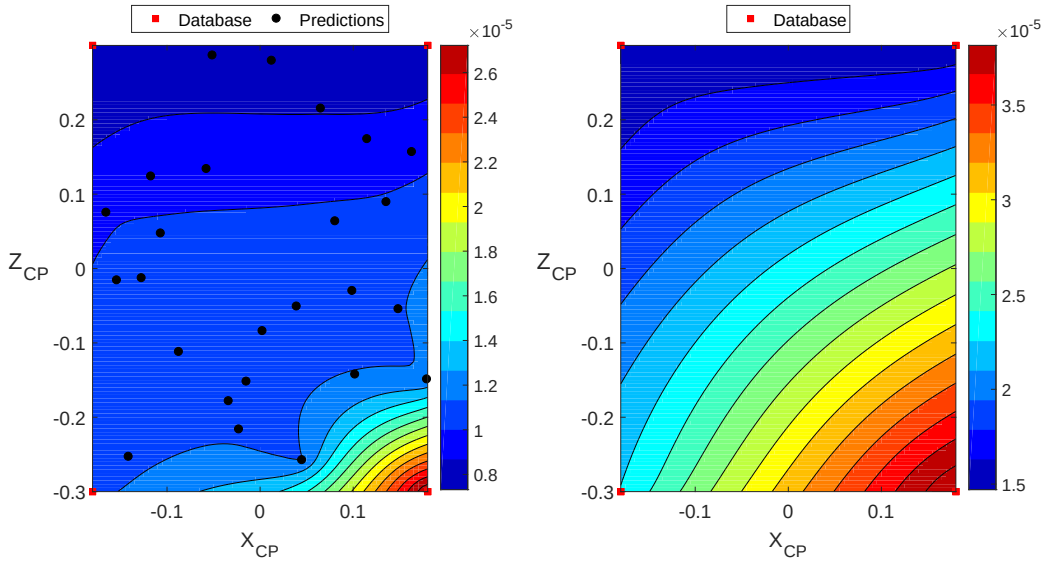


Fig. 6.7 2DCAR benchmark: response surfaces of the error indicator Δ_0 at the first greedy iteration for resGA-PODI (left) and resGA-L1O (right).

Hence, with reference to Algorithm 4, from the $\mathbf{y}^\star$ solutions we compute the corresponding approximate residuals vector and the scalar residual, namely $\mathbf{r}^\star$ and $r^\star$, and then we use the latter to drive the greedy algorithm. Since the interpolation and the residual evaluation can be performed at low cost, in this framework it is possible to select a very fine trial set, using for example quasi-random strategies for points generation, such as Sobolov sequences or latin-hypercube sampling. While showing promising results for trivial laminar test cases, the method fails for industrial turbulent flows, where the NSE residuals are usually higher, in a way that makes it impossible to discern between the error due to the numerical discretization and the one due to the POD approximation, as highlighted in Figure 6.7.

The main issue of this approach is represented by the fact that the greedy error indicator for in-sample configuration does not depend on the quality of the POD model, but it is an intrinsic feature of the flow field that is not affected by the enrichment of the database.

In order to improve the robustness of the algorithm, we propose a modified version based on the leave-one-out method. This allows us to compute the approximate solution $\mathbf{y}_k^\star$ by projecting the FOM solution $\mathbf{y}_k$ onto the POD basis built with the remaining snapshots in the database, and so without introducing an additional approximation layer, that is the interpolation. Moreover, since the full-order solution is available, we can

compare the actual FOM residual with the one computed with the POD solution. The resGA-L1O sampling strategy is outlined in Algorithm 5.

Algorithm 5 resGA-L1O
greedy algorithm with Δ_m based on NSE residuals and leave-one-out cross-validation

$\boldsymbol{\Xi}_0 = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, initial set of parametric points
$\boldsymbol{\Theta}_0 = [\mathbf{y}(\boldsymbol{\mu}_1), ..., \mathbf{y}(\boldsymbol{\mu}_N)]$, initial database of solutions
POD basis $\leftarrow \boldsymbol{\Theta}_0$
select a trial set of new parameters $\boldsymbol{\Xi}_{trial}$
while $\max \Delta_m > tol$ **do**
    for all $\boldsymbol{\mu}_i$ in $\boldsymbol{\Xi}_m$ **do**
        $\boldsymbol{\Theta}_i \leftarrow$ remove $\mathbf{y}(\boldsymbol{\mu}_i)$ from $\boldsymbol{\Theta}_m$
        POD basis $\leftarrow \boldsymbol{\Theta}_i$
        $\mathbf{u}^*(\boldsymbol{\mu}_i), p^*(\boldsymbol{\mu}_i) \leftarrow$ POD projection
        $\mathbf{r}^\star(\boldsymbol{\mu}_t) = NS(\mathbf{u}^\star(\boldsymbol{\mu}_t), p^\star(\boldsymbol{\mu}_t))$
        $r^\star(\boldsymbol{\mu}_t) \leftarrow \mathbf{r}^\star(\boldsymbol{\mu}_t)$
        $\Delta_m(\boldsymbol{\mu}_i) = |r(\boldsymbol{\mu}_i) - r^\star(\boldsymbol{\mu}_i)|$
    end for
    evaluate the response surface with $\Delta_m(\boldsymbol{\mu})$ as density function
    for all $\boldsymbol{\mu}_t$ in $\boldsymbol{\Xi}_{trial}$ **do**
        compute $\Delta_m(\boldsymbol{\mu}_t)$
    end for
    $\boldsymbol{\mu}_{m+1} \leftarrow \arg\max \Delta_m(\boldsymbol{\mu}_t)$
    $\boldsymbol{\Xi}_{m+1} = [\boldsymbol{\Xi}_m, \boldsymbol{\mu}_{m+1}]$
    compute $\mathbf{y}(\boldsymbol{\mu}_{m+1})$
    $\boldsymbol{\Theta}_{m+1} = [\boldsymbol{\Theta}_m, \mathbf{y}(\boldsymbol{\mu}_{m+1})]$
    POD basis $\leftarrow \boldsymbol{\Theta}_{m+1}$
end while

Despite preforming better with respect to the resGA-PODI approach, as shown in Figure 6.7 this second method still suffers from the dependence on the residual definition, that may vary among different CFD solvers and be not totally reliable for steady RANS simulations of turbulent complex flows. For this reason, we focus on a definition of Δ_m based on the POD projection error, as addressed in the following section.

6.3.2 Error indicator based on the POD projection error

The main idea behind this second approach is to use the leave-one-out method to evaluate the POD out-of-sample projection error for each snapshot of the database. The leave-one-out algorithm has been employed in the past as a procedure to perform adaptive

sampling in order to built robust POD models, as in the works of Braconnier et al. (2011) and Zhan et al. (2015).

Constrained Centroidal Voronoi Tessellation

In order to guarantee both the exploration and the exploitation of the parameter space, we coupled the greedy method with a Constrained Centroidal Voronoi Tessellation (CCVT), built using the greedy error indicator Δ_m as density function. We follow the same rationale proposed by Lombardi et al. (2011), but with a different definition of the error. Starting from an initial set of points, $\boldsymbol{\Xi}_0$, containing the vertices of the parametric domain, the new candidate points are represented by the centroids of the Voronoi tessellation elements computed with respect to the chosen density function. Among those points, the greedy algorithm selects the one where the error indicator $\Delta_m(\boldsymbol{\mu}_t)$, weighted with a measure of the corresponding tessellation element, reaches its highest value. In this strategy, well-spaced points are added iteratively, favouring the unexplored regions of the parameter space as well as the ones badly represented by the POD model. The proposed errors, are in fact good local error estimates in the neighbourhood of the points $\boldsymbol{\mu}_i \in \boldsymbol{\Xi}$, but little can be said far from these regions. The *pGA-CCVT* algorithm is outlined in Algorithm 6.

Algorithm 6 pGA-CCVT
greedy algorithm with Δ_m based on leave-one-out POD projection error and CCVT

$\boldsymbol{\Xi}_0 = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, initial set of parametric points
$\boldsymbol{\Theta}_0 = [\mathbf{y}(\boldsymbol{\mu}_1), ..., \mathbf{y}(\boldsymbol{\mu}_N)]$, initial database of solutions
while $\max \Delta_m > tol$ **do**
    for all $\boldsymbol{\mu}_i$ in $\boldsymbol{\Xi}_m$ **do**
        $\boldsymbol{\Theta}_i \leftarrow$ remove $\mathbf{y}(\boldsymbol{\mu}_i)$ from $\boldsymbol{\Theta}_m$
        POD basis $\leftarrow \boldsymbol{\Theta}_i$
        $\mathbf{y}^*(\boldsymbol{\mu}_i) \leftarrow$ POD projection
        $\Delta_m(\boldsymbol{\mu}_i) = \|\mathbf{y}(\boldsymbol{\mu}_i) - \mathbf{y}^*(\boldsymbol{\mu}_i)\|_X$ ▷ if goal-oriented: $f(\mathbf{y}^*(\boldsymbol{\mu}_i))$
    end for
    compute CCVT with $\Delta_m(\boldsymbol{\mu})$ as density function
    compute $\Delta_m(\boldsymbol{\mu})$ of the centroids $\boldsymbol{\mu}_c$
    $\boldsymbol{\mu}_{m+1} \leftarrow \arg\max \Delta_m(\boldsymbol{\mu}_c)$
    $\boldsymbol{\Xi}_{m+1} = [\boldsymbol{\Xi}_m, \boldsymbol{\mu}_{m+1}]$
    compute $\mathbf{y}(\boldsymbol{\mu}_{m+1})$
    $\boldsymbol{\Theta}_{m+1} = [\boldsymbol{\Theta}_m, \mathbf{y}(\boldsymbol{\mu}_{m+1})]$
end while

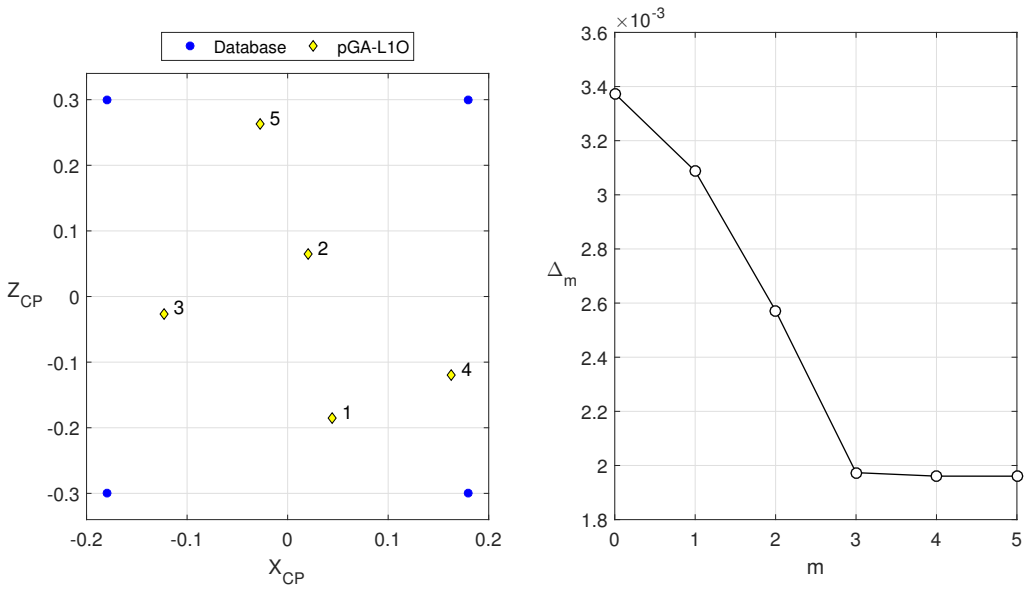
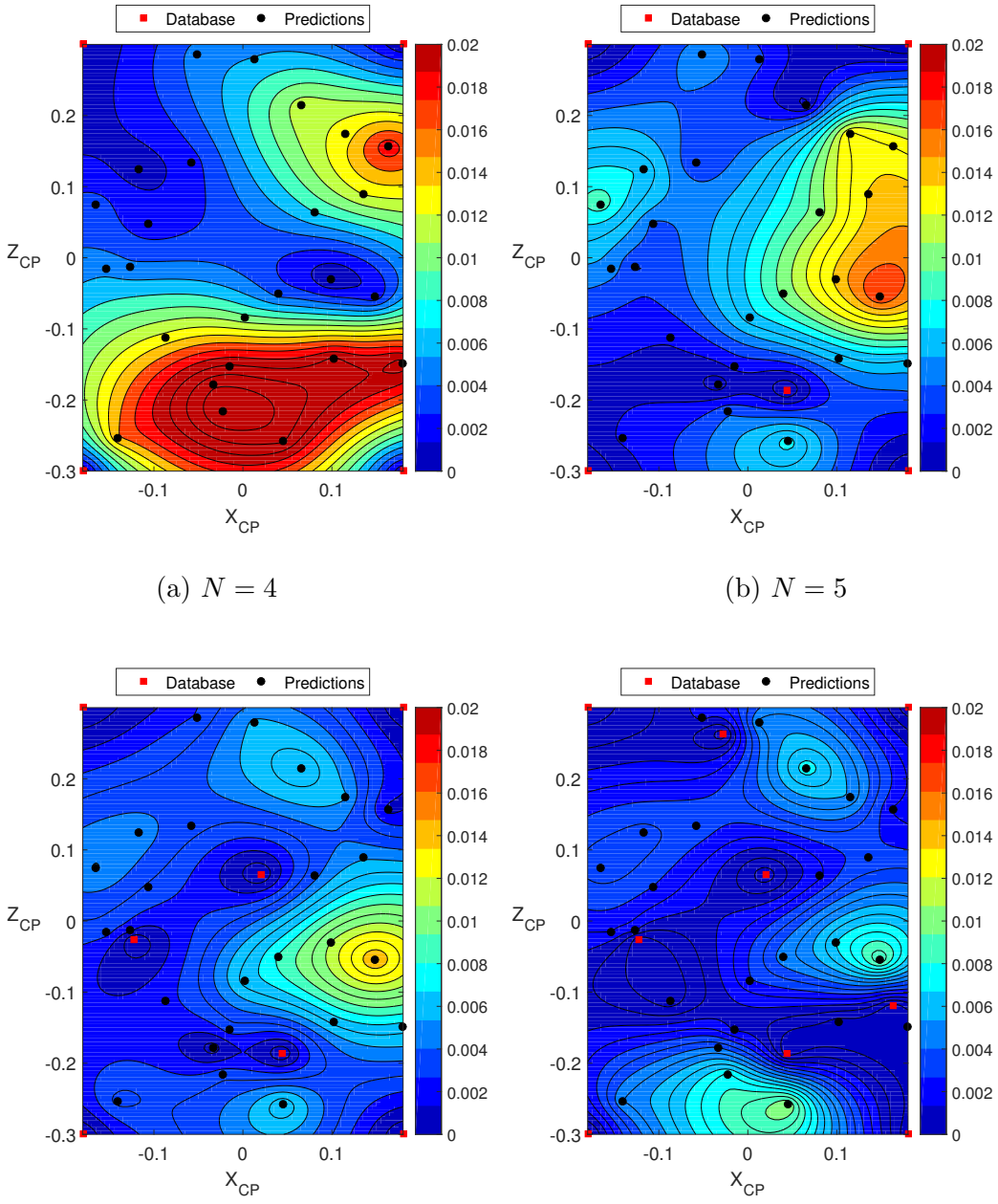


Fig. 6.8 pGA-CCVT algorithm: sampling sequence over the parameter space (left) and trend of the error indicator Δ_m w.r.t. the m^{th} greedy iteration.

The performance of the sampling strategy for the 2DCAR benchmark is shown in Figures 6.8 and 6.9, where we reported the greedy sampling sequence as well as the corresponding output errors on the drag coefficient, starting from an initial distribution containing only the vertices of the parameter space. The average and maximum errors over the prediction points at every greedy iteration, instead, are reported in Table 6.1. With respect to the uniform distribution with $N = 4$ used in Chapter 5, the average error on the prediction drag coefficients, $\bar{e}_{C_x}$, drops from 1.126% to 0.542%, by adding only two greedy points, whereas the maximum error, $\max e_{C_x}$, decreases from 2.584% to 1.379%. Moreover, the out-of-sample accuracy obtained with only 6 solutions is even better than the performance of the database with 9 uniformly distributed snapshots, characterised by an average error of 0.934% and a maximum error of 3.097%. Therefore, the pGA-CCVT sampling strategy represents an improvement with respect to the uniform one, leading to a sensible reduction of the prediction errors and proving that the leave-one-out projection error is a good indicator of the predictive performance of the zonal-POD approach.

Such sampling method can be easily transposed to parameter spaces of P dimension, but it requires the computation of the solutions in the extreme points of the space, i.e. 2^P solutions are needed in order to start the greedy algorithm. Such exponential growth of the computational effort with respect to the number of parameters involved represents the bottleneck of this strategy, that is highly affected by the curse of dimensionality.



(a) $N = 4$ (b) $N = 5$

(c) $N = 7$ (d) $N = 9$

Fig. 6.9 pGA-CCVT algorithm: overall drag coefficient error over the parameter space at different greedy iterations.

In order to mitigate this issue, other methods are taken into considerations. Specifically, another possible way to select the new greedy point is to employ surrogate models that involve interpolation or regression of the error dataset. Similarly to what happens in

Iteration	$\bar{e}_{C_x}$	$\max e_{C_x}$
0	0.01126	0.02584
1	0.00702	0.01763
2	0.00542	0.01379
3	0.00541	0.01521
4	0.00423	0.01081
5	0.00420	0.01073

Table 6.1 Average error $\bar{e}_{C_x}$ and maximum error $\max e_{C_x}$ over the prediction points at every iteration of the pGA-CCVT algorithm.

surrogate-based global optimization (see Chapter 7), an approximation of the error response surface is built over the entire parameter space and the new sampling point is chosen at its maximum. Among the available data fit models, we replace the CCVT with surrogate models based on Gaussian Process (GP) spatial interpolation.

Gaussian-Process Spatial Interpolation

The last strategy considered is based on GP interpolation and on the same principle of the Efficient Global Optimization (EGO) method, that will be discussed in detail in Chapter 7. The basic idea behind the method is to build a GP approximation of the Δ_m values, in the form

$$G_{\Delta_m}(\boldsymbol{\mu}) = \mathbf{h}(\boldsymbol{\mu})^\top \boldsymbol{\beta} + Z(\boldsymbol{\mu})$$

and to select the new candidate point as the one that maximise a certain Expected Improvement Function (EIF), providing a balance between exploiting regions with good solutions and exploring those where the prediction uncertainty is high. At the next iteration, the values of projection errors are updated and a new GP approximation is computed, as illustrated in Algorithm 7.

Algorithm 7 pGA-EGO
greedy algorithm with Δ_m based on leave-one-out POD projection error and GP interpolation

$\boldsymbol{\Xi}_0 = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, initial set of parametric points
$\boldsymbol{\Theta}_0 = [\mathbf{y}(\boldsymbol{\mu}_1), ..., \mathbf{y}(\boldsymbol{\mu}_N)]$, initial database of solutions
while $\max \Delta_m > tol$ **do**
    for all $\boldsymbol{\mu}_i$ in $\boldsymbol{\Xi}_m$ **do**
        $\boldsymbol{\Theta}_i \leftarrow$ remove $\mathbf{y}(\boldsymbol{\mu}_i)$ from $\boldsymbol{\Theta}_m$
        POD basis $\leftarrow \boldsymbol{\Theta}_i$
        $\mathbf{y}^*(\boldsymbol{\mu}_i) \leftarrow$ POD projection
        $\Delta_m(\boldsymbol{\mu}_i) = \|\mathbf{y}(\boldsymbol{\mu}_i) - \mathbf{y}^*(\boldsymbol{\mu}_i)\|_X$ ▷ if goal-oriented: $f(\mathbf{y}^*(\boldsymbol{\mu}_i))$
    end for
    compute $G_{\Delta_m}(\boldsymbol{\mu})$
    compute EIF
    $\boldsymbol{\mu}_{m+1} \leftarrow \arg\max_{\boldsymbol{\mu}}$ EIF
    $\boldsymbol{\Xi}_{m+1} = [\boldsymbol{\Xi}_m, \boldsymbol{\mu}_{m+1}]$
    compute $\mathbf{y}(\boldsymbol{\mu}_{m+1})$
    $\boldsymbol{\Theta}_{m+1} = [\boldsymbol{\Theta}_m, \mathbf{y}(\boldsymbol{\mu}_{m+1})]$
end while

Such strategy recalls the same principle of the pGA-CCVT algorithm, in the sense that it tends to add points also on those regions scarcely populated, as illustrated for instance in Figure 6.10. With respect to the first strategy, the trend of the Δ_m indicator is similar, even though the second approach prefers the exploration of the limits of the space, generating a different selection sequence. In terms of accuracy on the predictions (see Table 6.2 and Figure 6.11), the first iteration halves the errors on the drag coefficients, and the reduction trend is confirmed also by the next iterations. As in the previous case, the improvement with respect to the uniform distribution appears clear, and the performance of the pGA-CCVT and pGA-EGO algorithms after 5 greedy iteration is comparable. It is worth noting, however, that the latter only requires $P+1$ initial points (trend function with polynomial terms of order 0), although a minimum of $\frac{(P+1)(P+2)}{2}$ samples are usually recommended, and so it appears more suitable for large parameterizations.

Even if the pGA-EGO strategy is less affected by the curse of dimensionality than the pGA-CCVT one, both algorithms tend to become extremely costly when employed in high-dimensional parameter spaces, mainly because they required a far greater number of full-order simulations, but also due to the cost of POD model evaluations within the leave-one-out method. Given that we are usually dealing with databases of limited dimensions, the method selected to add and remove solution snapshots to the POD basis is not as critical as it would be for larger problems. However, both the pGA-CCVT and pGA-EGO strategies can benefit from an efficient update of the POD model, based

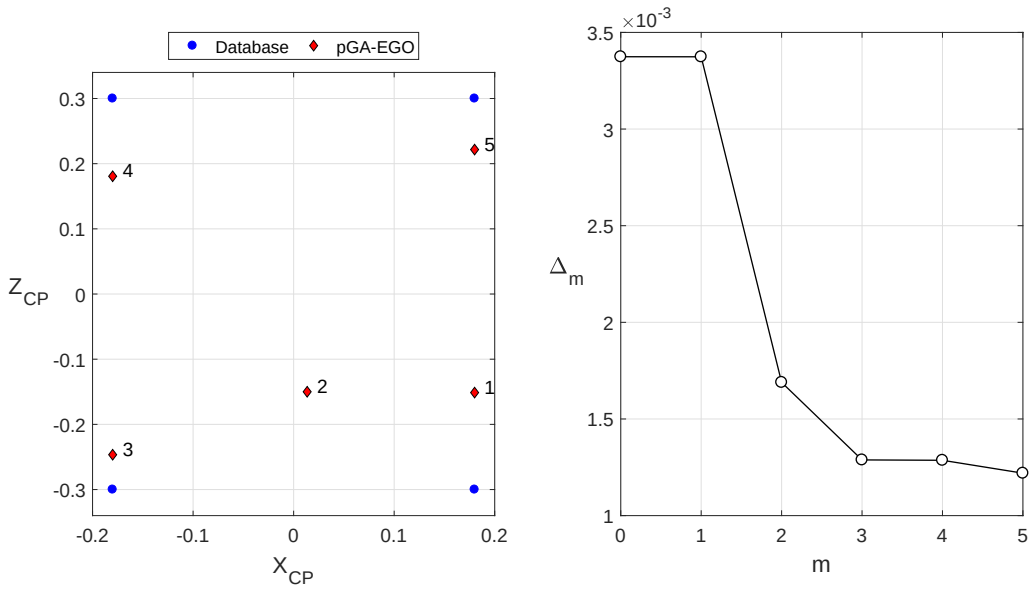
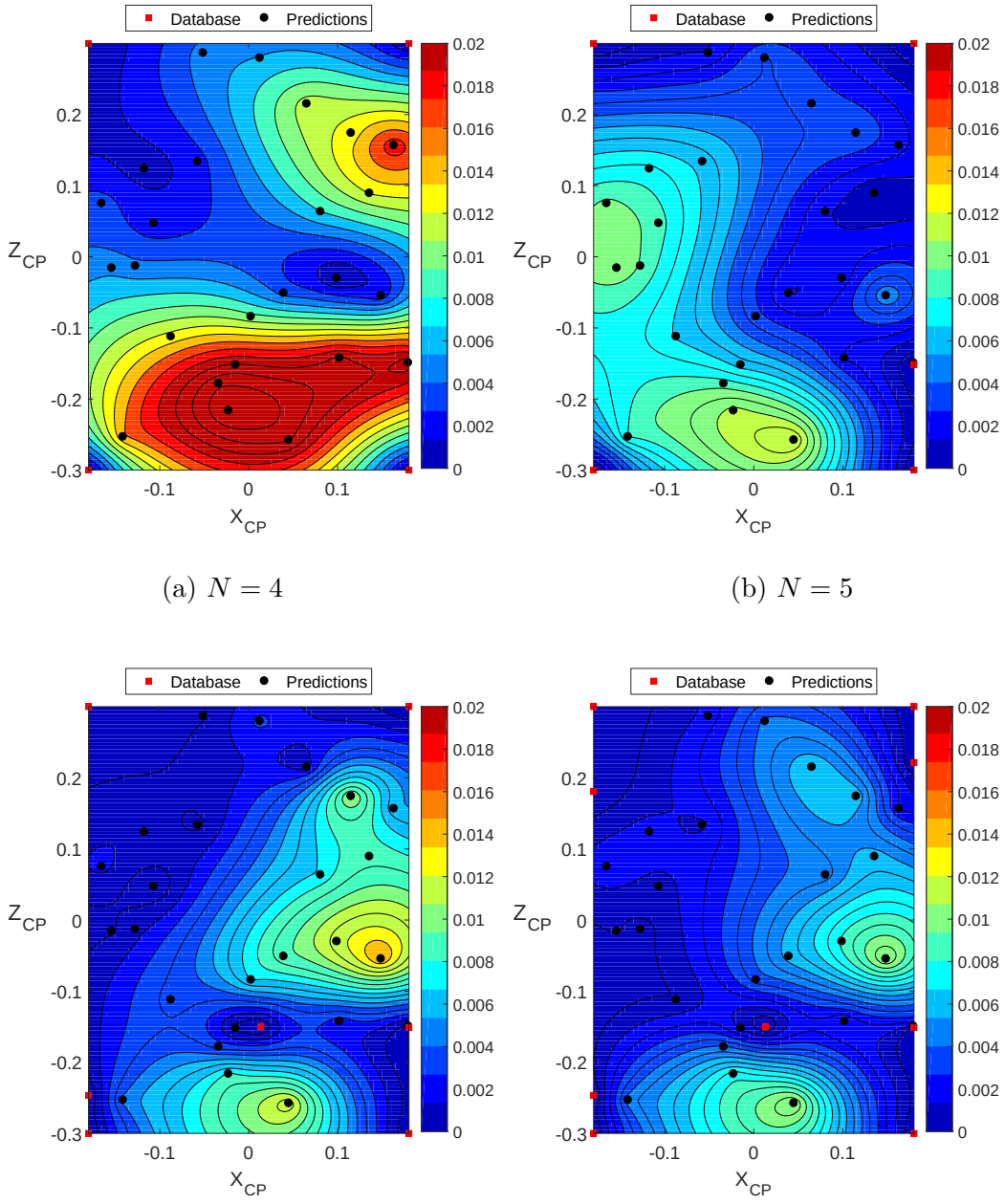


Fig. 6.10 pGA-EGO algorithm: sampling sequence over the parameter space (left) and trend of the error indicator Δ_m w.r.t. the m^{th} greedy iteration.

Iteration	$\bar{e}_{C_x}$	$\max e_{C_x}$
0	0.01126	0.02584
1	0.00559	0.01260
2	0.00515	0.01399
3	0.00497	0.01452
4	0.00489	0.01315
5	0.00400	0.01116

Table 6.2 Average error $\bar{e}_{C_x}$ and maximum error $\max e_{C_x}$ over the prediction points at every iteration of the pGA-EGO algorithm.

for instance on a SVD-update, as suggested by Zimmermann (2011) in his analysis of state-of-the-art algorithms. Additionally, even if the number of design variables is about 10-20 for the applications of interest, in other fields, such as aeronautics, it is not unusual to consider up to $O(100)$ parameters (see for instance Han et al. (2018)). In these cases, both the approaches may become intractable. For a given objective functional, as the leave-one-out error, the right choice of initial and in-fill sampling points is not a trivial problem and has been the main focus of several studies on surrogate-based optimization, without reaching a general rule. Numerical and computational problems associated with the EGO algorithm, and in general with surrogate models are further discussed in Chapter 7. It is worth noting, however, that one of the strengths of the zonal-POD method is that it requires a lower number of samples with respect to other POD-based



(a) $N = 4$ (b) $N = 5$

(c) $N = 7$ (d) $N = 9$

Fig. 6.11 pGA-EGO algorithm: overall drag coefficient error over the parameter space at different greedy iterations.

reduced-order techniques, and it makes possible to accelerate the FOM simulations, with reasonable results, also when the database is rather small. For high-dimensional optimization problems, it is more likely to allocate a fraction of the computational budget

for the training of the POD model, instead of fixing an error threshold on the greedy indicator. For this reason, it is preferably for us to asses the sampling performance in the first iterations, rather than in the long run.

Chapter 7

Optimization

In this chapter we provide an overview of the optimization algorithms and tools employed in the present work.

7.1 Methods

Optimization methods can be classified according to different criteria. In terms of *search method*, i.e. the approach used by the algorithm to locate new candidate points, two different classes of algorithms exist:

- **Gradient-based methods**, where gradients of the response functions are evaluated in order to identify the best direction of improvement;
- **Derivative-free methods**, such as pattern search methods and genetic algorithms, that do not rely on derivatives computation.

Generally speaking, gradient-based optimization methods are usually highly efficient in terms of convergence rates and represent the best choice for convex design spaces or when the local minimum is close to the initial point. However, they are lacking in robustness for more general cases, e.g. for non-smooth, non-convex or poorly behaved problems, and are rather sensitive to initial guesses. A critical aspect for these optimizers is represented by the accuracy of the computed derivatives: analytic gradients and Hessians would be ideal, but for most engineering applications this information is not available, and a finite difference method will be used to find a numerical estimate of the derivatives. Inaccuracy in this evaluation, due for instance to numerical noise or improper

finite difference step size, will lead to failures or unreliable results. In addition, numerical gradients are usually computationally expensive, since their cost is often associated with the number of design variables. Such issue may be overcome by using the adjoint method (see for instance Giles and Pierce (2000)), that represents the most efficient approach for gradient evaluation in very large design spaces, since its cost is virtually independent on the number of design variables. Despite that, adjoint solvers are difficult to be implemented in pre-existing numerical codes and represent and extremely intrusive tool for optimization. Derivative-free approaches, on the contrary, are more robust and intrinsically parallel, besides being designed to work with black-box functions. These methods, however, suffer from the curse of dimensionality, and may become intractable for high-dimensional problems.

In terms of performance, it is worth noting that while the first class of methods permits to find the optimum only locally, the second one can be used also for global optimization. According to the *search goal* of the algorithm, we refer to:

- **Local optimization**, if the goal is to find a best feasible design point in a neighbourhood of the initial point;
- **Global optimization**, when it is aimed at finding the best feasible objective function value over the whole design space.

The capability of derivative-free algorithms to find the global optimum is a feature that is obviously considered attractive in industrial applications. Moreover, since they do not rely on smoothness assumptions in the objective or in the constraints, as gradient-based methods do, they are also suitable for discrete problems. Nevertheless, global optimization usually requires a much larger number of functional evaluations than the local counterpart, and this number grows superlinearly with the number of design parameters. In addition, the convergence rate is much slower for derivative-free optimizers. Therefore, finding the global optimum rather than the local one will be more expensive in almost all cases. In many large-scale engineering problems, however, the optimization stopping criterion is often dictated by design costs and scheduling: therefore, for high-dimensional problems with expensive objective evaluations, it is common practice to assume a sub-optimal solution as converged point, provided that it introduces a valuable improvement with respect to the baseline.

Nevertheless, for our target applications, we prefer to focus on global optimization, given its non-intrusiveness and since it is more robust and versatile than the local

counterpart. Among the derivative-free global methods, we employ two common methods based on response surface surrogates, as described in the following sections.

7.1.1 Surrogate-based global methods

Surrogate-based global minimization is based on the construction of a global surrogate over a set of sample points, which is used by the optimizer to find the optimum. The method is often used within an iterative scheme, that consists of globally updating the surrogate, once new high-fidelity evaluations are available. At every iteration, minimizers of the surrogate are found, and then a subset of these points is evaluated with the high-fidelity model. In the next iteration, the new points are added to the set used to build the surrogate. The surrogate tends to become more accurate at each iteration, although there is no guarantee of convergence (see Adams et al. (2014a)). Nevertheless, it is particularly suited for multi-objective optimizers based on genetic algorithms, since it allows us to select points directly on the surrogate Pareto frontier. In other terms, the approach does not necessitate to create a set of surrogates, one for each objective function, as happens for instance in trust-regions methods, leading to a significant reduction of the number of points initially required.

The approximation of the response surface is given by a Gaussian Process (GP) spatial interpolation, also known as Kriging (see Cressie (1993)). Developed originally in geostatistics, the main idea behind the Kriging interpolation method is to use the spatial correlation between the input sampling points to interpolate the values in the parameter space. With respect to other surrogate models, GP models introduce not only a prediction of the objective function value, but also an estimate of the prediction variance, providing an indication of the uncertainty in the model.

Given an initial set of N training points, $\boldsymbol{\Xi} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N]$, the Kriging emulator $G(\boldsymbol{\mu})$ of the true response function, representing an estimated distribution of the unknown true surface, can be expressed in terms of a given deterministic component plus a stochastic correction

$$G(\boldsymbol{\mu}) = \mathbf{h}(\boldsymbol{\mu})^\top \boldsymbol{\beta} + Z(\boldsymbol{\mu}) \ ,$$

where $\mathbf{h}(\boldsymbol{\mu})$ and $\boldsymbol{\beta}$ represent the vector of N_β trend basis function and the corresponding generalized least-squares coefficients estimates, respectively, whereas $Z(\boldsymbol{\mu})$ is a stationary GP. Commonly, the trend function is chosen as a general polynomial function whose coefficients are determined through least-squares regression (universal Kriging), while $Z(\boldsymbol{\mu})$, also known as *error model*, introduces a correction to the trend function in order

to guarantee that the model interpolates the data points with null uncertainty. By construction, $Z(\boldsymbol{\mu})$ has zero mean, and the process covariance at two arbitrary points $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ is given by

$$\mathrm{Cov}\left[Z(\boldsymbol{\mu}_i, Z(\boldsymbol{\mu}_j)\right] = \sigma_Z^2 r(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j),$$

where σ_Z^2 is the unadjusted process variance and $r(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ is a correlation function, generally chosen among powered-exponential, Matérn and Cauchy families, as pointed out in Adams et al. (2014b). In this work, the squared exponential (Gaussian) correlation function is selected:

$$r(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \exp\left(-\sum_{k=1}^{P} \theta_k |\mu_{i,k} - \mu_{j,k}|^2\right) \ ,$$

with $\mu_{i,k}$ and $\mu_{j,k}$ denoting the k^{th} components of the P-dimensional vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$. The terms θ_k, instead, are adjustable correlation parameters, depending on the correlation lengths L_k (i.e. the standard deviations in the Gaussian distribution) as follows:

$$\theta_k = \frac{1}{2L_k^2} \quad > 0.$$

Thus, a large θ_k is representative of a small correlation length and vice versa.

Given a finite number of sample points, N, the suitable value of the vector $\boldsymbol{\beta}$ is usually not known a priori, but can be modelled in terms of a distribution of possible values. Assuming no preliminary knowledge of this distribution, i.e. under the so-called *vague prior* hypothesis, the maximum likelihood value of the coefficients is computed solving the generalized least-square problem:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^\top \mathbf{R}^{-1} \mathbf{H}\right)^{-1} \left(\mathbf{H}^\top \mathbf{R}^{-1} \mathbf{f}\right) \ ,$$

where $\mathbf{H}$ is the matrix containing the trend basis functions at all points in $\boldsymbol{\Xi}$, i.e. $H_{i,k} = h_k(\boldsymbol{\mu}_i)$, $\mathbf{f}$ denotes the vector of true responses and $\mathbf{R}$ is the correlation matrix for all the data point in the sample design, defined as

$$R_{i,j} = R_{j,i} = r(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = r(\boldsymbol{\mu}_j, \boldsymbol{\mu}_i).$$

The terms in $\mathbf{R}$ are therefore dependent on the vector of correlation parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_P]^\top$, that is determined via a Maximum Likelihood Estimation (MLE) procedure. MLE consists in searching the $\boldsymbol{\theta}$ set that maximises the plausibility of the Kriging model prediction, given the input data. Numerically, this is attained by maximising the natural

logarithm of the likelihood (i.e. the probability of observing the true response values $\mathbf{f}$ given $\mathbf{R}$), that under the vague prior assumption can be written as:

$$\log(\text{lik}(\boldsymbol{\theta})) = -\frac{1}{2}\left((N - N_\beta)\left(\frac{\hat{\sigma}_Z^2}{\sigma_Z^2} + \log(\sigma_Z^2) + \log(2\pi)\right) + \log(\det(\mathbf{R})) + \log(\det(\mathbf{H}^\top \mathbf{R}^{-1}\mathbf{H}))\right). \tag{7.1}$$

By substituting into Equation (7.1) the MLE of the unadjusted variance, i.e.

$$\hat{\sigma}_Z^2 = \frac{(\mathbf{f} - \mathbf{H}\hat{\boldsymbol{\beta}})^\top \mathbf{R}^{-1}(\mathbf{f} - \mathbf{H}\tilde{\boldsymbol{\beta}})}{N - N_\beta},$$

it follows that the maximization problem is equivalent to find the minimum of the functional

$$\mathcal{F}(\boldsymbol{\theta}) = \log(\hat{\sigma}_Z^2) + \frac{\log(\det(\mathbf{R})) + \log(\det(\mathbf{H}^T \mathbf{R}^{-1}\mathbf{H}))}{N - N_\beta},$$

where the determinants can be efficiently computed through Cholesky factorization. After evaluating the correlation parameters, $\mathbf{R}$ and $\hat{\boldsymbol{\beta}}$ are determined. The estimated response of the model at an arbitrary point is then given by the relation:

$$f_G(\boldsymbol{\mu}) = \mathrm{E}\left[G(\boldsymbol{\mu})|\mathbf{f}\right] = \mathbf{h}(\boldsymbol{\mu})^\top \hat{\boldsymbol{\beta}} + \mathbf{r}(\boldsymbol{\mu})^T \mathbf{R}^{-1}(\mathbf{f} - \mathbf{H}\hat{\boldsymbol{\beta}}), \tag{7.2}$$

where $\mathbf{r}(\boldsymbol{\mu})$ is the correlation function vector, defined as $r_k = (\boldsymbol{\mu}) = r(\boldsymbol{\mu}, \boldsymbol{\mu}_k)$, with $\boldsymbol{\mu}_k \in \boldsymbol{\Xi}$.

It is worth noting that Equation (7.2) represents a best linear unbiased estimator of the unknown true function, that interpolates the data as long as $\mathbf{R}$ is numerically non-singular. A known issue in determining the optimal θ_k for a GP is that the condition number of the correlation matrix $\mathbf{R}$ tends to significantly increase when many sampling points cluster in a small region, resulting in a poor predictive capability of the model. In addition, input spaces with large dimensions will result in a large $\mathbf{R}$ matrix, that must be inverted several times in order to determine the model parameters, making the process computationally expensive. For the applications of interest, matrix ill-conditioning is unlike to happen, especially at early optimization stages, when the sampling of the parameter space is rather coarse and well-spaced, but some problems may arise when the algorithm starts to converge, introducing a significant amount of duplicate information. In order to avoid this, two strategies are commonly employed. In the first one, some noise (often referred to as nugget) is added to the matrix diagonal to improve the conditioning

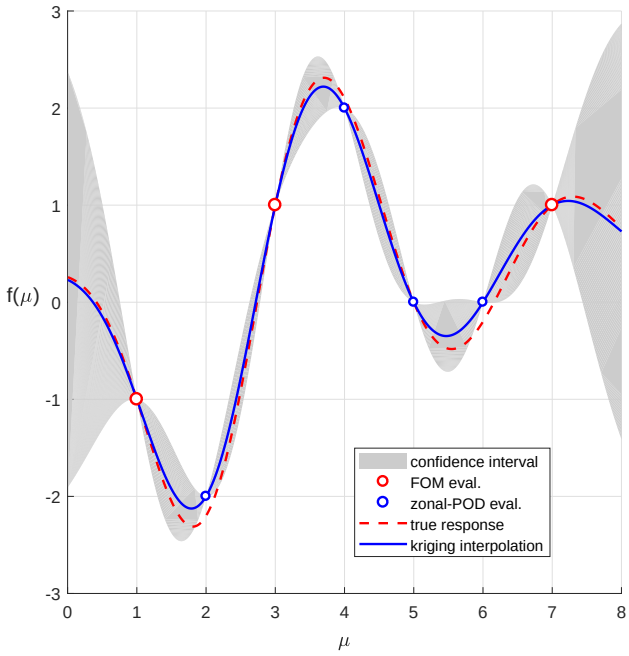


Fig. 7.1 Example of data interpolation by Kriging, using both FOM and zonal-POD simulations.

and account for measurement error in the input data, whereas the second one consists in choosing an optimal subset of points. Such choice can be made a priori, through a greedy heuristic, or within the maximum likelihood optimization loop, by discarding the points that contain the least unique information. Further detail can be found in Adams et al. (2014b).

The Surrogate-based Global Optimization (SBGO) strategy, as employed in the following sections, is outlined in Algorithm 8. Note that the initial response outputs are evaluated through the FOM, whereas during the optimization loop the zonal-POD approach is employed: the GP process is then built over solutions of mixed accuracy, but given the consistency of the proposed ROM, we have that the full-order and reduced order response surfaces coincide for the points belonging to the initial database, since $f(\boldsymbol{\mu}_k) = f^\star(\boldsymbol{\mu}_k) \quad \forall \boldsymbol{\mu}_k \in \boldsymbol{\Xi}$ (see Figure 7.1). As a consequence, the *true* response surface that the Kriging model tries to emulate maintains its smoothness.

Algorithm 8 SBGO algorithm

1: compute an initial distribution of $\boldsymbol{\mu}_k \in \boldsymbol{\Xi}$
2: evaluate the response outputs $f(\boldsymbol{\mu}_k) \quad \forall \boldsymbol{\mu}_k \in \boldsymbol{\Xi}$
3: **while** *convergence* = false **do**
4: determine $\boldsymbol{\theta}$ through MLE
5: evaluate $f_G(\boldsymbol{\mu})$
6: $\boldsymbol{\mu}_{\text{new}} \leftarrow \arg\min_{\boldsymbol{\mu}} f_G(\boldsymbol{\mu})$
7: evaluate the new response output $f^{\star}(\boldsymbol{\mu}_{\text{new}})$
8: update the training set and the responses set
9: **end while**

In terms of performance, the method relies on the choice of the initial population, used to start the algorithm, and on the criteria employed to sample the promising regions of the parameter space, the so-called in-fill sample criteria (see for instance Forrester and Keane (2009) for a complete review of the problem and Liu et al. (2012) for a comparison of infill techniques). For this work, we adopt a popular criterion, i.e. minimization of the surrogate prediction, whereas a more efficient strategy is illustrated in the following section.

7.1.2 Efficient Global Optimization

Efficient Global Optimization (EGO) is a surrogate-based global optimization technique originally proposed by Jones et al. (1998) for expensive black-box functions, as in the case of simulation driven optimizations. Starting from a sample of true simulations, a GP approximation of the objective function is build during each EGO iteration and the new point in the parameter space is chosen by maximising an Expected Improvement Function (EIF), defined as the expectation that any point in the design space will enhance the current best, in terms of the GP model predicted variances and expected values. The rationale behind the algorithm is to provide balance between exploiting regions with good solutions and exploring regions where the prediction uncertainty is high.

As in the previous section, the true response function is expressed as $G(\boldsymbol{\mu}) = \mathbf{h}(\boldsymbol{\mu})^T\boldsymbol{\beta} + Z(\boldsymbol{\mu})$, with the difference that the trend of the model is assumed constant for simplicity, whereas the trend coefficients $\boldsymbol{\beta}$ are selected as the mean of the input data. The process variance is determined through the MLE procedure, whereas the correlation function is assumed to be a squared-exponential (see Adams et al. (2014b)).

Thus, the model expected values f_G and variance σ_G^2 at the generic point $\boldsymbol{\mu}$ can be computed as:

$$f_G(\boldsymbol{\mu}) = \mathbf{h}(\boldsymbol{\mu})^T\boldsymbol{\beta} + \mathbf{r}(\boldsymbol{\mu})^T\mathbf{R}^{-1}(\mathbf{f} - \mathbf{H}\boldsymbol{\beta}) \ ,$$
$$\sigma_G^2(\boldsymbol{\mu}) = \sigma_Z^2 - \left[\mathbf{h}(\boldsymbol{\mu})^T\mathbf{r}(\boldsymbol{\mu})^T\right]\begin{bmatrix}\mathbf{0} & \mathbf{H}^T\\ \mathbf{H} & \mathbf{R}\end{bmatrix}\begin{bmatrix}\mathbf{h}(\boldsymbol{\mu})\\ \mathbf{r}(\boldsymbol{\mu})\end{bmatrix} \ ,$$

where the vector $\mathbf{r}(\boldsymbol{\mu})$ contains the covariation terms between $\boldsymbol{\mu}$ and every point $\boldsymbol{\mu}_k \in \boldsymbol{\Xi}$ and $\mathbf{R}$ is the $N \times N$ covariation matrix for the training points, both defined as in Section 7.1.1, the vector $\mathbf{f}$ contains the training points outputs, and $\mathbf{H}$ is a $N \times 1$ matrix whose rows are the trend function evaluated at the training points, i.e. $\mathbf{h}(\boldsymbol{\mu}_k)^\top$.

At every point of the design space, the GP prediction $G(\boldsymbol{\mu})$ is then a normal Gaussian distribution $N\left[f_G(\boldsymbol{\mu}); \sigma_G^2(\boldsymbol{\mu})\right]$. Thus, the following definition for the EIF can be derived:

$$\text{EIF}(G(\boldsymbol{\mu})) := E\left[\max\left(G^{\text{best}} - G(\boldsymbol{\mu}), 0\right)\right] \ ,$$

where G^{best} is the current best solution on the training set. The expected value is then computed integrating over the distribution $G(\boldsymbol{\mu})$, obtaining the following analytical formulation:

$$\text{EIF}(G(\boldsymbol{\mu})) = (G^{\text{best}} - f_G(\boldsymbol{\mu}))\Phi\left(\frac{G^{\text{best}} - f_G(\boldsymbol{\mu})}{\sigma_G(\boldsymbol{\mu})}\right) + \sigma_G(\boldsymbol{\mu})\phi\left(\frac{G^{\text{best}} - f_G(\boldsymbol{\mu})}{\sigma_G(\boldsymbol{\mu})}\right). \quad (7.3)$$

Generally speaking, from Equation (7.3) it is possible to note that the points maximising the EIF can be either the ones with good expected values or those with greater variance, providing a compromise between the exploitation of the parameter space and its exploration.

The basic steps of the EGO algorithms are summarised in the following pseudo-code (Algorithm 9). As in Section 7.1.1, the outputs are computed through both the FOM (initial distribution) and the hybrid ROM/FOM (EGO iterations).

Algorithm 9 EGO algorithm

1: compute an initial distribution of $\boldsymbol{\mu}_k \in \boldsymbol{\Xi}$
2: evaluate the response outputs $f(\boldsymbol{\mu}_k) \quad \forall \boldsymbol{\mu}_k \in \boldsymbol{\Xi}$
3: set $\text{EIF}(\boldsymbol{\mu}) = 1$
4: **while** $\text{EIF}(\boldsymbol{\mu}) \leq 0 + \epsilon$ **do**
5: build $G(\boldsymbol{\mu})$
6: evaluate G^{best}
7: compute EIF
8: $\boldsymbol{\mu}_{\text{new}} \leftarrow \arg\max_{\boldsymbol{\mu}} \text{EIF}$
9: evaluate the new response output $f^{\star}(\boldsymbol{\mu}_{\text{new}})$
10: update the training set and the responses set
11: **end while**

When dealing with constrained optimization problems, if the constraints are deterministic, they can be handled by introducing a merit function and using the EGO method straightforwardly. Otherwise a new formulation of the algorithm, which includes the definition of an Expected Feasibility Function (EFF), is required (see Adams et al. (2014b) for further details).

With respect to the standard Kriging model, the EGO algorithm presents a more efficient in-fill criterion, i.e. the maximization of the EIF. Nevertheless, for large design spaces and consequently complicated objective functions, the real global optimum is often difficult to obtain, especially when the deformation range varies a lot. This issue can be mitigated using a concept similar to the trust-region method, as demonstrated in Han et al. (2018), where a dynamic and adaptive search of the design space for the maximum of the EIF is performed, improving the optimizer performance in a problem of drag minimization for a full aircraft configuration with 80 design parameters.

7.2 Numerical tools

The optimization procedures are driven by the algorithms provided in the Dakota[1] toolkit, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis developed at Sandia National Laboratories (see Adams et al. (2014a)). Dakota contains a variety of iterative analysis methods and algorithms, in order to perform parameter studies, design of experiments, uncertainty quantification, calibration and optimization. Such capabilities

[1] https://dakota.sandia.gov/

may be used individually or as fundamental blocks in more advanced strategies, including surrogate-based optimization, optimization under uncertainty or mixed aleatory/epistemic uncertainty quantification. All the tools are designed to exploit High Performance Computing, providing a flexible interface towards numerical simulation codes.

Chapter 8

Industrial Applications

Part of the work described in this chapter has been previously submitted for publishing in Scardigli et al. (2019) and Salmoiraghi et al. (2018).

The results of the integrated approach proposed in the previous chapters are shown for two cases of interest, namely the front-bumper optimization of a three-dimensional car model (Section 8.1) and a sailing boat thrust maximization problem (Section 8.2).

8.1 DrivAer

In this section, we present the aerodynamic shape optimization of the front bumper of the DrivAer[1] car model. The DrivAer model 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 two medium-size cars (see Heft et al. (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 et al. (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.

[1] http://www.aer.mw.tum.de/en/research-groups/automotive/drivaer/

For the case under investigation, 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, characterized by separation, recirculation bubbles and unsteady wakes (Hucho (1987)). Details about the numerical setup are provided in the following sections.

8.1.1 FOM setup

A three-dimensional hex-dominant mesh of about 15×10^6 cells is generated around the car model by the `snappyHexMesh` utility, introducing a symmetry plane in the longitudinal direction, as shown in Figure 8.1. The computational domain Ω is 50 m long, with a height H of 12 m and a width W of 10 m, chosen in order to guarantee the same blockage ratio, defined in terms of the car frontal area A_f as

$$b = \frac{A_\mathrm{f}}{HW} \approx 8\%,$$

as the one considered by Heft et al. (2012) for the experimental investigation of the same problem.

In terms of boundary conditions, we impose inflow and outflow conditions at the boundaries corresponding to the wind tunnel inlet and outlet sections (Figure 8.1), whereas the ground is modelled as a moving wall with tangential velocity equal to the inlet free-stream velocity $U_\infty = 16$ m/s. The Reynolds number of the simulations, based on the free-stream speed and on the car reference length $l_\mathrm{ref} = 4.61$ m, is equal to 4.87×10^6.

The flow solution is obtained by solving the steady RANS equations in Ω, with the Spalart-Allmaras turbulence model, as presented in Section 3.2.5, introducing wall functions to describe the near-wall flow. The target y^+ for this application is set equal to 35.

The output of interest for the simulations is the aerodynamic drag coefficient, defined as

$$C_\mathrm{x} = \frac{\int_{S_\mathrm{car}} (-p\mathbf{n} + \boldsymbol{\tau}) \cdot \mathbf{e}_x \, \partial S}{\frac{1}{2}\rho U_\infty^2 A_\mathrm{f}} \tag{8.1}$$

where $\mathbf{n}$ represents the outward-pointing surface normal, $\mathbf{e}_x$ is the versor in the longitudinal direction, ρ the air density, and U_∞, A_f previously defined.

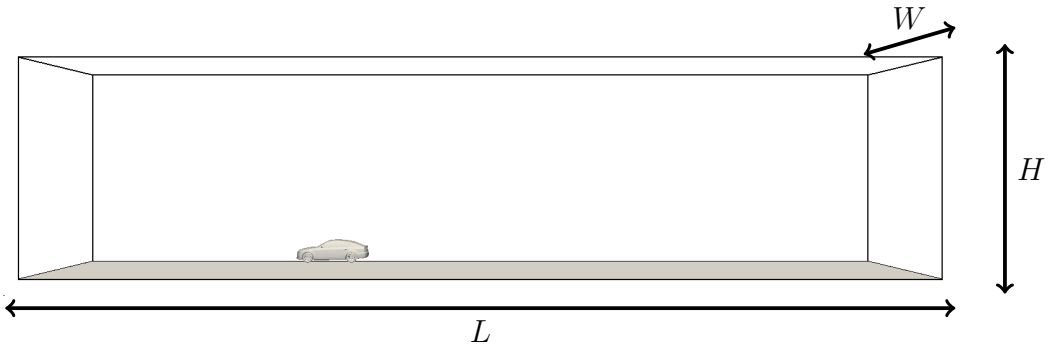


Fig. 8.1 Geometry and computational domain of the DrivAer car model. The moving ground is depicted in grey.

Undoubtedly, the physics of the problem is unsteady. The main focus of this simulation, however, is the aerodynamic performance of the car, that is usually expressed in terms of the average coefficients. A common practice in the industrial framework, is to employ nevertheless the steady RANS solver, and to obtain the actual coefficients from the field averaged over the SIMPLE iterations. In this case, we choose to average the quantities over the last 1000 solver iterations, as shown in the example of Figure 8.2. Even if the field oscillations are a numerical artefact and do not correspond to the physical counterpart, this approach leads to reasonable results: compared with experimental data (see Heft et al. (2012)), for instance, the numerical setup presented above produce an error smaller than 5% on the average C_{x}, which is considered acceptable for our purposes.

8.1.2 Geometry parameterization

In this application, we employ the FFD parameterization technique directly on the computational mesh, using the PyGEM tool. The mesh region close to the car front bumper is wrapped in a lattice of $6\times3\times4$ control points, as shown in Figure 8.3. In order to guarantee a smooth transition between the deformable and fixed regions of the mesh, most of such control points are not allowed to move, except the ones highlighted in red in Figure 8.3, i.e. the points with indices $(n_x, n_y, n_z) = (1,1,1), (1,2,1), (2,1,1), (2,2,1), (3,1,1), (3,2,1)$. This set of points is moved by the same quantity in the longitudinal and vertical directions, resulting in a two-dimensional parameter space, $\boldsymbol{\mu} = (\mu_1, \mu_2)$, whose upper and lower bounds are outlined in Table 8.1. The limits are expressed in percentage of the corresponding lattice dimension, and are chosen in order to avoid intersections between two consecutive points in the control grid.

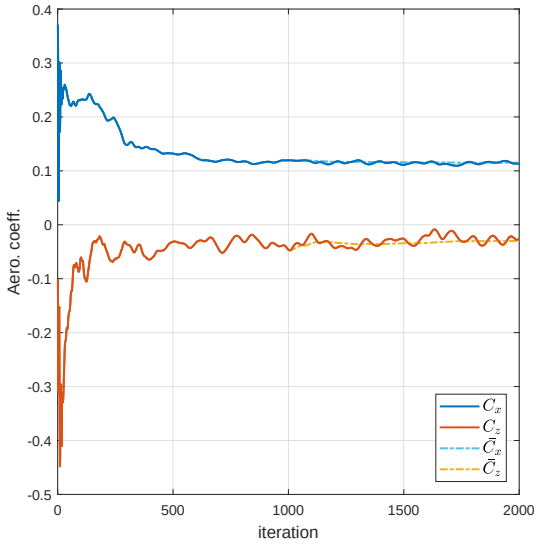


Fig. 8.2 Convergence of the aerodynamic coefficients for the baseline configuration: C_{x}, C_{z} and averaged quantities $\bar{C}_{\mathrm{x}}$ and $\bar{C}_{\mathrm{z}}$.

	μ_1	μ_2
lower bound	-0.18	-0.30
upper bound	0.18	0.30

Table 8.1 Limits of the parameter space.

The chosen parameterization guarantees the overall satisfaction of our set of mesh quality constraints, verified a posteriori for the limit configurations (i.e. the ones corresponding to the vertices of the parameter space). In Figure 8.4, we report the comparison between the original mesh and the resulting deformed mesh for the parameter $\boldsymbol{\mu} = (0.18, 0.30)$, shown on the symmetry plane in the inward-pointing direction.

Since the FFD is applied directly to the volumetric mesh, we have a set of topologically equivalent computational domain and there is no need to rebuild the mesh for the others configurations, resulting in a noticeable simplification of the model reduction and optimization workflow, as highlighted in the following sections.

8.1.3 ROM setup

For each physical variable involved, i.e. p, $\mathbf{u}$ and the viscosity-like turbulent quantity $\tilde{\nu}$, a separate POD basis is computed, using as forcing term of Equation (5.11) the

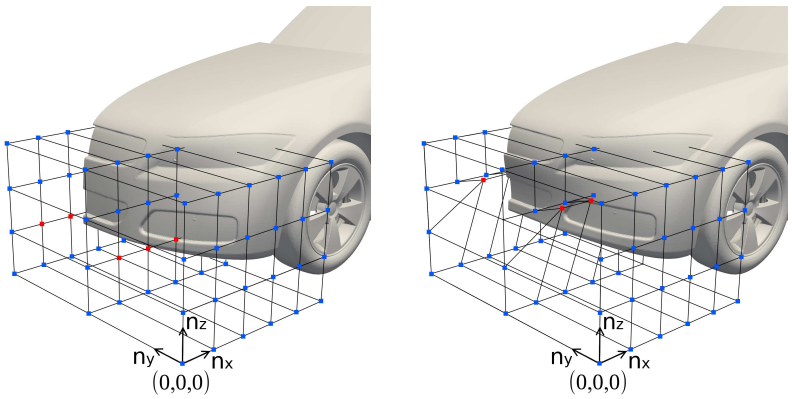


Fig. 8.3 FFD lattice and a possible deformation (red control points): $(\mu_1, \mu_2) = (0.18, 0.30)$.

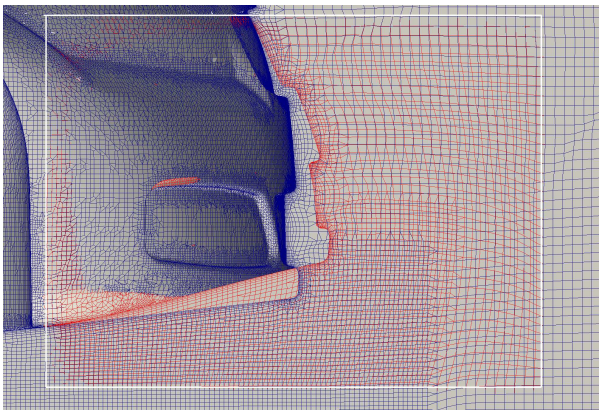


Fig. 8.4 Mesh morphing of the DrivAer front bumper: original mesh (blue lines) vs modified mesh (red lines) for $(\mu_1, \mu_2) = (0.18, 0.3)$. The white box identifies the deformed region.

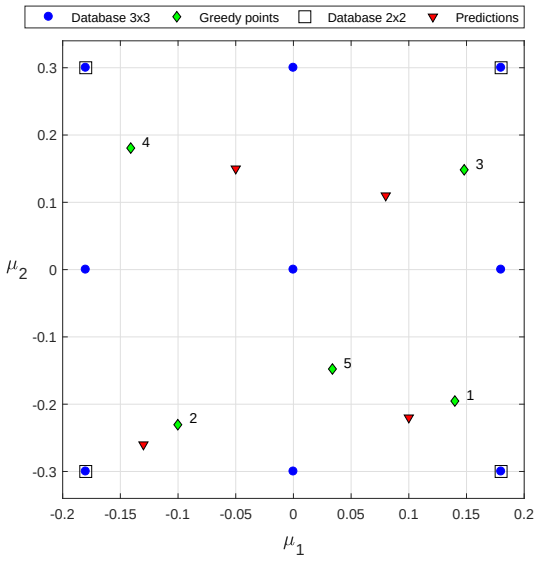
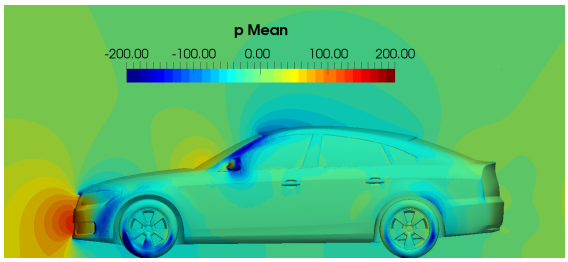


Fig. 8.5 Sampling of the parameter space: initial points (blue) and greedy points (green).

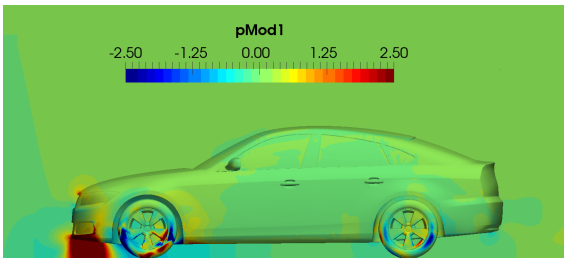
ensemble-averaged fields. As for the aerodynamics coefficients, the solution snapshots are also averaged over the last 1000 solver iterations.

The snapshot used to train the POD model are selected through the pGA-CCVT algorithm outlined in Section 6.3. As shown in Figure 8.5, we start from a uniform distribution of $N = 9$ points and then 5 more points are added iteratively, computing the error estimator using only the restriction of the pressure field to the surface of the model, i.e. $\Delta_m(\boldsymbol{\mu_i}) = \|p_{\text{wall}}(\boldsymbol{\mu_i}) - p^{\star}_{\text{wall}}(\boldsymbol{\mu_i})\|$, where p_{wall} represents the FOM solution and $p^{\star}_{\text{wall}}$ its POD approximation. The choice is justified by the fact that we are interested only in integral quantities over the car surface, and the pressure drag is the main drag component for this kind of applications. The most critical region appears to be the south-east quadrant of the parameters space, but the algorithm also explores regions with fewer points.

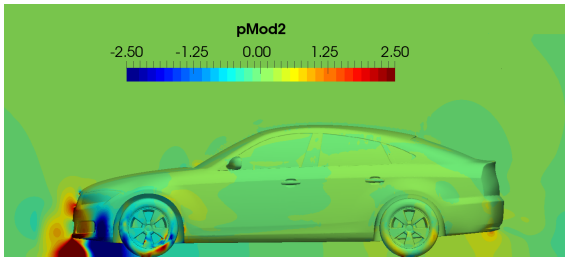
Then we extract the POD bases that will be used in the optimization computations. The pressure forcing term and the first three modes are reported in Figure 8.6, whereas the eigenvalues for all the physical variables are presented in Figure 8.7. The RIC analysis shows that for the velocity and pressure bases, all 13 modes are required in order to capture the 99.99% of the energy, and therefore the complete bases will be employed in the following. The contribution of the modes is localized in the front region of the model, and in particular near the bumper. Nevertheless, as expected, also the underbody



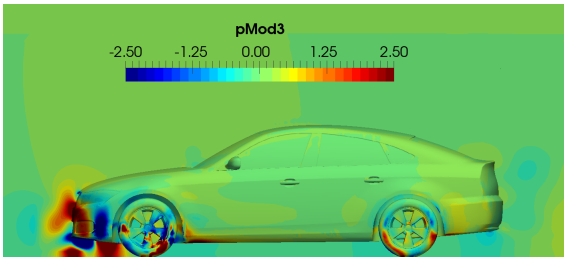
(a) $\bar{p}$.



(b) φ_1.



(c) φ_2.



(d) φ_3.

Fig. 8.6 Average pressure field and three more energetic POD modes on the DrivAer symmetry plane.

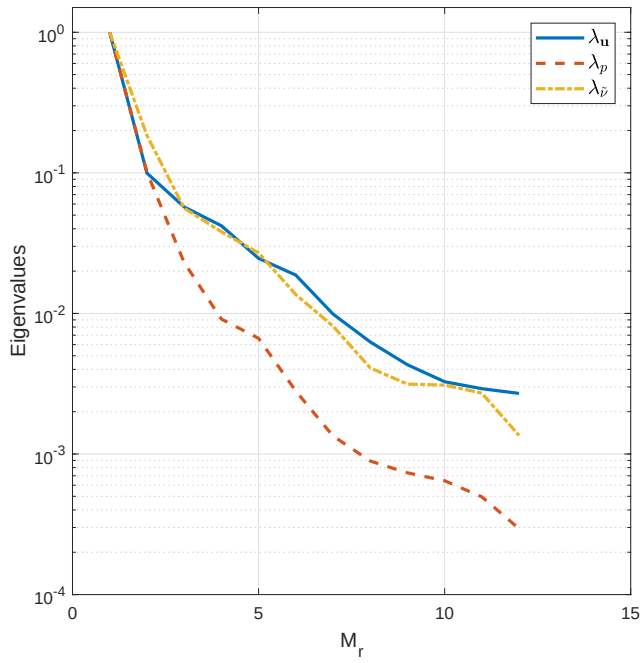


Fig. 8.7 POD eigenvalues for velocity, pressure and turbulent quantity, scaled with the corresponding maximum value.

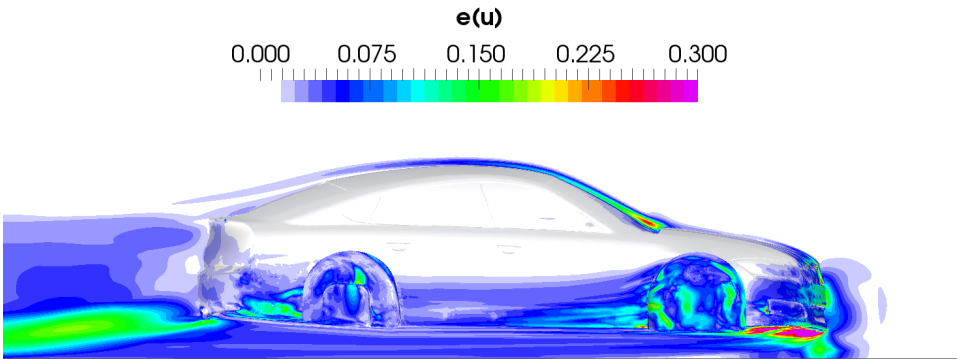


Fig. 8.8 Leave-one-out error distribution for the velocity field.

flow is affected by the geometry changes in the front and, consequently, the turbulent wakes in the back of the car.

Such trend is also confirmed from the error maps that we compute through the leave-one-out cross-validation method (see Section 6.1), in order to identify a suitable domain decomposition. The results of this analysis are shown in Figure 8.8, where we report the velocity error map evaluated using the snapshots previously computed. The $\Omega_1(\boldsymbol{\mu})$ domain is then chosen by imposing an error threshold on the velocity $\sigma_R = 0.25U_\infty$: as a rule of thumb in the selection of such subdomain, we make sure to include all the cells in the prismatic layers and to position the Γ_1 interface sufficiently away from the wheels. These requirements are due to the performance of the full-order solver, which may not be able to handle very irregular boundaries in critical regions such as boundary layers and in proximity of rotating components. The discretization of the resulting domain, shown in Figure 8.9, includes only $2 \cdot 10^6$ cells (with respect to the $15 \cdot 10^6$) and leads to a reduction in the computational time of a factor 8, considering that the FOM requires around 160 cpuh, whereas the zonal-POD model only 20. In this application, given that both the hybrid and full-order solutions need to be averaged over a thousand solver iterations, the acceleration due to the flow initialization is less relevant, and the achievable speed-up is mainly limited to the geometrical contribution.

In order to assess the predictive capacity of the hybrid approach, four additional out-of-sample configurations (marked in red in Figure 8.5) are tested, varying the size of the database. The results of this analysis are illustrated in Figure 8.10, where we plot the average error (among the four predictions) on the two drag contributions, i.e. $C_{\mathrm{x},1}$ and $C_{\mathrm{x},2}$. When considering only the average fields ($M_r = 0$), both errors are quite high: around 2% in the outer region and more than 5% in the inner domain. This condition significantly improves when using the databases with $N = 4$ or $N = 9$ snapshots, and the

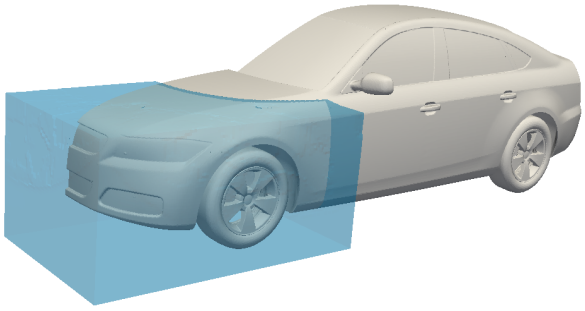


Fig. 8.9 $\Omega_1(\boldsymbol{\mu})$ domain (light blue) for the DrivAer problem, as selected through the leave-one-out method.

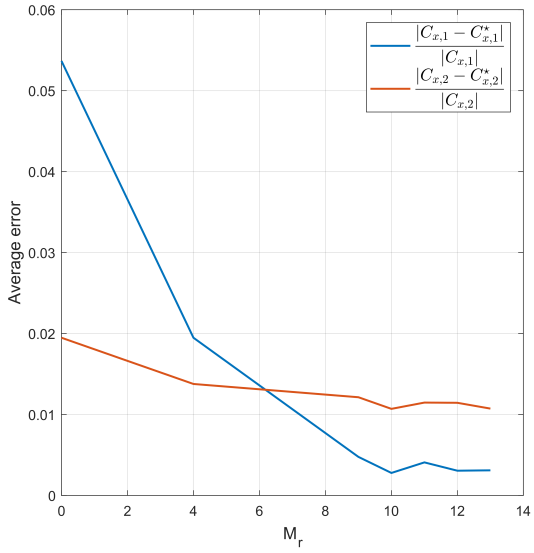
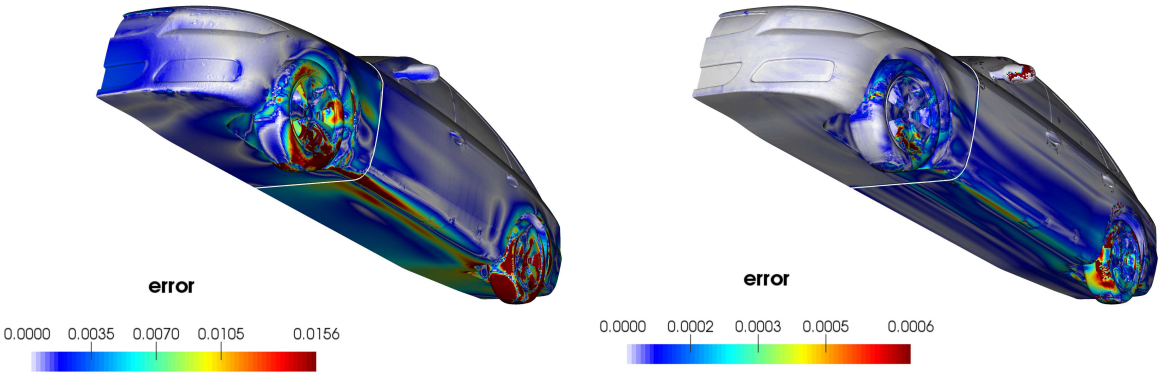


Fig. 8.10 Average error on predictions varying M_r.

corresponding POD bases with $M_r = N - 1$: with $N = 4$ and $M_r = 3$, the error in $\Omega_1(\boldsymbol{\mu})$ drops to 1% and keeps getting better by adding new snapshots to the database. Using all the 13 snapshots, for instance, the average error on the total drag coefficient is equal to 0.4%. All the computed errors are normalized with respect to the corresponding full-order drag coefficient. In Figure 8.11 we report the errors fields between the full-order and hybrid solution of the worst (in terms of accuracy) out-of-sample configuration, evaluated on the surface of the car and adimensionalised with respect to the dynamic pressure $p_{\text{dyn}} = \frac{1}{2}\rho U_\infty^2$. Both the wall shear-stress $\boldsymbol{\tau}_{\text{wall}}$ and pressure p_{wall} are well approximated by the zonal-POD model in Ω_1 (delimited by a white line in the figure), except in the wheel region, where the difference between the FOM solution and the hybrid one become more noteworthy. This is not surprising given the complexity of the flow in such area,



(a) $\frac{|p_{\text{wall}} - p^{\star}_{\text{wall}}|}{p_{\text{dyn}}}$. (b) $\frac{|\boldsymbol{\tau}_{\text{wall}} - \boldsymbol{\tau}^{\star}_{\text{wall}}|}{p_{\text{dyn}}}$.

Fig. 8.11 Surface error fields for the worst out-of-sample configuration: the white line identifies the interface between the FOM and ROM regions.

highly characterised by turbulence and unsteady phenomena. The higher errors, about 2% of the dynamic pressure, can be partially justified by the fact the ROM model is tuned on the overall drag coefficient, as well as the window for the fluctuations average. The approximation of the solution in this region would presumably benefit from a longer statistics accumulation. In addition, it should be noted that the Γ_1 interface is located quite close to the wheel (see Figure 8.9): the pressure distribution, therefore, can be easily influenced by the boundary proximity. The effects of the turbulent wakes originating from wheels and wheelhouses appear evident also considering the error in the outer domain, where the solution is given by a linear combination of the POD modes: the approximation in this region is less accurate, and the errors are localized mainly past the wheels and on the underbody, as also predicted by the leave-on-out error analysis. The combination of such local contributions, however, leads to a 0.8% error on the integral objective function.

8.1.4 Optimization setup

The optimization procedure is driven by the EGO method introduced in Section 7.1.2, using the parameter set previously defined, i.e. $\boldsymbol{\mu} = (\mu_1, \mu_2)$, and the total drag coefficient as objective function. In terms of constraints, the design parameters are bounded in order to guarantee, for every new configuration of the bumper, a vehicle approach angle of at least 20 degrees. The response surface is modelled through a GP, which initially is based on the 13 high-fidelity simulations evaluated with the FOM, during the training

of the POD model. The GP model is then updated using the zonal-POD simulations, whenever a new functional evaluation is required. The point that maximises the EIF is found through the DIRECT algorithm (Jones (2001)) available in Dakota.

The optimization process is stopped either when convergence is attained, or when the number of evaluations exceeds a certain maximum value.

8.1.5 Results

The configuration that minimizes the C_x coefficient of the vehicle is found after 13 full-order and 25 zonal-POD simulations. Compared to a a standard optimization strategy employing global surrogate models, the proposed approach allows us to save around the 58% of the computational time, considering also the offline phase required to train the POD model. In fact, the whole cost of the optimization is roughly 2580 cpuh, against the 6080 cpuh that will be required to reproduce the same process with only full-order evaluations of the optimization functional.

The optimal configuration is found for the parameter set

$$\boldsymbol{\mu}_{\text{best}} = (0.0866, 0.2999),$$

corresponding to a control point displacement of $(x_{\text{cp}}, z_{\text{cp}}) = (0.0693 \text{ mm}, 0.1799 \text{ mm})$, that results in the front-bumper modification highlighted in Figure 8.12. With respect to the baseline configuration, the optimizer tends to lift the bumper, in the attempt to reduce the frontal area exposed to the flow and, consequently, the extent of the stagnation region (see Figure 8.14). As shown in Figure 8.13, this allows the velocity streamlines to make a gentler turn around the car, reducing the flow separation after the bumper edge. This permits to reduce the vehicle aerodynamic resistance by 1.1%, verified on the full-order solution. The comparison between the coefficient values predicted by the ROM and the ones verified through the full-order simulation is reported in Table 8.2. With respect to the objective function, i.e. the drag coefficient, the hybrid model is found to be rather accurate, leading to a relative error $< 0.3\%$. In this case, the predicted drag reduction is slightly greater than the actual value. In terms of lift coefficient, instead, the error on prediction is 2.4%, which is not surprising given that the ROM setup is tuned only on the objective function.

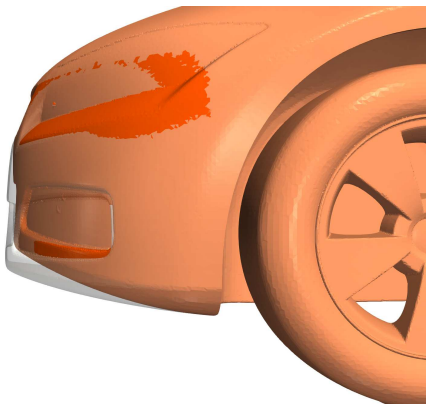
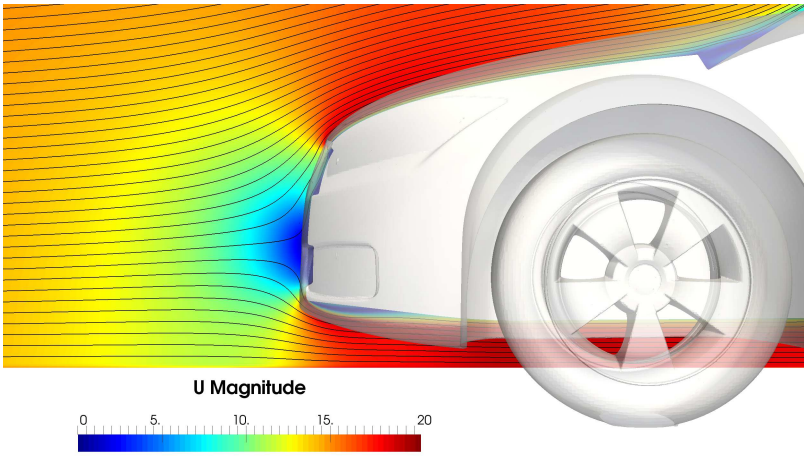
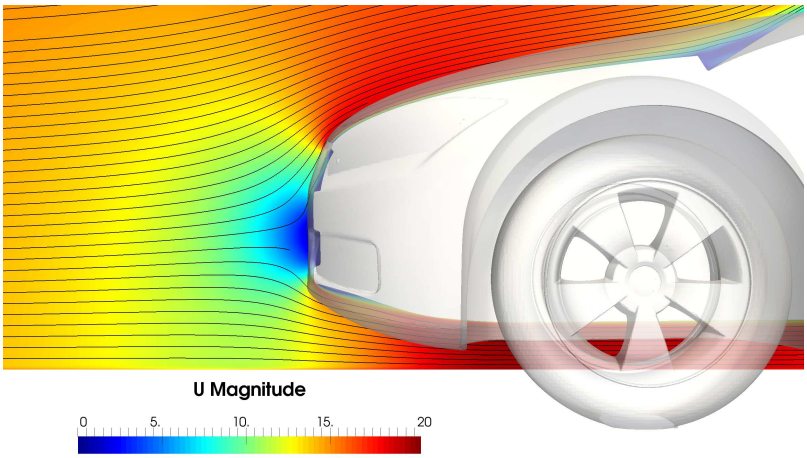


Fig. 8.12 DrivAer optimization: best front-bumper configuration (orange) vs baseline (white).

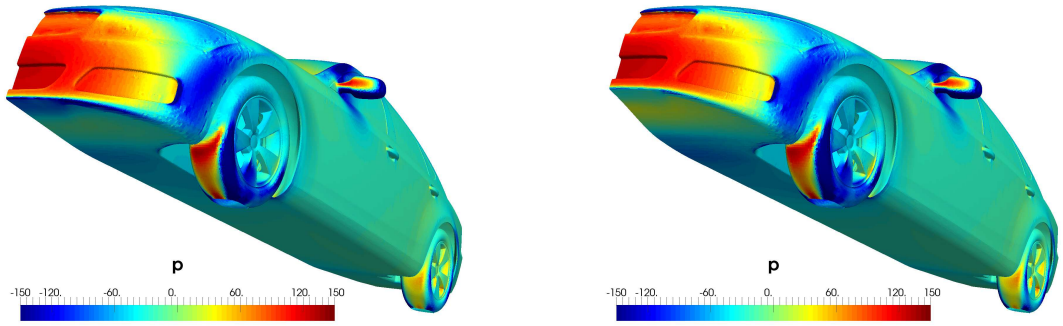


(a) Baseline configuration.



(b) Best configuration.

Fig. 8.13 Velocity streamlines on the symmetry plane for the DrivAer model: front region detail.



(a) Baseline configuration. (b) Best configuration.

Fig. 8.14 Surface pressure field p_{wall} for the DrivAer model.

	FOM	ROM	$\lvert C_i - C_i^\star \rvert / \lvert C_i \rvert$
C_{x}	0.11488	0.11456	0.0028
C_{z}	−0.02623	−0.02686	0.0240

Table 8.2 Aerodynamic coefficients for the optimized configuration: full-order solution, reduced-order solution and relative errors.

8.2 J80 sailing boat

In this application, a combined shape optimization and optimal control problem of a sailing boat is addressed. Specifically, the sailing boat under investigation is equipped with an inflatable device, which controls the shape of the mainsail profile (see Figure 8.15). The introduction of such system is aimed at reducing the flow separation zone due to the mast, allowing the flow to recover faster and improving the efficiency of the mainsail, as shown in Scardigli et al. (2019). The final goal of the problem is to maximize the thrust developed by the sailing system, through the optimization of the inflatable device and the definition of the correct trim angles of the sails, in a condition of close-hauled sailing. In order to simplify the problem, the mainsail and the jib are considered rigid, whereas no wind-wave modelling is introduced.

8.2.1 FOM setup

In Figure 8.16 we report the geometry of the problem and the full-order solution domain. In terms of working conditions, the boat sails at a speed $V_{\text{boat}} = 6.474$ knots with an

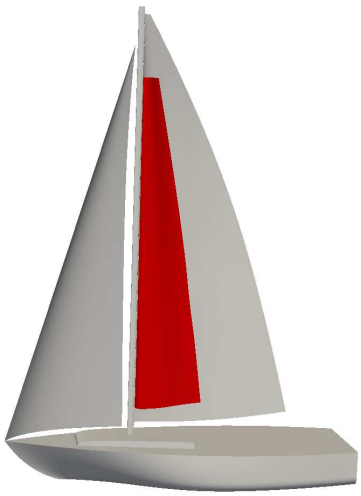


Fig. 8.15 J80 sailing boat with inflatable device on the mainsail (red).

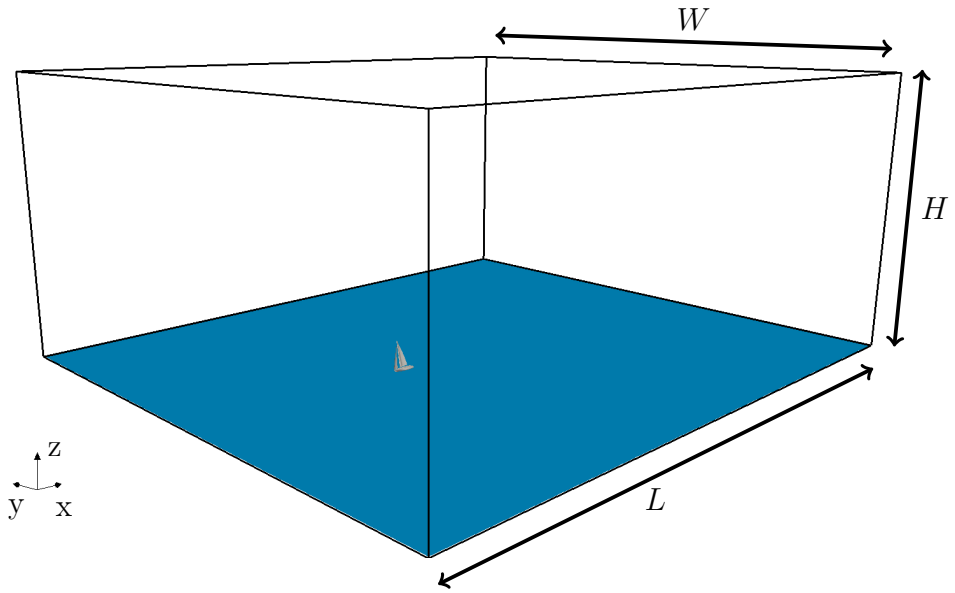


Fig. 8.16 Geometry of the J80 sailing boat model. The sea surface is depicted in blue.

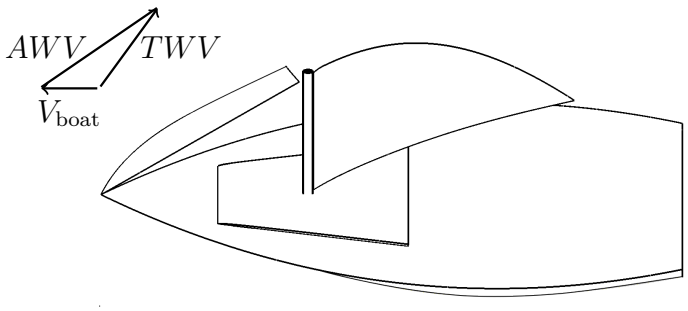


Fig. 8.17 True and apparent wind conditions.

apparent wind velocity $VAW = 11.32$ knots, sensed at a reference height of 10.6 m above water. The bearing respect to the apparent wind is $BAW = 37.7°$, whereas the angle of heel is equal to 10°. Given the apparent wind conditions, true wind conditions are derived from the relations:

$$VTW = \sqrt{(VAW\sin(BAW))^2 + (VAW\cos(BAW) - V_{\text{boat}})^2)}\ ,$$
$$BTW = \arctan\left(\frac{VAW\sin(BAW)}{VAW\cos(BAW) - V_{\text{boat}}}\right)\ ,$$

where VTW and BTW denote the true wind velocity and bearing, respectively (see Figure 8.17). True wind conditions are used to define the free-stream conditions. For this purpose, we need to introduce a fully-developed Atmospheric Boundary Layer (ABL) profile. As addressed for instance in Blocken et al. (2007), the ABL profiles for the mean wind speed, and the turbulent kinetic energy and dissipation rate, are usually simplified by assuming a constant shear-stress distribution along the vertical direction. This leads to the following definitions:

$$\mathbf{u}_{\text{w}}(z) = \frac{\mathbf{u}^{\star}_{\text{ABL}}}{\kappa}\ln\left(\frac{z+z_0}{z_0}\right)\ , \tag{8.2}$$
$$k(z) = \frac{|\mathbf{u}^{\star}_{\text{ABL}}|^2}{\sqrt{C_\mu}}\ ,$$
$$\epsilon(z) = \frac{|\mathbf{u}^{\star}_{\text{ABL}}|^3}{\kappa(z+z_0)}\ ,$$

where $\mathbf{u}^{\star}_{\text{ABL}}$ represents the ABL friction velocity, z the height coordinate and κ the von Karman constant, set to 0.41. z_0, instead, is the aerodynamic roughness length: according to the classification by Davenport and Wieringa (1992), for an open sea landscape we can assume $z_0 = 0.0002$ m. The friction velocity is obtained by substituting in Equation 8.2

the true wind velocity derived from wind readings at the reference height, that leads to

$$\mathbf{u}^{\star}_{\mathrm{ABL}} = \frac{\kappa \mathbf{u}_{\mathrm{w}}(z_{\mathrm{ref}})}{\ln\left(\frac{z_{\mathrm{ref}}+z_0}{z_0}\right)} \ .$$

The ABL velocity profile $\mathbf{u}_w$ is then composed with the boat velocity $\mathbf{u}_{\mathrm{boat}} = (V_{\mathrm{boat}}, 0, 0)$ to obtain the correct profile of the free-stream apparent velocity, which is a warped profile whose intensity and direction at the reference height is equal to the measured apparent velocity.

The boundary conditions are assigned consequently: the sea surface is modelled as a slip wall, with tangent velocity equal to V_{boat}, the boat surfaces are treated as solid walls, under the assumption of rigid sails, whereas free-stream conditions are prescribed in the remaining boundaries.

The atmosphere physical properties are standard sea level conditions at the reference temperature of 20 °C. It follows that the Re number, defined in terms of the VAW and a reference length $l_{\mathrm{ref}} = 4$ m, is set equal to $1.54 \cdot 10^6$.

The computational domain Ω has an extension of 224 m × 196 m × 98 m and it is discretized through a grid of approximately $26 \cdot 10^6$ cells, generated through the `snappyHexMesh` utility, with prismatic boundary layers on both the geometry and the sea surfaces. In Ω the steady RANS equation are solved, employing the realisable $k-\epsilon$ turbulence model (see Section 3.2.5), with wall functions for near-wall treatment. Depending on the jib and mainsail angles, the fluctuations of the flow may become more noticeable in the numerical solution, meaning that the steady solver is no longer capable to reach a converged solution. In principle, in order to capture the time dynamics, a transient solver should be introduced. Nevertheless, since the main focus of the simulation is finding out the aerodynamic forces on the sails, we employ the steady solver even in this eventuality, as it is common practice in similar industrial problems. The actual forces are obtained from the fields averaged over the SIMPLE iterations.

The sailing system performances are assessed in terms of the overall driving force (the developed thrust) and lateral force, defined as:

$$T = -\int_{S_{\mathrm{Jib}}+S_{\mathrm{Main}}} (-p\mathbf{n} + \boldsymbol{\tau}) \cdot \mathbf{e}_x \, \partial S, \qquad F_{\mathrm{L}} = -\int_{S_{\mathrm{Jib}}+S_{\mathrm{Main}}} (-p\mathbf{n} + \boldsymbol{\tau}) \cdot \mathbf{e}_y \, \partial S, \quad (8.3)$$

where $\mathbf{n}$ is the outward-pointing surface normal, and $\mathbf{e}_x$ and $\mathbf{e}_y$ denote the unit vectors in the x and y directions, respectively.

	$\boldsymbol{\theta_M}$	$\boldsymbol{\theta_J}$	$\boldsymbol{p_1}$	$\boldsymbol{p_2}$	$\boldsymbol{p_3}$
lower bound	$-10°$	$-5°$	-10 mm	-5 mm	-5 mm
upper bound	$10°$	$5°$	100 mm	50 mm	50 mm

Table 8.3 Limits of the parameter space.

8.2.2 Geometry parameterization

With reference to Figures 8.19a and 8.18, the following five design variables are chosen to modify the trim of the sails and the device inflation:

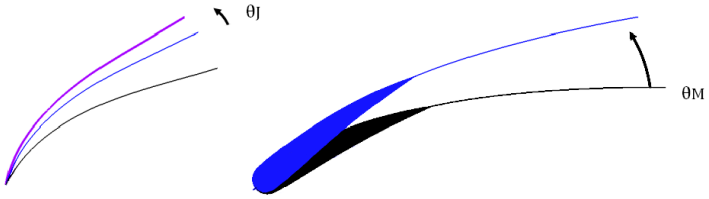
- two rigid rotations, i.e. θ_M, the absolute rotation of the sailing system, and θ_J, the relative rotation of the jib, with respect to the fastening axes;
- three independent parameters controlling the inflatable profile deformation, i.e. p_1, p_2 and p_3.

More specifically, the device shape parameters represent the displacements along the surface normals of the interior control points of the inflatable root contour. In order to enhance a better control over the geometry, the lattice is mapped directly on the surface. In this application it is crucial to control the feasibility of the design by introducing some manufacturing constraints. In particular, the inflatable chambers on the port and starboard surfaces of the mainsail must be symmetric, and the fold that originates where the device is attached to the sail must be preserved, as well as the curvature of the profile at the leading and trailing edges. This latter requirement is attained through the level-set approach introduced in Section 2.2.2: in particular, the deformability field of the points belonging to the geometry surface is weighted by the scalar field depicted in Figure 8.19b. The control points corresponding to the leading and trailing edges of the profile are then fixed, whereas the others move symmetrically with respect to the profile chord, as illustrated in Figure 8.19a. The displacements along the tangent direction to the mainsail are suppressed. Between the head-side control points and the foot-side, a linear relation is imposed, so that $(p_1, p_2, p_3)_{\text{foot}} = 3 \cdot (p_1, p_2, p_3)_{\text{head}}$.

The bounds of the parameters are reported in Table 8.3.

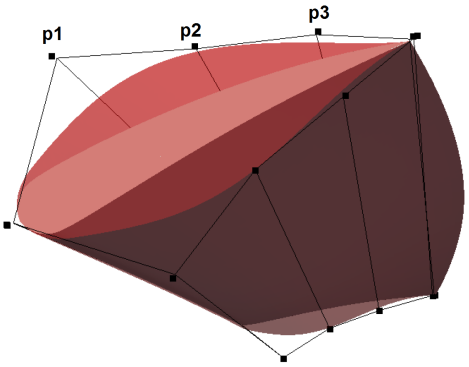
With the current set of parameters, the maximum thickness on the chord of the inflatable device ranges from a minimum value of 10 cm up to a maximum of 30 cm, whereas for the reference configuration the maximum thickness is equal to 14 cm.

(a) Initial configuration of the sail system.

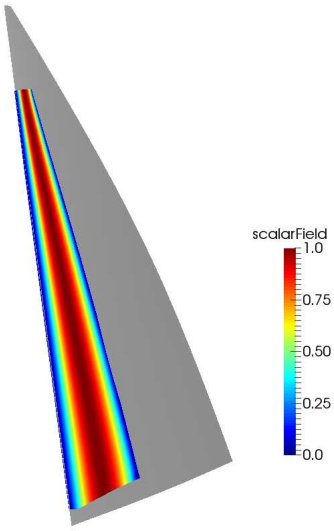


(b) Jib and mainsail rotations.

Fig. 8.18 Trimming of sails around mast and jib axes. Positive rotations in counter-clockwise direction.



(a) Lattice of control points.



(b) Deformation weighting field based on the approximate geodesic distance.

Fig. 8.19 Parameterization of the J80 sailing boat inflatable device.

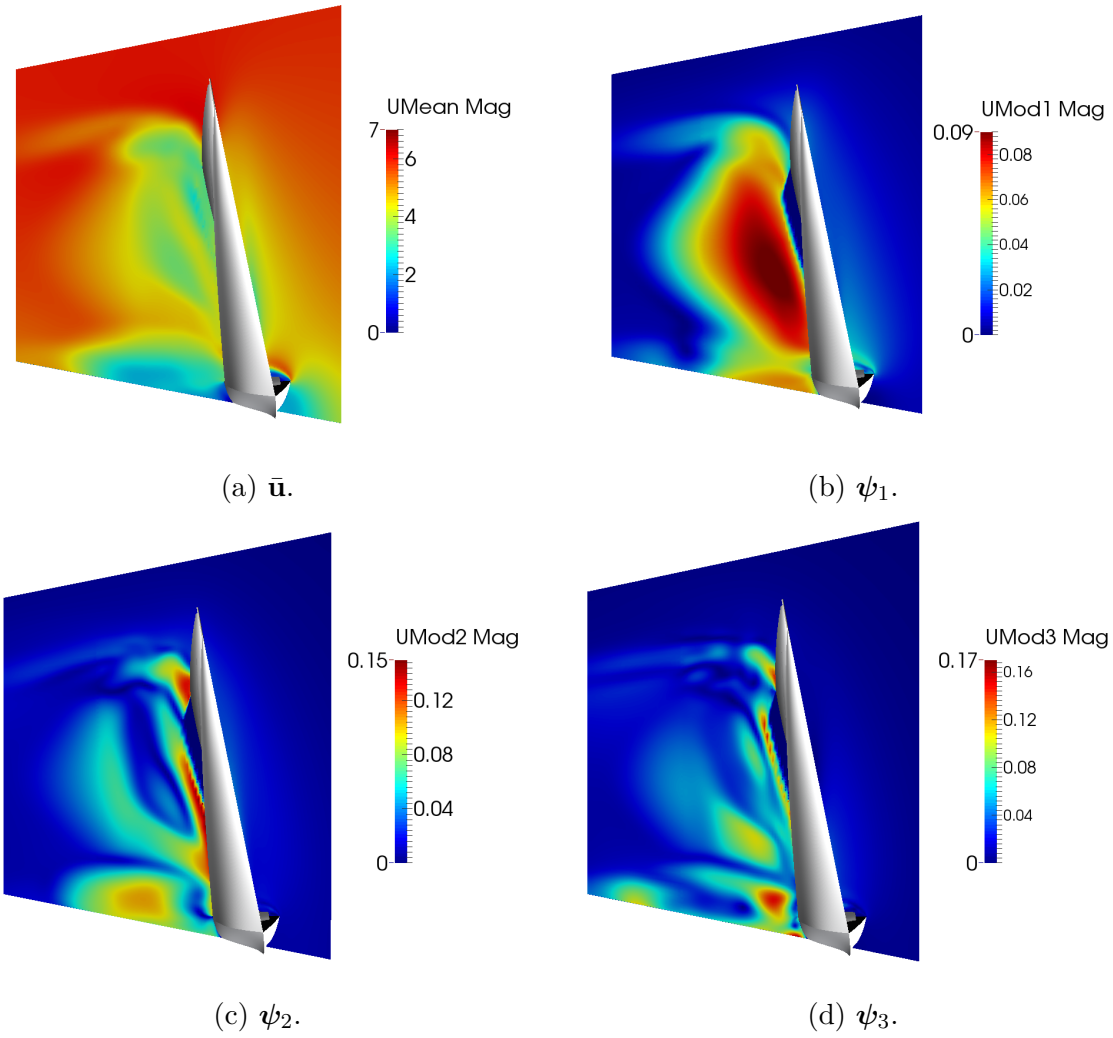
Since the two rigid rotations introduce relevant displacements of the sails trailing edges, we prefer to parameterize the surfaces rather than the computational grid. This means that the mesh is regenerated for each analysed configuration.

8.2.3 ROM setup

In order to build the POD model, an initial database of simulations is created, using as snapshots the fields obtained for each sampling point, averaged over the steady-state solver iterations. In this application, we use only $N = 9$ full-order simulations, including the baseline configuration, corresponding to the parameter set $\boldsymbol{\mu}_1 = (0, 0, 0, 0, 0)$. The remaining sampling points are given by the 2^3 combinations of the limits of the parameters excursion, considering the maximum and minimum sails rotations and the set of shape parameters p_1, p_2 and p_3 that lead to the maximum and minimum inflation of the device. No greedy sampling strategy is employed at this stage.

A separate POD basis is computed for each physical variable (i.e. p, $\mathbf{u}$ and the turbulence quantities k and ϵ). As a consequence of the fact that we pre-process the fields by removing the average among the snapshots, choosing to represent through POD the fluctuations with respect to such average fields, the maximum size of the POD basis will be equal to $N - 1$. The POD modes for the velocity are depicted in Figure 8.20: as expected, the more energetic modes are characterized by bigger flow structures, localized mostly in the turbulent wakes. Since we deal with surface morphing, rather than mesh morphing, the modes are defined in physical space, through the interpolation of the flow snapshots over a reference grid. In terms of the overlapping domain Ω_0 we are forced to remove all those cells swept by both the sails rotations and the inflation of the device, i.e. the ones interested by the geometry variation, where the POD bases cannot be defined. Figure 8.21, instead, shows the eigenvalues decay for the velocity, pressure, turbulent kinetic energy and turbulent dissipation rate fields. To chose a suitable size of the POD basis, we use the RIC criterion, truncating the expansion after the first term that guarantees an energy content greater than the 99.99%. Anyway, due to the relatively small size of the snapshots database, the desired level of information can be achieved only by considering the totality of the modes. Therefore, the zonal-POD computations are carried out with $M_r = 8$ for each POD basis.

The domain decomposition and the position of the fluid interface Γ_1 are chosen accordingly to the algorithm presented in Section 6.2. Thanks to the leave-one-out method, we can obtain a separate error map for each physical variable in Ω_2: an example



(a) $\bar{\mathbf{u}}$. (b) $\boldsymbol{\psi}_1$.

(c) $\boldsymbol{\psi}_2$. (d) $\boldsymbol{\psi}_3$.

Fig. 8.20 Average velocity field and three more energetic POD modes represented in a mid-section of the sailing boat.

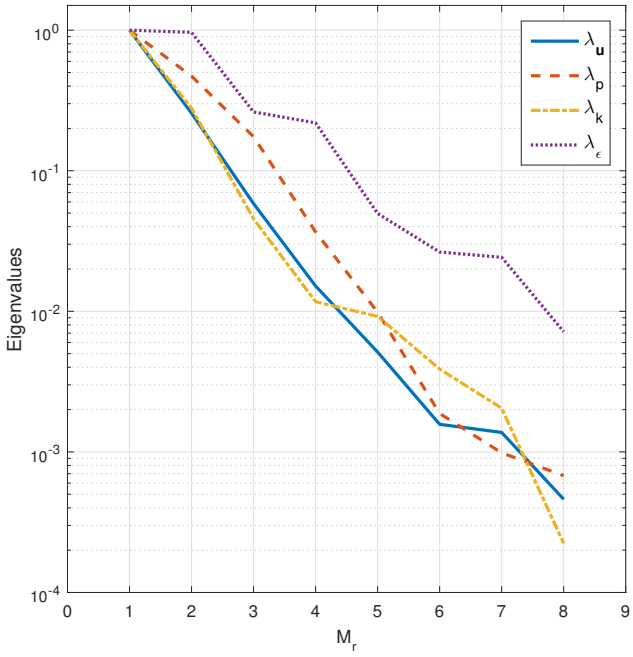
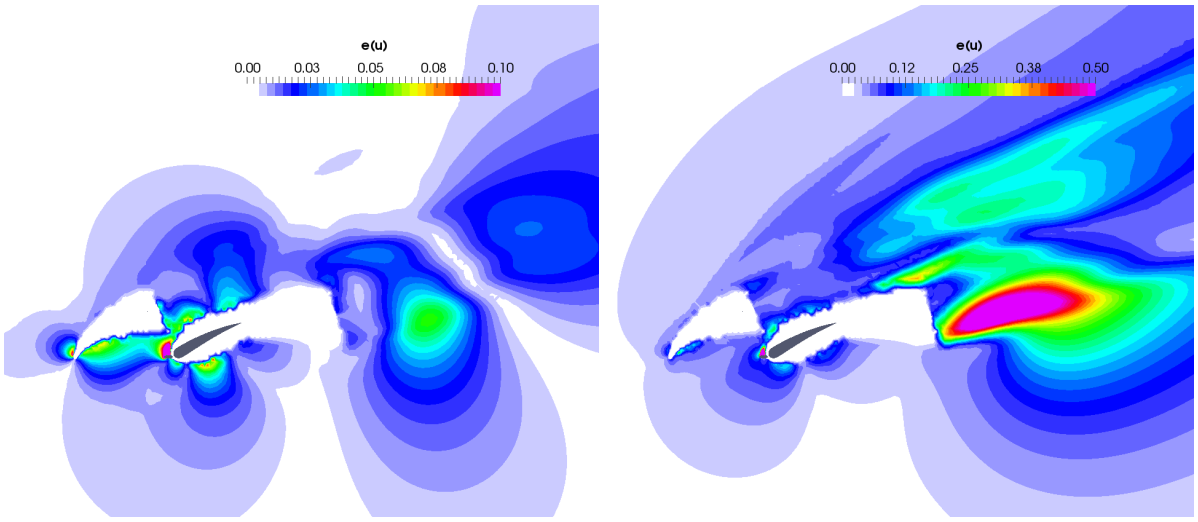


Fig. 8.21 POD eigenvalues for velocity, pressure and turbulent quantities, scaled with the corresponding maximum value.

of these error distributions can be found in Figure 8.22, where we report $e_p(\mathbf{x})$ and $e_{\mathbf{u}}(\mathbf{x})$ for a plane cut along the z-axis, normalised with the reference dynamic pressure and the reference velocity, respectively. As expected, the most critical regions are localised close to the body and in the turbulent wakes, especially in the proximity of the mainsail.

In order to derive a suitable decomposition, we consider different error thresholds σ_R for the field $e_{\mathbf{u}}(\mathbf{x})$, as shown in Figure 8.24. The corresponding $\Omega_1(\boldsymbol{\mu})$ domains are rectangular boxes containing all the deformable part of the geometry and all those cells that exceed the threshold value. For each domain, we verify the predictive capabilities of the setup for an out-of-sample configuration, corresponding to an intermediate maximum thickness of the inflatable device, equal to 20 cm. The performance of the zonal-POD approach for the domains of Figure 8.24 are collected in Table 8.4, where the term $\epsilon_T = |T - T^\star|/|T|$ represents the relative error on the objective function, i.e. the sailing system thrust. For the optimization, we chose the domain corresponding to $\sigma_R = 0.4U_{\text{ref}}$, characterized by an error less than 1% and by a speed-up factor of 19 with respect to the FOM. It should be noted how the overall speed-up factor is given not only by the ratio between the number of cells in the computational domain, but it is enhanced by a faster convergence of the hybrid model: the average number of iterations required



(a) $e_p(\mathbf{x})$. (b) $e_{\mathbf{u}}(\mathbf{x})$ magnitude.

Fig. 8.22 Pressure and velocity leave-one-out error distributions on the $z = 4$ m section. The white areas near the sails represent those cells where the POD modes cannot be defined.

to obtain convergence on the forces is at least three times smaller than the full-order case, thanks to a better flow initialization, as shown in Figure 8.23. Considering for instance a sail configuration not included in the database, we observe that the simulation residuals (Figure 8.23(left)) tend to get stable after about 1000 iterations for the hybrid approach, whereas the FOM requires approximately 3000 iterations. The same is trend is confirmed also by the evolution of the integral objective function T (Figure 8.23(right)), although with less evidence: convergence in the driving force for the ROM is reached in half the iterations with respect to the FOM simulation. For this application anyway, we employ a convergence criterion based on residuals, rather than objective functions As a consequence, while the full-order simulation requires about 230 cpuh, for the zonal-POD approach in the selected domain the computational time drops to 12 cpuh.

domain	σ_R	ϵ_T	#cells	iters	speed-up
full			$25.7 \cdot 10^6$	3000	1
(a)	0.1	0.0066	$5.3 \cdot 10^6$	1000	15
(b)	0.2	0.0093	$4.8 \cdot 10^6$	1000	17
(c)	0.4	0.0099	$4.1 \cdot 10^6$	1000	19

Table 8.4 Performance of the zonal-POD approach for the reduced domains (a), (b) and (c) of Figure 8.24. The error on the objective function, ϵ_T, refers to a predictive simulation for an intermediate configuration not included in the initial database.

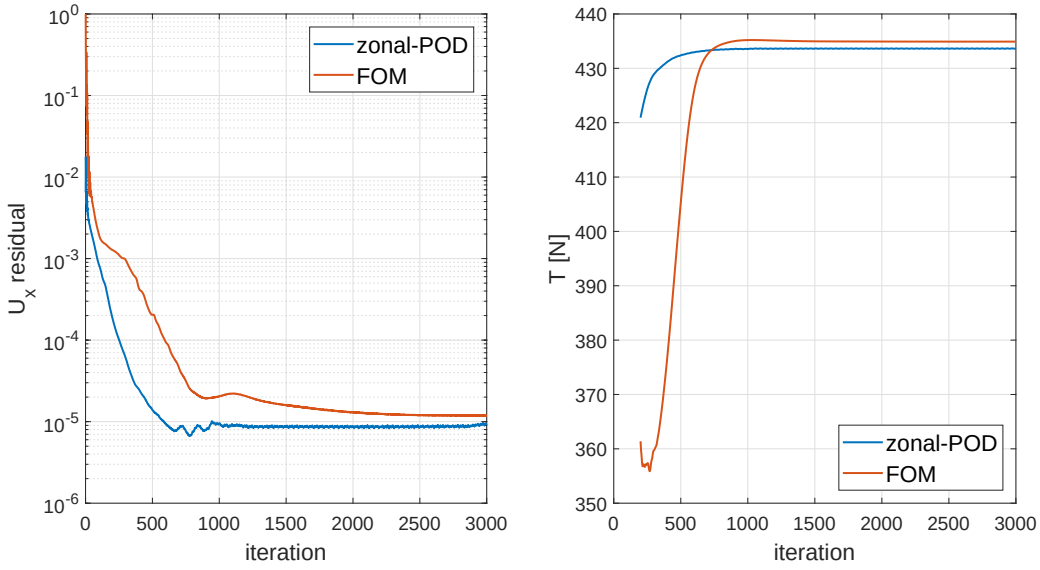
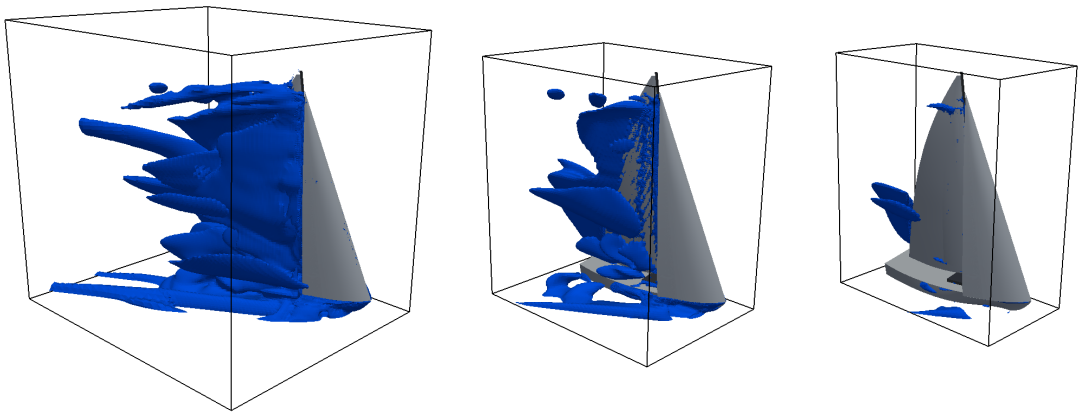
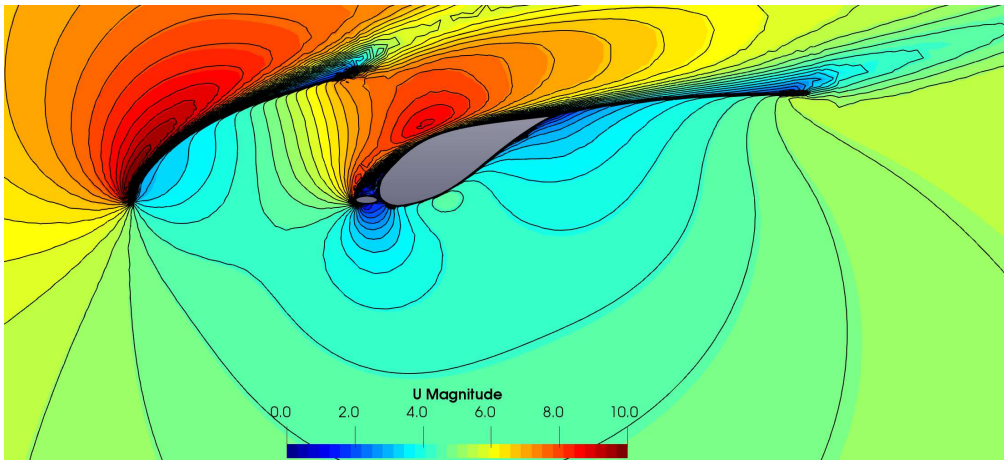


Fig. 8.23 Convergence performance of the ROM w.r.t. the FOM for an out-of-sample configuration: residual of the u_x component of the velocity field (left) and sailing system thrust (right) over the SIMPLE iterations.

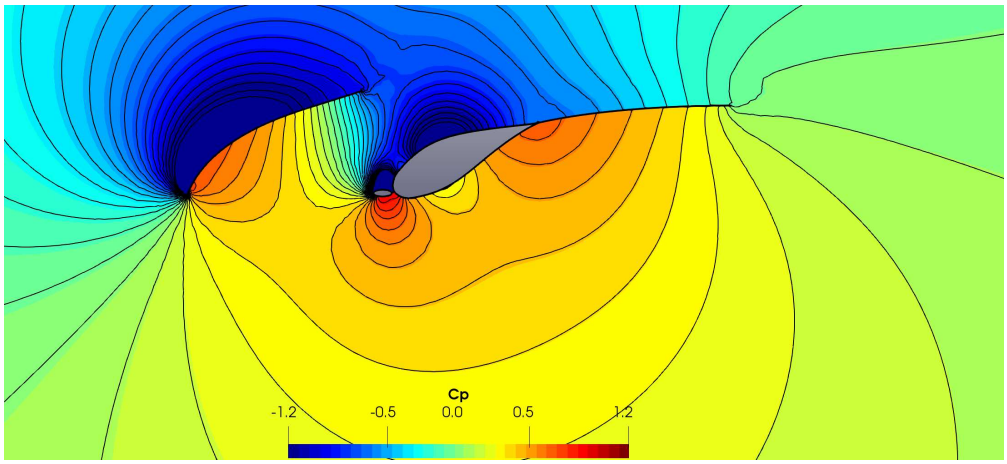


(a) $\sigma_R = 0.1U_{\text{ref}}$. (b) $\sigma_R = 0.2U_{\text{ref}}$. (c) $\sigma_R = 0.4U_{\text{ref}}$.

Fig. 8.24 The domain $\Omega_1(\boldsymbol{\mu})$ for three different tolerance values of the leave-one-out error on the velocity field $e_{\mathbf{u}}(\mathbf{x})$. The blue isosurfaces represent the error envelope for a given error threshold, whereas the black lines delimit the corresponding reduced domain used in the computations.



(a) $\mathbf{u}$ (colormap) vs $\mathbf{u}^\star$ (black isolines).



(b) C_p (colormap) vs $C_p^\star$ (black isolines).

Fig. 8.25 FOM-ROM fields comparison for an out-of-sample configuration (optimization best) on the $z = 4$ m section.

In this case, since we include the whole sailing boat within the $\Omega_1(\boldsymbol{\mu})$ domain, the optimization objective function is given only by the fluid dynamic contribution addressed by the full-order model in the reduced domain, and there is no need to recover the solution outside. Thus, the POD is calculated only in a slightly bigger domain than $\Omega_1(\boldsymbol{\mu})$, to interpolate correctly the boundary values on the interface.

The accuracy of the method for an out-of-sample configuration is shown in Figure 8.25, where we report a comparison between the full-order and hybrid solutions for the velocity and pressure coefficients fields. The ROM approximation is quite good on the windward side of the sails, whereas more discrepancy can be observed on the leeward side and in the wakes of both jib and main sail, in agreement with the error analysis discussed in the previous paragraphs.

8.2.4 Optimization setup

The optimization procedure is driven by a surrogate-based global method (see Section 7.1.1), using the parameter set given by $\boldsymbol{\mu} = (\theta_{\mathrm{M}}, \theta_{\mathrm{J}}, p_1, p_2, p_3)$ and the driving force T as objective function, as introduced in the previous sections. Starting from the 9 high-fidelity simulations performed with the FOM, a meta-model of the thrust response surface is built, using a Kriging regression model. When the algorithm requires to update the response surface, a new functional evaluation is performed using the zonal-POD approach.

The optimization process is stopped when convergence is attained, or when the number of evaluations exceeds a certain prescribed value.

8.2.5 Results

The optimum, in terms of maximum sailing boat thrust, is found after 9 full-order and 90 zonal-POD functional evaluations. As for the computational costs, the hybrid full-order/reduced-order model permits to significantly reduce the total cost of the optimization process, compared to standard strategies. The total cost of the process is about 3000 cpuh, including also the time spent for the mesh generation and the interpolation of the POD modes. If we consider an analogous surrogate-based optimization, performed using 99 full-order simulations, whose individual cost is about 230 cpuh, we obtain that the boost factor given by the hybrid strategy let us save around the 86% of the cpu time.

The optimal configuration is found for the parameter set

$$\boldsymbol{\mu}_{\mathrm{best}} = (-0.726^\circ, 0.588^\circ, 93.8 \text{ mm}, 17.81 \text{ mm}, 24.26 \text{ mm}),$$

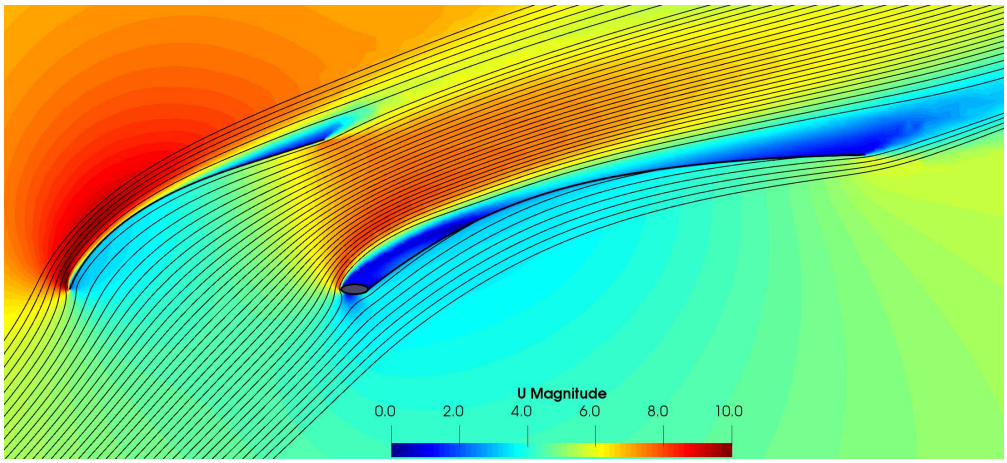
that leads to a driving force $T_{\mathrm{best}} = 434.91$ N, with a net gain (evaluated on the FOM simulations) of 4.03% with respect to the baseline mainsail configuration without inflatable device ($T_{\mathrm{base}} = 418.07$ N). As shown in Table 8.5, the zonal-POD prediction of the aerodynamic force originated by the sailing system is very accurate (relative error $< 0.3\%$) for the x-component, i.e. in the thrust direction, whereas the errors are more relevant in the lateral and vertical directions, about 1% and 3% respectively. This is not surprising given that the ROM has been set using the error on the driving force. Moreover, the prediction in this case is conservative, meaning that the hybrid approach tends to underestimate the objective function, leading to a *true* improvement that slightly exceeds the expectation. A comparison between the full-order and reduced-order solution

for the best configuration is depicted in Figure 8.25, representing the discrepancies in the flow field, in terms of velocity and pressure. The results of the optimization are shown

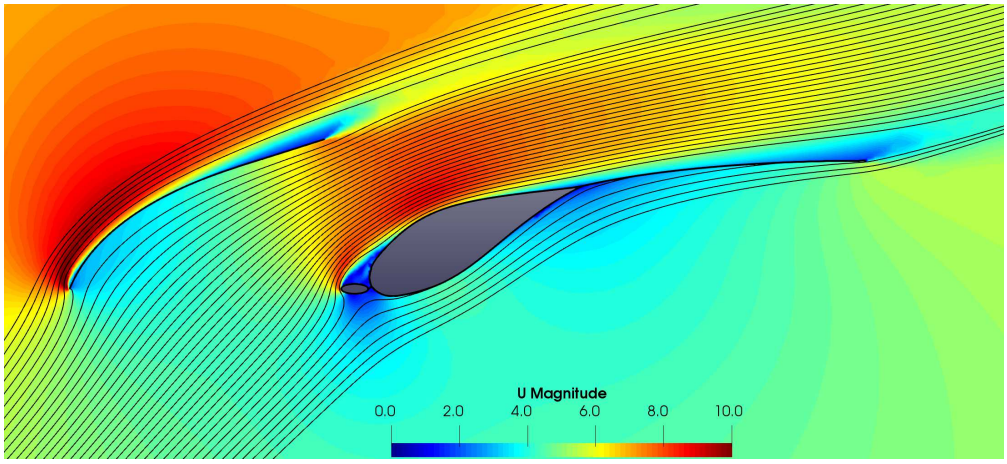
	FOM	ROM	$\lvert F_i - F_i^\star \rvert / \lvert F_i \rvert$
$F_{\mathrm{x}} (= -T)$	-434.91 N	-433.64 N	0.0029
F_{y}	936.15 N	946.77 N	0.0113
F_{z}	-85.60 N	-88.12 N	0.0294

Table 8.5 Aerodynamic forces for the optimized configuration: full-order solution, reduced-order solution and relative errors.

in Figures 8.27, 8.28 and 8.26. The new position of the sails is rather close to the baseline configuration, with global rotations of $-0.138°$ for the jib and $0.588°$ for the main sail, meaning that the device introduction does not significantly alter the trim of the system. The inflation of the profile can be observed in Figure 8.26, showing the velocity fields on a plane cut at $z = 4$ m for both the baseline and best configuration: as an effect of the altered mainsail profile, the flow recovers faster, allowing the leeward side of the sail to work in more efficient conditions, as appears clear from the behaviour of the streamlines in this area. Such result is also confirmed in Figure 8.27, representing the recirculation bubble that originates downstream of the mast: the separation is considerably reduced in the optimized configuration, with the exception of the head region, where the device is truncated. In terms of pressure distribution (Figure 8.28), the profile increases the suction on the leewards side, while partially reducing the pressure on the windward one. In this way, a fraction of the benefits are lost: generally speaking, an asymmetrical profile would probably be more efficient in terms of driving force generated by the system, but manufacturability and tack manoeuvrability requirements would be more challenging to meet.

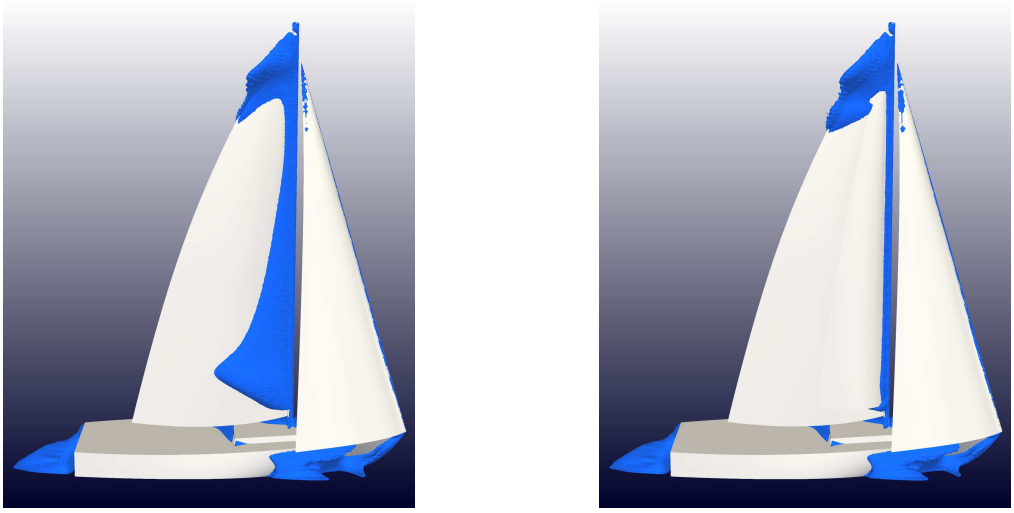


(a) Baseline configuration.



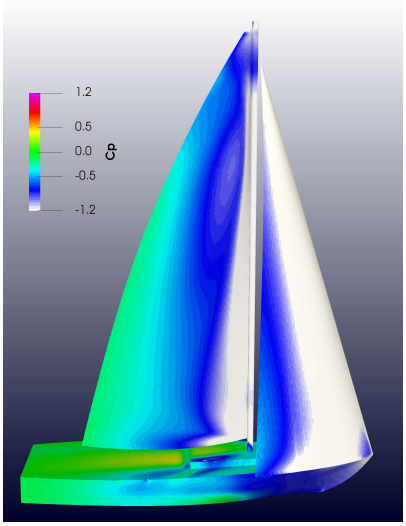
(b) Optimized configuration.

Fig. 8.26 Velocity streamlines and velocity field on the $z = 4$ m section.

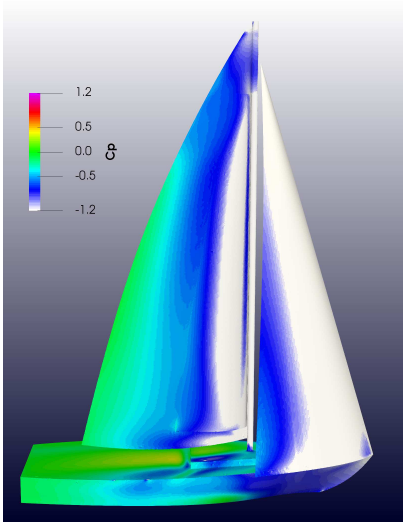


(a) Baseline configuration. (b) Optimized configuration.

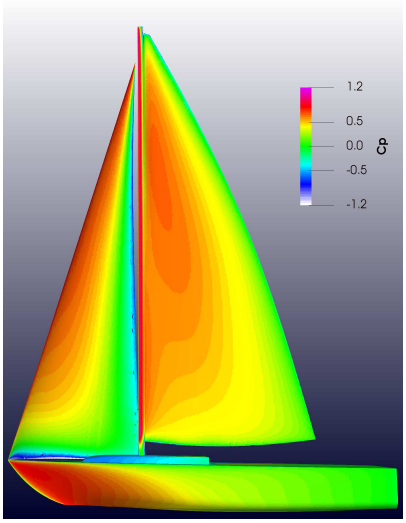
Fig. 8.27 Flow separation over the mainsail.



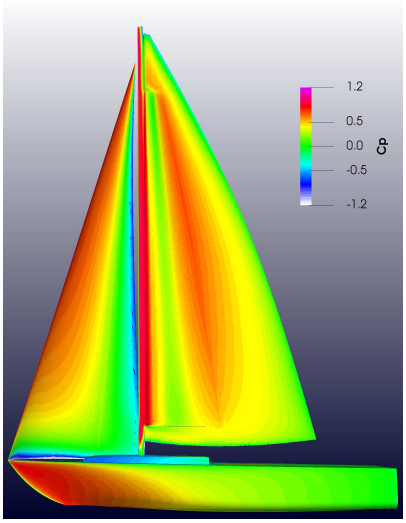
(a) Baseline configuration.



(b) Optimized configuration.



(c) Baseline configuration.



(d) Optimized configuration.

Fig. 8.28 C_p distribution on the windward (bottom) and leeward (top) sides of the sailing system.

Chapter 9

Conclusions

In the present work we have dealt with the shape optimization of large-scale aerodynamic problems, focusing in particular on the definition of an accurate surrogate model from POD-based model order reduction techniques. Generally speaking, the non-linearity and complexity of NSE for turbulent flows make the use of ROMs extremely challenging for real-life parameterized CFD problems, mainly due to stability and robustness issues, as discussed in Chapter 5. Moreover, for this kind of problems, the training of the ROM is extremely demanding, further limiting the applicability of such techniques in the industrial framework.

In order to mitigate some of these difficulties, we have adapted the zonal-POD approach first proposed by Buffoni et al. (2009), to the solution of incompressible flow on unstructured grids, further investigating the convergence properties of the method, as well as its capability to perform predictive simulations, like the ones required during an optimization loop. In this approach, the computational domain is split in two overlapping subdomains, $\Omega_1(\boldsymbol{\mu})$ and Ω_2, and different approximation methods are employed in each one: in particular, the canonical FOM is used to describe the flow field within a crucial region, where the solution strongly depends on the design parameters, while the rest of the domain is described by a semi-empirical model based on POD. The two model are then coupled in an overlapping region through a modified Schwarz method, resulting in a non-local boundary condition for the full-order solver on the reduced subdomain.

The convergence conditions of the algorithm have been discussed and then mathematical proof of convergence has been presented for a simplified linear problem: for the NS and RANS operators, stability and convergence properties have been verified on a

two-dimensional turbulent problem representative of the applications of interest, i.e. the 2DCAR benchmark.

Starting from a set of preliminary results on the 2DCAR benchmark, we have tested the hybrid approach performance for two industrial applications, namely the DrivAer car model and the J80 sailing boat, under realistic working condition (i.e. Re of $O(10^6)$). The method has been proved to be stable and robust thanks to the CFD feedback, as well as able to perform predictive simulations, enabling the low cost evaluation of fluid dynamic quantities, with a speed-up factor in terms of computational time of $O(10-20)$. Although the numerical results presented in this work have been obtained for the finite volume discretization of the RANS equations, the choice of the numerical discretization is not relevant for the definition of the hybrid method, making the approach potentially suitable for a wide class of fluid dynamic problems, as shown in recent applications (Iuliano and Quagliarella (2013), Scardigli et al. (2019), Bergmann et al. (2018)).

In terms of accuracy, we have shown that the approximation error for out-of-sample simulations depends on both the choice of the decomposition and the selection of the solution snapshots used to build the underlying POD model, as addressed in Chapter 6. For this reason, we have initially focused on domain decomposition, through the detection of the crucial zones where the POD basis fails to represent non-linearities, that need to be addressed by the FOM. In order to build an error indicator, we have adopted a leave-one-out strategy which consists in iteratively projecting one snapshot of the database onto the subspace spanned by all the remaining ones. This cross-validation procedure is an out-of-sample estimate of how much the samples are independent and allows one to obtain a spatial error map for each snapshot. By combining all these error fields, it is possible to evaluate a global error map that can be used to identify the most critical regions, hence a suitable domain decomposition. Besides being non-intrusive and efficient to compute, this error indicator built on the full-order solutions has been proven to be a good indicator also for the hybrid full-order/reduced-order model, as shown by numerical results in Chapters 6 and 8, where the correlation between leave-one-out error threshold and predictive results accuracy is clearly documented. For the cases under examination, the error map built on the velocity fields has been used and we have selected $\Omega_1(\boldsymbol{\mu})$ as the bounding box of the cells characterized by certain error values: however, there is no evidence at this stage whether such choice is optimal or a more meaningful error maps combination should be preferable. In addition, taking the bounding box of the error map is surely a suboptimal option: this approach simplifies the workflow, but forces to include in $\Omega_1(\boldsymbol{\mu})$ also cells that are well represented by the POD model, penalising the achievable reduction. Considering only those cells whose error exceeds the

assigned threshold would fix this issue, but it may results in a $\Omega_1(\boldsymbol{\mu})$ domain given by a set of disjoint subdomains, each with a number of cells equal or greater than 1: an eventuality that may originates problems for the CFD solver. Thus, a more sophisticated approach for the Γ_1 interface positioning needs to be introduced and it is currently under investigation.

The possibility to employ a masked least-squares approach for the selection of the overlapping region has been also taken into account, but the results on the 2DCAR benchmark have shown limited variations in terms of solution accuracy.

Then, we have focused on the definition of an efficient sampling strategy for selecting the snapshots to build the POD model. As in standard model reduction techniques, the sampling of the parameter space represent a crucial point in assessing the predictive performance of the method. Since the systematic exploration of the parameter space tends to be prohibitive for industrial applications, sampling strategies based on the Greedy method represent a natural choice for optimal space identification. The core concept of greedy sampling is the definition of a sharp and inexpensive a posteriori error bound, to be used to drive the algorithm, by computing only winning candidate snapshots. Two possible definitions of such error indicator have been proposed. In the first approach, the error indicator is based on the residuals of the fluid dynamics equations evaluated by projecting the flow solution onto the POD basis. While promising in principle, both the resGA-PODI and resGA-L1O methods fail for industrial turbulent flows, where the FOM residuals are usually higher, in a way that makes it impossible to discern between the error due to the numerical discretization and the one due to the POD approximation. For this reason, we have chosen to employ as greedy error indicator the projection error built exploiting the leave-one-out method. In order to guarantee both the exploration and the exploitation of the parameter space, we have started by coupling the Greedy method with a CCVT, built using the projection error as density function. In this strategy, new well-spaced points are added iteratively, enriching the database in those regions where the error indicator exceeds a fixed tolerance. This leads to good results for both academic and industrial test cases (as shown in the 2DCAR benchmark and in the DrivAer car model optimization), allowing us to obtain the same predictive accuracy on the output of interest with a very limited number of sampling point, compared to uniform distributions. In particular, for the 2DCAR problem, the pGA-CCVT strategy shows better results on the average prediction error with respect to the uniform sampling, by using only 5 full-order simulations instead of 9. The main drawback of this method appears to be the curse of dimensionality, since the number of sampling points required to start the algorithm depends on the number of parameters P

through the relation 2^P. Another method taken into consideration in order to partially minimize this issue is GP regression, coupled with the EGO algorithm, in place of the CCVT. When a coarse initial sampling is provided, the pGA-EGO approach has comparable performances with respect to the pGA-CCVT, but it needs only a maximum of $\frac{(P+1)(P+2)}{2}$ points for its initialization, which is an attractive feature of this second strategy. In order to extend the sampling methodology for higher-dimensional problems, however, more efficient POD-basis updating strategies need to be introduced, as well as some remedies to improve the performance of the error response surrogate model.

The complete model-reduction pipeline has been applied to two industrial shape optimization problems, i.e. the optimization of the DrivAer car model front bumper with 2 design variables, and of the J80 sailing boat, with 5 parameters, resulting in a considerable speed-up of the process (58% and 86% respectively), evaluated taking into account both the offline phase required to train the POD-based surrogate model and the time spent online for new function evaluations. In terms of accuracy, even in these more complex cases, the zonal-POD approach guarantees a good approximation of the full-order solution, with small errors on the outputs of interest ($< 0.4\%$ and $< 1\%$, respectively). The method appears to be particularly convenient for the second problem, where the deformations are global and the design space is characterized by a greater number of parameters, but similar benefits are found also in cases where localized geometry variations are considered, as in the DrivAer case. The two problems introduce a geometry parameterization based on FFD techniques, applied to volume meshes, as in the first case, or to surfaces, as in the latter, whereas the optimizer relies on multi-fidelity global optimizations strategies, employing three different levels of fidelity: the FOM, the zonal-POD model and a GP model. Despite such differences in the parameterization approach, the zonal-POD model has been successfully applied in both cases, as a further proof of the flexibility of the method.

Thus, the proposed framework, based on non-intrusive tools for model reduction, appears to be suitable for a wide class of problems and has been successfully integrated within industrial workflows. Moreover, since the POD method does not rely on the parameterization, it allows one to re-use pre-existing simulation data (e.g. in the automotive industry, where new car models often differ from older ones for minor details), leading to significant cost savings in various design stages. Future perspectives include increasing the dimensions of the design space, in the limit of the range of applicability of global optimization.

The extension of the developed methodology to the study of unsteady turbulent flows through DNS or LES, although easy in principle, represents a future challenge, due to the extreme variability of the temporal and spatial scales appearing in the flow solutions, that usually require a very large POD basis.

References

Adams, B. M., Bauman, L. E., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hough, P. D., Hu, K. T., Jakeman, J. D., Stephens, J. A., Swiler, L. P., Vigil, D. M., and Wildey, T. M. (2014a). Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual. *Sandia National Laboratories, Tech. Rep. SAND2014-4633*. Updated May 2017 (Version 6.6).

Adams, B. M., Bauman, L. E., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hough, P. D., Hu, K. T., Jakeman, J. D., Stephens, J. A., Swiler, L. P., Vigil, D. M., and Wildey, T. M. (2014b). Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 Theory Manual. *Sandia National Laboratories, Tech. Rep. SAND2014-4253*. Updated May 2017 (Version 6.6).

Ahmed, S. R., Ramm, G., and Faltin, G. (1984). Some Salient Features of the Time-Averaged Ground Vehicle Wake. *SAE Transactions*, 93:473–503.

Alliez, P., Ucelli, G., Gotsman, C., and Attene, M. (2008). *Recent Advances in Remeshing of Surfaces*, pages 53–82. Mathematics and Visualization. Springer.

Amoiralis, E. I. and Nikolos, I. K. (2008). Freeform Deformation Versus B-Spline Representation in Inverse Airfoil Design. *J. Comput. Inf. Sci. Eng.*, 8(2):024001–024001–13.

Amsallem, D., Farhat, C., and Zahr, M. (2013). On the Robustness of Residual Minimization for Constructing POD-Based Reduced-Order CFD Models. In *21st AIAA Fluid Dynamics Conference*. San Diego, CA. AIAA 2013-2447.

Anderson, G. R., Aftosmis, M. J., and Nemec, M. (2012). Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. Nashville, TN. AIAA 2012-0965.

Andreoli, M., Janka, A., and Désidéri, J. A. (2003). Free-form-deformation parameterization for multilevel 3D shape optimization in aerodynamics. Research Report RR-5019, INRIA.

Astrid, P., Weiland, S., Willcox, K., and Backx, T. (2008). Missing Point Estimation in Models Described by Proper Orthogonal Decomposition. *IEEE Transactions on Automatic Control*, 53(10):2237–2251.

References

Aubry, N. (1991). On the hidden beauty of the proper orthogonal decomposition. *Theor. Comp. Fluid. Dyn.*, 2(5):339–352.

Ballarin, F., Manzoni, A., Rozza, G., and Salsa, S. (2014). Shape Optimization by Free-Form Deformation: Existence Results and Numerical Solution for Stokes Flows. *J. Sci. Comput.*, 60(3):537–563.

Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math.*, 339(9):667–672.

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

Bergmann, M., Bruneau, C.-H., and Iollo, A. (2009). Enablers for robust POD models. *J. Comput. Phys.*, 228(2):516–538.

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

Bergmann, M., Ferrero, A., Iollo, A., Lombardi, E., Scardigli, A., and Telib, H. (2018). A zonal Galerkin-free POD model for incompressible flows. *J. Comput. Phys.*, 352:301–325.

Berkooz, G., Holmes, P., and Lumley, J. L. (1993). The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annu. Rev. Fluid Mech.*, 25(1):539–575.

Blocken, B., Stathopoulos, T., and Carmeliet, J. (2007). CFD simulation of the atmospheric boundary layer: wall function problems. *Atmos. Environ.*, 41(2):238–252.

Braconnier, T., Ferrier, M., Jouhaud, J.-C., Montagnac, M., and Sagaut, P. (2011). Towards an adaptive POD/SVD surrogate model for aeronautic design. *Comput. Fluids*, 40(1):195–209.

Buffoni, M., Telib, H., and Iollo, A. (2009). Iterative methods for model reduction by domain decomposition. *Comput. Fluids*, 38(6):1160–1167.

Buffoni, M. and Willcox, K. (2010). Projection-based model reduction for reacting flows. In *40th Fluid Dynamics Conference and Exhibit*. Chicago, IL. AIAA 2010-5008.

Bui-Thanh, T. (2003). Proper Orthogonal Decomposition Extensions and Their Applications in Steady Aerodynamics. Master's thesis, Singapore-MIT Alliance.

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

Carlberg, K., Bou-Mosleh, C., and Farhat, C. (2011). Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *Int. J. Numer. Meth. Eng.*, 86(2):155–181.

Cawley, G. C. and Talbot, N. L. C. (2003). Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. *Pattern Recogn.*, 36(11):2585–2592.

Chaturantabut, S. and Sorensen, D. C. (2010). Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764.

Chinesta, F., Huerta, A., Rozza, G., and Willcox, K. (2017). *Model Reduction Methods*, volume 3 of *Encyclopedia of Computational Mechanics*. John Wiley & Sons, 2nd edition.

Chinesta, F., Leygue, A., Bordeu, F., Aguado, J. V., Cueto, E., González, D., Alfaro, I., Ammar, A., and Huerta, A. (2013). PGD-Based computational vademecum for Efficient Design, Optimization and Control. *Arch. Comput. Methods Eng.*, 20(1):31–59.

Cogotti, A. (1998). A Parametric Study on the Ground Effect of a Simplified Car Model. SAE Techincal Paper 980031.

Cordier, L. and Bergmann, M. (2003). Proper Orthogonal Decomposition: an overview. *Lecture series 2003-2004 on post-processing of experimental and numerical data*, Von Karman Institute for Fluid Dynamics. https://www.math.u-bordeaux.fr/~mbergman/PDF/OuvrageSynthese/vki03-1.pdf.

Crane, K., Weischedel, C., and Wardetzky, M. (2013). Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.*, 32(5):152:1–152:11.

Cressie, N. (1993). *Statistics for Spatial Data*. John Wiley & Sons.

Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. SIAM.

Désidéri, J.-A., El Majd, B. A., and Janka, A. (2007). Nested and self-adaptive Bézier parameterizations for shape optimization. *J. Comput. Phys.*, 224(1):117–131.

Dolci, V. and Arina, R. (2016). Proper Orthogonal Decomposition as Surrogate Model for Aerodynamic Optimization. *Int. J. Aerospace Eng.*, 2016.

Duvigneau, R. (2006). Adaptive Parameterization using Free-Form Deformation for Aerodynamic Shape Optimization. Research Report RR-5949, INRIA.

Elisseeff, A. and Pontil, M. (2003). Leave-one-out error and stability of learning algorithms with applications. *NATO science series sub series iii computer and systems sciences*, 190:111–130.

Ferziger, J. H. and Peric, M. (2002). *Computational Methods for Fluid Dynamics*. Springer-Verlag, 3rd edition.

Forrester, A. I. J. and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progr. Aerosp. Sci.*, 45(1–3):50–79.

Forti, D. and Rozza, G. (2014). Efficient geometrical parametrisation techniques of interfaces for reduced-order modelling: application to fluid-structure interaction coupling problems. *Int. J. Comput. Fluid D.*, 28(3-4):158–169.

Fröhlich, J. and von Terzi, D. (2008). Hybrid LES/RANS methods for the simulation of turbulent flows. *Prog. Aerosp. Sci.*, 44(5):349–377.

Geisser, S. (1993). *Predictive Inference: An Introduction*, volume 55 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, 1st edition.

Giere, S., Iliescu, T., John, V., and Wells, D. (2015). SUPG reduced order models for convection-dominated convection-diffusion-reaction equations. *Comput. Methods Appl. Mech. Eng.*, 289:454–474.

Giles, M. B. and Pierce, N. A. (2000). An Introduction to the Adjoint Approach to Design. *Flow Turbul. Combust.*, 65(3-4):393–415.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. JHU Press, 4th edition.

Grepl, M. A. and Patera, A. T. (2005). A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM Math. Model. Numer. Anal.*, 39(1):157–181.

Han, Z.-H., Abu-Zurayk, M., Görtz, S., and Ilic, C. (2018). Surrogate-Based Aerodynamic Shape Optimization of a Wing-Body Transport Aircraft Configuration. In Heinrich, R., editor, *AeroStruct: Enable and Learn How to Integrate Flexibility in Design*, volume 138 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 257–282.

Heft, A. I., Indinger, T., and Adams, N. A. (2011). Investigation of Unsteady Flow Structures in the Wake of a Realistic Generic Car Model. In *29th AIAA Applied Aerodynamics Conference*. Honolulu, HI. AIAA 2011-3669.

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

Hesthaven, J. S., Rozza, G., and Stamm, B. (2016). *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Springer.

Hinze, M. and Volkwein, S. (2005). Proper Orthogonal Decomposition Surrogate Models for Nonlinear Dynamical Systems: Error Estimates and Suboptimal Control. In *Dimension reduction of large-scale systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 261–306. Springer.

Holmes, P., Lumley, J., and Berkooz, G. (1996). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1st edition.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 24(6):417–441.

Hucho, W.-H. (1987). *Aerodynamics of Road Vehicles: From Fluid Mechanics to Vehicle Engineering*. Elsevier, Butterworth-Heinemann, 1st edition.

Iollo, A., Lanteri, S., and Désidéri, J. A. (2000). Stability Properties of POD-Galerkin Approximations for the Compressible Navier-Stokes Equations. *Theor. Comp. Fluid Dyn.*, 13(6):377–396.

Iuliano, E. and Quagliarella, D. (2013). Proper Orthogonal Decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *Comput. Fluids*, 84:327 – 350.

Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, University of London and Imperial College.

Jasak, H., Jemcov, A., and Tukovic, Z. (2007). OpenFOAM: A C++ library for complex physics simulations. In *International Workshop on Coupled Methods in Numerical Dynamics*. IUC Dubrovnik, Croatia.

Jasak, H., Weller, H. G., and Gosman, A. D. (1999). High resolution NVD differencing scheme for arbitrarily unstructured meshes. *Int. J. Numer. Methods Fluids*, 31(2):431–449.

Jerison, D. S. and Kenig, C. E. (1995). The Inhomogeneous Dirichlet Problem in Lipschitz Domains. *J. Funct. Anal.*, 130:161–219.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag.

Jones, D. R. (2001). Direct Global Optimization Algorithm. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 431–440. Springer US.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *J. Global Optim.*, 13(4):455–492.

Jones, W. P. and Launder, B. E. (1972). The prediction of laminarization with a two-equation model of turbulence. *Int. J. Heat Mass Transf.*, 15(2):301 – 314.

Karhunen, K. (1946). Zur Spektraltheorie stochastischer Prozesse. In *Ann. Acad. Sci. Fenn.*, volume 34 of *Series A.I. Mathematica-Physica*.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *14th International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1145. Montreal, Canada.

Kosambi, D. D. (1943). Statistics in Function Space. *J. Indian Math. Soc.*, 7:76–88.

Koshakji, A., Quarteroni, A., and Rozza, G. (2013). Free Form Deformation techniques applied to 3D shape optimization problems. *Comm. App. Industr. Math.*, 4.

Kunisch, K. and Volkwein, S. (2002). Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics. *SIAM J. Numer. Anal.*, 40(2):492–515.

Lassila, T. and Rozza, G. (2010). Parametric free-form shape design with PDE models and reduced basis method. *Comput. Methods Appl. Mech. Eng.*, 199(23–24):1583–1592.

Launder, B. E. and Sharma, B. I. (1974). Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Lett. Heat Mass Trans.*, 1(2):131 – 138.

LeGresley, P. A. and Alonso, J. (2000). Airfoil Design Optimization Using Reduced Order Models Based on Proper Orthogonal Decomposition. In *Fluids 2000 Conference and Exhibit*. Denver, CO.

LeGresley, P. A. and Alonso, J. (2003). Dynamic Domain Decomposition and Error Correction for Reduced Order Models. In *41st Aerospace Sciences Meeting and Exhibit,*. Reno, NV.

Liu, F. (2016). A Thorough Description Of How Wall Functions Are Implemented In OpenFOAM. *CFD with OpenSource Software*. http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016/FangqingLiu/openfoamFinal.pdf.

Liu, J., Han, Z., and Song, W. (2012). Comparison of infill sampling criteria in Kriging-based aerodynamic optimization. In *28th Congress of the International Council of the Aeronautical Sciences*. Brisbane, Australia.

Loéve, M. (1955). *Probability theory. Foundations. Random sequences*. D. Van Nostrand Company Inc., 1st edition.

Lombardi, E., Bergmann, M., Camarri, S., and Iollo, A. (2011). Low-order models. Optimal sampling and linearized control strategies. *Journal Européen des Systèmes Automatisés*, 45(7–10):575–593.

Lucia, D. J., King, P. I., and Beran, P. S. (2003). Reduced order modeling of a two-dimensional flow with moving shocks. *Comput. Fluids*, 32(7):917–938.

Lumley, J. (1967). The Structures of Inhomogeneous Turbulent Flow. In Yaglom, A. M. and Tatarski, V. I., editors, *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178. Publishing House Nauka.

Maday, Y., Patera, A. T., and Turinici, G. (2002). A Priori Convergence Theory for Reduced-Basis Approximations of Single-Parameter Elliptic Partial Differential Equations. *J. Sci. Comput.*, 17(1-4):437–446.

Mathew, T. (2008). *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, volume 61 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag.

Mifsud, M., Zimmermann, R., and Görtz, S. (2015). Speeding-up the computation of high-lift aerodynamics using a residual-based reduced-order model. 6(1):3–16.

Mohammadi, B. and Pironneau, O. (2010). *Applied Shape Optimization for Fluids*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2nd edition.

Noack, B. R., Morzynski, M., and Tadmor, G. (2011). *Reduced-Order Modelling for Flow Control*, volume 528 of *CISM International Centre for Mechanical Sciences*. Springer-Verlag.

Noack, B. R., Papas, P., and Monkewitz, P. A. (2005). The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows. *J. Fluid Mech.*, 523:339–365.

Östh, J., Noack, B. R., Krajnovic, S., Barros, D., and Borée, J. (2014). On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. *J. Fluid Mech.*, 747:518–544.

Patankar, S. V. (1980). *Numerical Heat Transfer and Fluid Flow*. Computational Methods in Mechanics and Thermal Sciences. CRC press.

Perotto, S., Reali, A., Rusconi, P., and Veneziani, A. (2017). HIGAMod: A Hierarchical IsoGeometric Approach for MODel reduction in curved pipes. *Comput. Fluids*, 142:21–29.

Pope, S. B. (2011). *Turbulent Flows*. Cambridge University Press.

Prud'Homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2001). Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods. *J. Fluids Eng.*, 124(1):70–80.

Quarteroni, A. and Rozza, G. (2014). *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A - Modeling, Simulation and Applications*. Springer.

Quarteroni, A. and Valli, A. (1999). *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation. Oxford University Press.

Sagaut, P. (2006). *Large Eddy Simulation for Incompressible Flows: An Introduction*. Scientific Computation. Springer-Verlag, 3rd edition.

Salmoiraghi, F., Ballarin, F., Corsi, G., Mola, A., Tezzele, M., and Rozza, G. (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 *VII European Conference on Computational Methods in Applied Sciences and Engineering*. Crete, Greece.

Salmoiraghi, F., Ballarin, F., Heltai, L., and Rozza, G. (2016b). Isogeometric analysis-based reduced order modelling for incompressible linear viscous flows in parametrized shapes. *Adv. Model. and Simul. in Eng. Sci.*, 3(1).

Salmoiraghi, F., Scardigli, A., Telib, H., and Rozza, G. (2018). Free Form Deformation, mesh morphing and reduced order methods: enablers for efficient aerodynamic shape optimization. *Int. J. Comput. Fluid Dyn.* To appear.

Samareh, J. A. (2001). Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization. *AIAA journal*, 39(5):877–884.

Scardigli, A., Arpa, R., Chiarini, A., and Telib, H. (2019). Enabling of Large Scale Aerodynamic Shape Optimization Through POD-Based Reduced-Order Modelling and Free Form Deformations. In Minisci, E., Vasile, M., Periaux, J., Gauger, N., Giannakoglou, K., and Quagliarella, D., editors, *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences.*, volume 48 of *Computational Methods in Applied Sciences*, pages 49–63. Springer.

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.*, 656:5–28.

Sederberg, T. W. and Parry, S. R. (1986). Free-Form Feformation of Solid Geometric Models. *ACM SIGGRAPH Computer Graphics*, 20(4):151–160.

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2nd edition.

Shih, T.-H., Liou, W. W., Shabbir, A., Yang, Z., and Zhu, J. (1995). A new k-ϵ eddy viscosity model for high reynolds number turbulent flows. *Comput. Fluids*, 24(3):227–238.

Sieger, D., Menzel, S., and Botsch, M. (2015). On Shape Deformation Techniques for Simulation-Based Design Optimization. In Perotto, S. and Formaggia, L., editors, *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, volume 5 of *SEMA SIMAI Springer Series*, pages 281–303. Springer.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. Part I, II and III. *Q. Appl. Math.*, 45(3):561–590.

Sobieczky, H. (1997). Geometry Generator for CFD and Applied Aerodynamics. In *New Design Concepts for High Speed Air Transport*, volume 366 of *International Centre for Mechanical Sciences*, pages 137–157. Springer.

Spalart, P. R. and Allmaras, S. R. (1992). A One-Equation Turbulence Model for Aerodynamic Flows. In *30ty Aerospace Sciences Meeting and Exhibit*. Reno, NV.

Spalart, P. R. and Allmaras, S. R. (1994). A One-Equation Turbulence Model for Aerodynamic Flows. *La Recherche Aérospatiale*, 1:5–21.

Spalding, D. B. (1961). A Single Formula for the "Law of the Wall". *J. Appl. Mech.*, 28(3):455–458.

Vendl, A., Faßbender, H., Görtz, S., Zimmermann, R., and Mifsud, M. (2014). Model order reduction for steady aerodynamics of high-lift configurations. *CEAS Aeronaut. J.*, 5(4):487–500.

Versteeg, H. K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2nd edition.

Volkwein, S. (2013). Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. *Lecture Notes*, University of Konstanz. http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf.

Wang, Z., Akhtar, I., Borggaard, J., and Iliescu, T. (2012). Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Comput. Methods Appl. Mech. Eng.*, 237–240:10–26.

Warming, R. F. and Beam, R. M. (1976). Upwind Second-Order Difference Schemes and Applications in Aerodynamic Flows. *AIAA Journal*, 14(9):1241–1249.

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.*, 12(6):620–631.

Weller, J., Camarri, S., and Iollo, A. (2009a). Feedback control by low-order modelling of the laminar flow past a bluff body. *J. Fluid Mech.*, 634:405–418.

Weller, J., Lombardi, E., Bergmann, M., and Iollo, A. (2010). Numerical methods for low-order modeling of fluid flows based on POD. *Int. J. Numer. Methods Fluids*, 63(2):249–268.

Weller, J., Lombardi, E., and Iollo, A. (2009b). Robust model identification of actuated vortex wakes. *Physica D*, 238(4):416–427.

Wieringa, J. (1992). Updating the Davenport roughness classification. *J. Wind Eng. Ind. Aerod.*, 41(1-3):357–368.

Wilcox, D. C. (2006). *Turbulence Modeling for CFD*. DCW Industries, 3rd edition.

Zhan, Z., Habashi, W. G., and Fossati, M. (2015). Local Reduced-Order Modeling and Iterative Sampling for Parametric Analyses of Aero-Icing Problems. *AIAA Journal*, 53(8):2174–2185.

Zimmermann, R. (2011). A Comprehensive Comparison of Various Algorithms for Efficiently Updating Singular Value Decomposition Based Reduced Order Models. DLR-Interner Bericht DLR-IB 124-2011/3, DLR. https://elib.dlr.de/70251/.

Zimmermann, R. and Görtz, S. (2010). Non-linear reduced order models for steady aerodynamics. *Procedia Comput. Sci.*, 1(1):165–174.

Zimmermann, R. and Görtz, S. (2012). Improved extrapolation of steady turbulent aerodynamics using a non-linear POD-based reduced order model. *Aeronaut. J.*, 116(1184):1079–1100.

Zimmermann, R. and Willcox, K. (2016). An Accelerated Greedy Missing Point Estimation Procedure. *SIAM J. Sci. Comput.*, 38(5):A2827–A2850.