UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXVIII CICLO DEL DOTTORATO DI RICERCA IN

INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

NON-INTRUSIVE MULTI-FIDELITY PARAMETRIC REDUCED ORDER MODEL FOR INDUSTRIAL PROBLEMS

Settore scientifico-disciplinare: **ING-IND/08**

DOTTORANDO
FAUSTO DICECH

COORDINATORE
PROF. FULVIO BABICH

SUPERVISORE DI TESI
PROF. LUCIA PARUSSINI

CO-SUPERVISORE DI TESI
PROF. MARCO MANZAN

ANNO ACCADEMICO 2024/2025

Preface

A pure and naive drive for curiosity set the path for an unexpectedly winding three years journey. Every day I walked on this path, it made me take a step in a direction that I could have never predicted before. At the end of the trail, I realize how vital it is to question everything, especially our beliefs. Research taught me greatly, and I wish to return something to research with this dissertation.

This journey would not have been possible, nor would it have been as interesting, without the support of many people. I sincerely wish to express my gratitude to:

Professor Lucia Parussini, my PhD supervisor, for all the fruitful discussions and continuous guidance.

All the Optimad team, especially Alessandro Alaia, Haysam Telib, Konstantinos Gkaragkounis, Angela Scardigli, and Marco Cisternino, without whom I would not have been able to learn as much.

The wonderful people I met in Bordeaux, from the INRIA Memphis team and the Institut de Mathématiques, in particular professor Angelo Iollo, for the invaluable experience.

Finally, a special thanks to my family and Michela for their irreplaceable support and patience.

"Ma se capirai, se li cercherai fino in fondo
Se non sono gigli, son pur sempre figli, vittime di questo mondo"
– Fabrizio De André

Abstract

Industrial problems often involve parametric design and deal with complex computational models or expensive experimental measurements. The study of multiple design configurations, such as optimization or uncertainty quantification, can become prohibitive, especially when the number of observations becomes too high. Consequently, there is a profound interest in lowering the computational cost of parametric problems. Surrogate models, as interpolations or regression models, are a great solution to this issue, as they allow for learning a map between the input parameter space and the corresponding outputs. They usually approximate problem-specific quantities of interest, and their training is relatively inexpensive. Nevertheless, if the quantity of interest is not a scalar value, but it is a vector quantity, most regression models have a limited range of application, especially if the vector space is high-dimensional. This is the case for numerical simulation solutions, such as those for Computational Fluid Dynamics (CFD) or structural Finite Element Models (FEM), where the quantities of interest are vector fields, e.g., velocity or pressure fields. It would not be feasible to approximate an industrially relevant parametric solution with a single regression model due to the model's limitations. Reduced Order Models (ROM) aim to solve exactly this problem. They operate a dimensionality reduction of the vector fields, here called snapshots, by encoding them with Proper Orthogonal Decomposition (POD). Successively, the obtained snapshots' latent representation is exploited to predict unseen design configurations. In this work, a novel non-intrusive multi-fidelity ROM has been developed to address the high offline computational cost of classical ROMs. Low-fidelity (LF) information is exploited to improve the high-fidelity (HF) knowledge of the problem in a multi-fidelity manner. In particular, the proposed multi-fidelity ROM has been designed to handle industrial applications, while favoring the automation of the process.

The first part of the thesis discusses the methodological core of the work. Starting from ROMs, to regression models, their relationship is deepened, specifically for the non-intrusive ROM formulations. Successively, multi-fidelity regression models are

introduced, illustrating how different approaches to multi-fidelity techniques work. Finally, the novel non-intrusive multi-fidelity ROM is presented, combining the previous methods. The multi-fidelity ROM has two main characteristics: a mixed fidelity snapshot matrix and the use of a multi-fidelity regression model, i.e., the Nonlinear AutoRegressive multi-fidelity Gaussian Process (NARGP). The mixed fidelity snapshots enrich the POD basis functions, increasing the capability to represent unseen snapshots, while the NARGP improves the accuracy of the POD latent predictions through the addition of LF snapshots.

The second part of the thesis shows how the multi-fidelity ROM performs in industrially relevant use cases and what the main challenges are. External aerodynamics and internal flows are both considered, and different types of parameterizations are examined, ranging from geometrical to operational input variables. Many query applications are considered, e.g., uncertainty quantification, highlighting the potential of ROMs in these fields. Moreover, the problem of non-coherent snapshots due to different fidelity snapshots and large geometrical deformations is addressed in the proposed applications. In general, all the results are compared to equivalent single-fidelity ROMs to perform a meaningful comparison.

Contents

Acronyms

ARD	Automatic Relevance Determination;
CAD	Computer Aided Design;
CFD	Computational Fluid Dynamics;
DMD	Dynamic Mode Decomposition;
DNS	Direct Numerical Simulation;
DoE	Design of Experiments;
FD	Finite Differences;
FEM	Finite Element Method;
FOM	Full Order Model;
GP	Gaussian Process;
HF	High Fidelity;
L-BFGS	Limited memory Broyden–Fletcher–Goldfarb–Shanno;
LES	Large Eddy Simulation;
LHS	Latin Hypercube Sampling;
ML	Machine Learning;
MLE	Maximum Likelihood Estimation;
LF	Low Fidelity;
NARGP	Nonlinear AutoRegressive multi-fidelity Gaussian Process;
NI	Non-Intrusive;
NN	Neural Networks;
ODE	Ordinary Differential Equation;
PDE	Partial Differential Equation;
POD	Proper Orthogonal Decomposition;
RANS	Reynolds-Averaged Navier-Stokes;
RBF	Radial Basis Function;
RMSE	Root Mean Squared Error.
RQ	Rational Quadratic;
ROM	Reduced Order Model;
SDF	Signed Distance Function;
SE	Squared Exponential;
SVD	Singular Value Decomposition;
TPMS	Triply Periodic Minimal Surface;
VAE	Variational Auto-Encoder;
UQ	Uncertainty Quantification.

Symbols

a	POD coefficient;
C_D	drag coefficient;
C_P	power coefficient;
d	parameter space dimension;
D	dataset;
e_{int}	ROM interpolation error;
e_{prj}	ROM projection error;
e_{rec}	ROM reconstruction error;
F_T	thrust force;
F_R	radial force;
I	identity matrix;
k	kernel;
l	lengthscale;
$\mathcal{M}$	map;
N	snapshots space dimension;
n	number of snapshots;
n_{modes}	number of POD modes;
r	reduced basis dimension;
$\mathbf{S}$	snapshots matrix;
$\mathbf{u}$	snapshot;
$\mathbf{x}$	input parameter vector;
y	scalar output;
ϵ	radial basis function parameter;
ε	POD energy;
η	radial basis function;
λ	singular value;
$\boldsymbol{\mu}$	mean vector;
$\boldsymbol{\psi}$	POD mode;
$\boldsymbol{\Psi}$	POD basis set;
σ	standard deviation;

List of Figures

List of Tables

Chapter 1

Introduction

Computational resources are becoming more affordable over time, enabling a wider public to tackle increasingly complex problems. In an industrial context, this translates to a more accurate physical description, setting a higher threshold for what is tractable within a reasonable time frame. From a practical standpoint, this defines the high-fidelity representation of a problem. For instance, in numerical simulations, a higher fidelity level can be derived from more refined physical models, finer meshes, or higher-order schemes. Nonetheless, the amount of available high-fidelity information is always limited by time and resources. Real-life industrial scenarios often require multiple high-fidelity simulations and consist of parametric studies. Typical examples are optimization or uncertainty quantification. Surrogate models aim to overcome these limitations by providing high-fidelity approximations with reasonable accuracy in a portion of the parametric space. Response surfaces have proven to be good approximators for scalar quantities. However, the quantity of interest might be a large vector, e.g., a field from a numerical solution, and response surfaces are too demanding or directly intractable. Vectorized fields can be more informative than single scalar quantities and are necessary in numerous applications. Reduced Order Models (ROMs) are ideal surrogate models in this context. By definition, a ROM is an inexpensive Full Order Model (FOM) replacement, where a FOM is, i.e., a numerical solver. ROMs can play a crucial role in fields as shape optimization, multi-physics systems, simulation-based control, digital twins, or, in general, in many query workflows.

A ROM approximates numerical simulation outputs in quasi real-time. Yet, ROMs still require us to perform a set of high-fidelity simulations, affecting the overall computational cost. This is mandatory for both intrusive and non-intrusive

ROMs, where the ROM's intrusiveness stays in the direct access of the governing equations. In industrial frameworks, commercial solvers are employed, limiting the applicability of intrusive ROMs to a restricted selection of problems. Therefore, in this thesis, we will focus on the non-intrusive ROM formulations. To train a non-intrusive ROM a first design of experiment has to be computed. This constitutes most of the computational effort. Despite being often overlooked, the problem of training a ROM with limited resources drives the need for more data-efficient techniques. Appropriate sampling strategies for the design of experiments lower the number of high-fidelity simulations and have already been studied. Another possibility to explore is multi-fidelity methodologies. In analogy with regression models, where multi-fidelity methods have been developed first, the idea is to leverage heterogeneous information sources to improve the high-fidelity knowledge of the problem. Multi-fidelity models exploit the correlation between low- and high-fidelity information in a data fusion process. When the low-fidelity information is several times less expensive than the high-fidelity equivalent, multi-fidelity models can reduce the data-related computational cost. Consequently, transferring multi-fidelity techniques from classical regression models to ROMs aims to have a similar positive impact.

We focus on Proper Orthogonal Decomposition (POD)-based non-intrusive ROMs. All numerical solutions, also known as snapshots, have to be coherent to train the ROM. In other words, all snapshots are vectors with the same dimensionality. Since variable fidelity numerical simulations are often characterized by different discretizations, i.e., coarse and fine meshes, this poses a problem for POD-based ROMs. Geometrical parameterizations and large shape deformations raise a similar issue, particularly common in industrial applications. Hence, the harmonization of different snapshots, due to variable fidelity and/or geometries, is of utmost importance to extend multi-fidelity techniques to non-intrusive ROMs. Ensuring coherent snapshots is not always straightforward and can require case-specific solutions, such as mapping and geometrical registration.

In this thesis, we propose a POD-based parametric non-intrusive multi-fidelity ROM tailored to industrial applications. We aim to address sparse high-fidelity problems, eventually characterized by geometrical parameterization and non-coherent discretizations. In this novel multi-fidelity ROM framework, we highlight how the mixing of variable fidelity snapshots improves the encoding capabilities of POD. Moreover, the latent representation of the variable fidelity snapshots is exploited by a multi-fidelity regressor. Keeping in mind the industrial vocation of the proposed ROM, we utilize the Nonlinear AutoRegressive multi-fidelity Gaussian Pro-

cess (NARGP) regression model. NARGP is an inherently nonlinear multi-fidelity regressor that does not assume any correlation pattern in the variable fidelity latent snapshot representations. Together with a modest computational training cost, this makes NARGP a perfect candidate for our multi-fidelity ROM framework. We demonstrate the capabilities and limitations of the proposed multi-fidelity ROM through a series of industrially relevant test cases, primarily in the computational fluid dynamics field. A first application is for an automotive external aerodynamics study, with a geometrical parameterization. Free-form deformation has been used to morph a car geometry, and the resulting deformation fields are exploited to map the numerical solutions onto a reference car. To the author's knowledge, mapping surface fields from non-coherent boundary tessellations through free-form deformation is novel. A second application is an internal flow problem, with both geometrical and operational design parameters. Here, we show the synergy between meshless solvers and the proposed multi-fidelity ROM, with the aim of further simplifying and automating parametric studies. In a third application, we present an uncertainty quantification study of a vertical axis wind turbine under uncertain rotational speed. Finally, we use our multi-fidelity ROM to characterize the flow inside a triply periodic minimal surface for heat exchanger applications, with a focus on the field mapping strategy. Throughout these industrial test cases, it also emerges that geometrical parameterizations require particular attention in the context of ROMs.

1.1 General problem definition

In this thesis, all problems share the same fundamental structure, with their own characteristic and peculiarities. Here, we provide a simple problem definition to better contextualize this work.

Without loss of generality, we suppose that there are two fidelity levels, namely the low-fidelity (LF) and high-fidelity (HF) levels. Each fidelity level has a corresponding numerical solver, i.e., for computational fluid dynamics, which is more or less accurate according to the fidelity level. Given a parameter $\mathbf{x} \in \mathbb{R}^d$, where d is the number of parameters, the LF solver output of interest is $\mathbf{y}_{LF}(\mathbf{x}) \in \mathbb{R}^{N_L}$ and the corresponding HF output is $\mathbf{y}_{HF}(\mathbf{x}) \in \mathbb{R}^{N_H}$. N_L, N_H are the LF and HF discretization cardinalities, respectively, and can be different ($N_L < N_H$) or identical, depending on the variable fidelity characterization. Moreover, these cardinalities can differ from each other, even at the same fidelity level, e.g., if there are geometrical

deformations due to the parameterization.

The first step consist of computing a design of experiment, obtaining a set of LF simulations $S_{LF} = \{\mathbf{y}_{LF}(\mathbf{x}_1), ..., \mathbf{y}_{LF}(\mathbf{x}_{n_L})\}$ and HF simulations $S_{HF} = \{\mathbf{y}_{HF}(\mathbf{x}_1), ..., \mathbf{y}_{HF}(\mathbf{x}_{n_H})\}$, with $n_L < n_H$, and n_L, n_H being the number of low- and high-fidelity simulations, respectively. To perform the POD and successively train the multi-fidelity ROM, all $\mathbf{y} \in S_{LF}, S_{HF}$ have to be of the same dimension N. To achieve this, we name a mapping a transformation Φ that manipulates all the original solutions $\mathbf{y}$ so they become coherent snapshots. Generally, a mapping first applies a geometrical transformation that moves all points of $\mathbf{y}$ to a reference geometry, if needed, followed by a field interpolation.

After preprocessing all the snapshots, it is possible to assemble the matrix S, named the matrix of snapshots. This is done by appending all LF snapshots to the HF ones, resulting in

$$S = \begin{bmatrix} | & & | & | & & | \\ \mathbf{y}_1^{HF} & ... & \mathbf{y}_{n_H}^{HF} & \mathbf{y}_1^{LF} & ... & \mathbf{y}_{n_L}^{LF} \\ | & & | & | & & | \end{bmatrix}, \tag{1.1}$$

where the generic i-th snapshot $\mathbf{y}_i^{LF}, \mathbf{y}_i^{HF} \in \mathbb{R}^N$ is a solution that has been mapped on the reference discretization. Each snapshot in S corresponds to an input parameter vector $\mathbf{x}$, having

$$X = \{\mathbf{x}_1^{HF}, ..., \mathbf{x}_{n_H}^{HF}, \mathbf{x}_1^{LF}, ..., \mathbf{x}_{n_L}^{HF}\} \tag{1.2}$$

being the set of all training input parameters.

Successively S, together with the corresponding input parameters X, is used to train the multi-fidelity ROM. Once the multi-fidelity ROM has been trained, it can be used to predict an approximation unseen HF outputs y^{HF} for any input parameter vector $\mathbf{x}$. Figure 1.1 gives an intuition of a general multi-fidelity ROM training and usage.

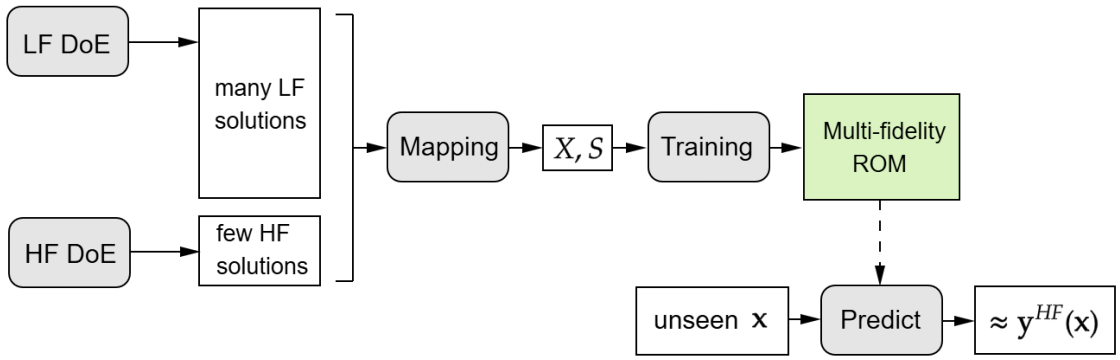


Figure 1.1: Intuitive multi-fidelity ROM workflow schematics.

1.2 Thesis outline

This thesis is organized into five chapters. Following this introductory chapter, Chapter 2 introduces the concept of POD-based parametric ROMs. Here, we define the POD, how it is used in intrusive ROMs, and why intrusive ROMs are not suitable for most of the industrial applications. Successively, we focus on the single-fidelity non-intrusive ROM formulation and the problem of mapping heterogeneous snapshots. Finally, we provide a shape optimization example for an heat exchanger application to give more context to non-intrusive ROMs.

Chapter 3 focus on regression models and is structured in two parts: single-fidelity models and multi-fidelity models. In the first one, we present GP regression models, and introduce other models such as radial basis function interpolations or artificial neural networks. In the second part, we discuss how multi-fidelity methods works, presenting the NARGP multi-fidelity model. Successively, we introduce other multi-fidelity approaches, such as Cokriging, multi-fidelity neural networks, and an original simplified multi-fidelity model.

Chapter 4 integrates non-intrusive POD-based ROMs of Chapter 2 with multi-fidelity modeling of Chapter 3. Here, the main focus is on how variable fidelity snapshots are managed, and how the multi-fidelity regression problem in the POD latent space is set. Finally, the choice of NARGP as the main multi-fidelity regressor is motivated.

Chapter 5 is a collection of numerical experiments where we apply the multi-

fidelity ROM proposed in Chapter 4. The first application is an external aerodynamic study for an automotive problem, with six geometrical parameters. The second application is an internal flow study inside an annular pipe with parametrized wavy surfaces, with the governing equations solved by a meshless solver. In the third application, we perform an uncertainty quantification of the forces acting on a vertical axis wind turbine under uncertain rotational speed. Finally, in the fourth application, we study a second internal flow problem, with a focus on the mapping strategy for gyroid periodical surfaces.

In Appendix A we present two collateral studies to this thesis. Appendix A.1 reports the results of the original simplified multi-fidelity regression model, detailed in Chapter 3, for an industrial problem related to flame positioning in hydrogen burners. Appendix A.2 illustrates the preliminary results of an original mapping approach based on deep geometrical encoding. Here, the purpose is to mitigate the intrinsic limitations of ROMs due to geometrical parameterizations.

Chapter 2

Reduced order models

Reduced order models (ROM) are a family of models that aim to approximate the solution of a Full Order Model (FOM). Generally, an FOM is a numerical model of Partial Differential Equations (PDEs) defined over a discretized domain, which solutions are typically discretized fields. For example, in Computational Fluid Dynamics (CFD) simulations, we might be interested in the pressure and velocity FOM solutions, and, similarly, this concept applies to other fields as well. Once trained, ROMs can perform quasi-real time predictions and approximate the FOMs with remarkable accuracy and robustness.
ROMs have demonstrated versatility through its application in diverse disciplines, not only in CFD [1, 2] , but also in experimental fluid-dynamics [3, 4], structural analysis, and many other fields [5, 6].

Usually, a ROM exploits a compression technique to reduce the dimensionality of the problem, thus describing the solutions with a low-dimensional state vector. Historically, this is achieved through Proper Orthogonal Decomposition (POD), which, nowadays, is still the most established methodology. Indeed, multiple works with POD-based ROMs showed great results [7, 8]. Other approaches involve more complex techniques, such as Dynamic Mode Decomposition (DMD), or non-linear methods as Variational Auto-Encoders (VAE) [9, 10]. This work mainly addresses POD-based ROMs.

When the dimensionality reduction technique has access to the problem PDEs, as in Galerkin or Petrov-Galerkin projections [8, 11], a ROM is called intrusive. Instead, when a ROM acts as a black-box and the model does not have any knowledge of the governing equations, the ROM is called non-intrusive (NI). Usually, the implementation of an NI ROM involves machine learning (ML) regression models or other interpolation techniques.

This Chapter will present how POD works in Section 2.1, how intrusive ROMs exploit POD in Section 2.2, it discusses NI ROMs in Section 2.3, which are the models this thesis is focused on, and finally, we provide an example in Section 2.4.

2.1 Proper orthogonal decomposition

POD, also known as Karhunen-Loeve Expansion, is able to describe a high-dimensional dataset [12, 13] through a set of orthogonal basis functions, in practice, encoding the data. Formally, these basis functions are obtained as the solution of an optimization problem to capture most of the energy norm associated with the basis.

Let the dataset $\mathbf{S} = \{\mathbf{u}_1, ..., \mathbf{u}_n\}$ be a set of vectors, with $\mathbf{u}_i \in \mathbb{R}^N \, \forall \, i \in [1, ..., n]$, and N being the domain discretization cardinality where $\mathbf{u}$ is defined, e.g., the computational grid when $\mathbf{u}$ is a CFD model's solution. In the context of POD, $\mathbf{u}$ is referred to as snapshot, and the set $\mathbf{S}$ can be rearranged as a matrix in $\mathbb{R}^{N \times n}$, where each column corresponds to a snapshot. Hence, $\mathbf{S}$ is also called the snapshots matrix.

Successively, an eigenproblem for $\mathbf{S}$ is solved, often through a Singular Value Decomposition (SVD). A set $\mathbf{\Psi} = \left\{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_{n_{modes}}\right\}$ of orthonormal linearly independent generators of $\mathbf{S}$ is found

$$\mathbf{\Psi} \xleftarrow{\text{SVD}} \mathbf{S}\,, \tag{2.1}$$

so that for all the basis elements $\boldsymbol{\psi}_j \, \forall \, j = 1, ..., n_{modes}$, called modes, is true that

$$\mathbf{S}\mathbf{S}^T \boldsymbol{\psi}_j = \lambda_j^2 \boldsymbol{\psi}_j\,, \tag{2.2}$$

where λ_j is the singular value of $\mathbf{S}$ associated to the j-th mode $\boldsymbol{\psi}_j$, and n_{modes} is the number of modes obtained with the SVD. Being a mode $\boldsymbol{\psi}$ a basis for the snapshots matrix $\mathbf{S}$, a mode and a snapshot $\mathbf{u}$ have the same dimensionality N; therefore, $\boldsymbol{\psi} \in \mathbb{R}^N$.

A common practice is to truncate the basis, reducing the number of modes to r. Often the truncation leads to significantly fewer modes than snapshots, so that $r < n_{modes} \leq n$. For this truncation, different criteria exist, but the most common approach is to sort in decreasing order all the modes from the most energetic one and put a threshold to the POD energy ε, defined as

$$\varepsilon(r) = \frac{\sum_{j=1}^{r} \lambda_j^2}{\sum_{j=1}^{n_{modes}} \lambda_j^2}\,. \tag{2.3}$$

The reduced number of modes r is set so that $\varepsilon(r) \leq$ threshold. Even though truncating the basis $\mathbf{\Psi}$ means to waste some potential information, the lowest energy modes are associated with high-frequency behaviors, generally related to data noise. Thus, the POD basis truncation is often beneficial.

Different linear algebra techniques other than SVD can be used to find the orthonormal basis functions for the POD, such as eigenvalue decomposition or rank-revealing QR factorization. The most widely used one in literature is the SVD, and, in this thesis, we will perform POD only through SVD.

With the POD modes, it is possible to encode the snapshots. To do so, the snapshots have to be projected on the reduced basis $\mathbf{\Psi} = \{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_r\}$, obtaining a scalar value a for each snapshot and for each mode, called POD coefficient. Let's consider a snapshot $\mathbf{u}_i \in \mathbf{S}$ and the j-th mode $\boldsymbol{\psi}_j$, the corresponding POD coefficient is

$$a_j^i = \mathbf{u}_i \cdot \boldsymbol{\psi}_j \,, \tag{2.4}$$

where $\cdot$ represents the scalar product in $\mathbb{R}^N$ used for projection. Hence, the POD representation of the snapshot $\mathbf{u}_i$ on the POD basis $\mathbf{\Psi}$ is given by the vector $[a_1^i, ..., a_r^i]$. Since, in general, $r \ll N$, where N is the snapshots' cardinality, the POD operates a substantial compression of the information with a controlled trade-off on the accuracy. From the POD coefficients and the POD modes, it is possible to compute an approximation of the snapshot with their linear combination

$$\mathbf{u}_i \sim \mathbf{u}_i^* = \sum_{j=1}^{r} a_j^i \boldsymbol{\psi}_j \,. \tag{2.5}$$

With the approximation in Equation 2.5, the POD projection error e_{prj} can be computed

$$e_{prj} = \frac{\|\mathbf{u}_i - \mathbf{u}_i^*\|_2}{\|\mathbf{u}_i\|_2} \,, \tag{2.6}$$

where $\|\cdot\|_2$ is the Euclidean norm in $\mathbb{R}^N$. This error metric measures the POD encoding capabilities: the lower the projection error, the better the POD can describe the snapshots. This metric can be evaluated both on the snapshot matrix utilized for the POD, as well as on any other out-of-sample snapshot, allowing for understanding how the POD behaves on unseen snapshots.

2.2 Intrusive reduced order models

Intrusive ROMs are mostly used for time-dependent problems, and they require access to the governing equations. There are different approaches, and historically, the POD-Galerkin method was widely utilized, especially for CFD problems. Through POD Galerkin projection, it is possible to find a low-dimensional system of Ordinary Differential Equations (ODE) that approximates the FOM problem in the space spanned by the POD basis functions [14, 15]. To illustrate the general idea behind POD-Galerkin methods, let us consider the 1D heat equation

$$\frac{\partial \mathbf{u}(x,t)}{\partial t} = \nu \nabla^2 \mathbf{u}(x,t)\,, \tag{2.7}$$

where here $\mathbf{u}$ represents the temperature, x the spatial coordinate, t the time, and ν the diffusivity. If we have multiple snapshots for different values t, it is possible to expand, similarly to Equation 2.5, the temperature solution $\mathbf{u}$ with the POD

$$\mathbf{u}(x,t) \sim \sum_{j=1}^{r} a_j(t)\boldsymbol{\psi}_j(x)\,. \tag{2.8}$$

This representation of $\mathbf{u}$ can be substituted into Equation 2.7, obtaining

$$\frac{\partial}{\partial t}\left(\sum_{j=1}^{r} a_j(t)\boldsymbol{\psi}_j(x)\right) = \nu\nabla^2\left(\sum_{j=1}^{r} a_j(t)\boldsymbol{\psi}_j(x)\right)\,, \tag{2.9}$$

and further simplifying, it results that the heat equation can be written as

$$\sum_{j=1}^{r} \frac{\partial a_j(t)}{\partial t}\boldsymbol{\psi}_j(x) = \nu \sum_{j=1}^{r} a_j(t)\nabla^2\boldsymbol{\psi}_j(x)\,. \tag{2.10}$$

Successively, Equation 2.10 can be projected on the reduced basis $\{\boldsymbol{\psi}_i(x) : i = 1, ..., r\}$

$$\left\langle \sum_{j=1}^{r} \frac{\partial a_j(t)}{\partial t}\boldsymbol{\psi}_j(x),\, \boldsymbol{\psi}_i(x) \right\rangle = \left\langle \nu \sum_{j=1}^{r} a_j(t)\nabla^2\boldsymbol{\psi}_j(x),\, \boldsymbol{\psi}_i(x) \right\rangle\,, \tag{2.11}$$

where $\langle\cdot,\cdot\rangle$ is the projection operator. Keeping in mind the POD basis functions orthogonality, a system of ODEs for the POD coefficients can be obtained

$$\frac{\partial a_i(t)}{\partial t} = \nu \sum_{j=1}^{r} a_j(t)\left\langle \nabla^2\boldsymbol{\psi}_j(x),\, \boldsymbol{\psi}_i(x)\right\rangle \quad \forall\, i \in [1, ..., r]\,. \tag{2.12}$$

This methodology allows for a significant decrease in the online computational cost [16]. Indeed, the FOM is reduced to a system of r ODEs with extremely fewer degrees of freedom than the FOM, thus far less expensive to solve, since $r \ll N$.

Depending on the problem and the sampling of the training snapshots decomposed with the POD, an intrusive ROM can be very accurate and inexpensive. However, the intrusive nature of the model always requires access to the PDEs and to project them with Galerkin projection, making it difficult to implement. Moreover, intrusive ROMs have some applicability constraints, making them well-suited only for specific cases, suffering from stability and robustness issues. As an example, highly turbulent or advection-dominated problems with a time parameterization can present significant challenges.

Often, the applicability of such intrusive models in an industrial context is not straightforward, mostly because there is no guarantee that there is access to the source code of the numerical solver used. Therefore, in this context, non-intrusive approaches are preferred due to their black-box behavior and their ease of use with non-time-dependent parameterizations.

2.3 Non-intrusive reduced order models

Intrusive ROMs, presented in Section 2.2, are not always a viable solution and can present several challenges in their implementation; on the other hand, NI ROMs can overcome such limitations. Indeed, NI ROMs do not require access to the governing PDEs of the FOM. Since NI ROMs act as "black-boxes" and are data-driven techniques, they can be trained on any distributed quantity, even if the snapshots are not solutions of a set of PDEs. The main requirement is that all the snapshots can be represented as a vector $\mathbf{u} \in \mathbb{R}^N$associated with a parameter vector $\mathbf{x} \in \mathbb{R}^d$. Consequently, whatever quantity that is defined on a discretized domain can be considered as a snapshot. For example, in this thesis, NI ROMs have been used to approximate not only full CFD solutions, but also the pressure acting on a boundary surface or force signals. From an industrial point of view, the vectorized quantities of interest might be, or not, PDE solutions, as well as not even deriving from them, e.g., coming from experimental measurements, hence making NI ROMs compelling for such applications.

2.3.1 Hypothesis

Before describing how an NI ROM works, let us establish some hypotheses:

1. all the snapshots $\mathbf{u}$ have the same cardinality N;
2. all the snapshot discretizations are coherent from a topological standpoint;
3. all the snapshots $\mathbf{u}$ are obtained from the same FOM;
4. unseen snapshots of interest are likely to be approximated by a linear combination of the known snapshots.

Hypotheses 1 and 2 are crucial to perform a meaningful POD on the training snapshots $\mathbf{u} \in \mathbf{S}$, which is the first step for any POD-based NI ROM. If the cardinality between two different snapshots changes, it is not possible to perform the POD on the training snapshots. If hypothesis 2 no longer holds because the ordering of the elements between two different snapshots $\mathbf{u}_i, \mathbf{u}_j$ is not consistent, the POD can still be computed; however, it is no longer viable for the ROM. These concerns arise especially for geometrical parameterizations where the shape of the discretized domain can change. As it will be discussed later in Section 2.3.3, the snapshots will need to be defined on a reference discretization prior to computing the POD. Hypothesis 3 means that all the snapshots are characterized by the same fidelity level. As a consequence, we will assume that a generic NI ROM will be a single-fidelity ROM. This is important in this thesis since in Chapter 4 a multi-fidelity version of a POD-based NI ROM will be discussed. Finally, hypothesis 4 does not hinder the POD computation or the NI ROM applicability; however, for their intrinsic nature, POD-based ROMs are fundamentally linear and can have some limitations in that sense.

2.3.2 Model

Similarly to intrusive ROMs, the very first step is to compute the POD of the training snapshots. Each snapshot corresponds to a parameter vector $\mathbf{x} \in \mathbb{R}^d$, e.g., containing some physical properties, operational parameters, geometrical parameters, or boundary condition; therefore, $\mathbf{S}$ can be rewritten as $\mathbf{S} = \{\mathbf{u}(\mathbf{x}_1), ..., \mathbf{u}(\mathbf{x}_n)\}$. The POD can be computed on $\mathbf{S}$, obtaining a set of POD modes $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_r\}$. The POD coefficients a_j^i are obtained, as in Equation 2.4, projecting the snapshot $\mathbf{u}_i$ onto the $\boldsymbol{\psi}_j$ mode. Since each snapshot is associated to a parameter $\mathbf{x}$, also the training POD coefficients a_j^i will be associated to $\mathbf{x}$

$$a_j^i = \mathbf{u}(\mathbf{x}_i) \cdot \boldsymbol{\psi}_j = a_j(\mathbf{x}_i)\,. \tag{2.13}$$

Therefore, for each mode $j = 1, ..., r$, we can define a dataset D_j

$$D_j = \{(\mathbf{x}_i, a_j(\mathbf{x}_i)) : i = 1, ..., n\}\ , \tag{2.14}$$

where each element is a couple of a parameter and the corresponding j-th POD coefficient for a given training snapshot. Then, it is possible to find a function $\hat{a}_j(\mathbf{x})$ that approximates the POD coefficient a_j for any unseen parameter value $\mathbf{x}$, fitting an approximation model on D_j. The model for $\hat{a}_j(\mathbf{x})$ can be both a regression or an interpolation model, where the most common ones used for POD-based NI ROMs are

- Radial Basis Functions (RBF);
- Gaussian Processes (GP);
- Neural Networks (NN).

This thesis will focus mainly on GP-based regression models, and Chapter 3 will provide a more in-depth discussion of this topic. In general, RBF interpolation models are fast and can handle larger training datasets with ease, and they work well with very smooth functions. On the other hand, NNs are computationally expensive, but they can manage a certain degree of nonlinearity in the POD coefficient functions. Finally, GPs regression models stand in between these two models and, in addition, they can handle well noise in the training data. From now on, we will consider each approximation model to be a continuous map between the parameter space and the j-th POD coefficient space

$$\hat{a}_j(\mathbf{x}) : \mathbb{R}^d \longrightarrow \mathbb{R}\,. \tag{2.15}$$

Usually, each POD coefficient a_j is treated separately with its own regression model. It is possible to model a single multi-output NN (or eventually a GP); however, POD coefficients associated with different modes are typically uncorrelated since they are obtained through the projection of a snapshot onto orthonormal POD modes.

Once the maps $\hat{a}_1(\mathbf{x}), ..., \hat{a}_r(\mathbf{x})$ have been trained, it is possible to predict an approximation for each POD coefficient for unseen parameters $\mathbf{x}$, allowing to approximate an out-of-sample snapshot

$$\mathbf{u}(\mathbf{x}) \approx \sum_{j=1}^{r} \hat{a}_j(\mathbf{x}) \boldsymbol{\psi}_j \,. \tag{2.16}$$

To evaluate the accuracy of the ROM prediction, the most widely used metric is the reconstruction error e_{rec}

$$e_{rec} = \frac{\left\| \mathbf{u}(\mathbf{x}) - \sum_{j=1}^{r} \hat{a}_j(\mathbf{x}) \boldsymbol{\psi}_j \right\|_2}{\|\mathbf{u}(\mathbf{x})\|_2} \,, \tag{2.17}$$

where e_{rec} measures how well the ROM can approximate a generic snapshot $\mathbf{u}$, taking into account all the accuracy loss introduced by both the POD encoding and the POD coefficient regression models. The reconstruction error formulation in Equation 2.17 is similar to the projection error one in Equation 2.6. Indeed, the reconstruction error can be interpreted as the sum of the projection error and a second component, which we can call interpolation error e_{int}, as in Equation 2.18. As previously stated, the projection error measures the POD capabilities to encode a snapshot with the given POD modes; therefore, the remaining error e_{int} can be attributed only to the POD coefficient approximation effect on the ROM prediction.

$$e_{rec} = e_{prj} + e_{int} \,. \tag{2.18}$$

Remark The prediction of a POD-based NI ROM is obtained as a linear combination of POD modes. Recalling the hypothesis 4 from Section 2.3.1, it is clear that this methodology is not suitable for any given problem. Figure 2.1 illustrates this problem with a naive example.

Here we have two training snapshots, which are 1D Gaussian bell functions centered in two different positions, and their center abscissa is the problem parameter. Conveniently, the two snapshots are quasi-orthogonal[1], so the POD modes are scaled versions of the same two snapshots. If we aim to predict an unseen target snapshot in between the two training snapshots, we can approximate the POD coefficients by linear interpolation. However, the ROM prediction will be way off, as Figure 2.1 shows, and that is because there is no possible linear combination of the 2 POD modes that can approximate well the target snapshot. This behavior is well known, and it is typical of advection-dominated problems. E.g., if we consider a fluid-dynamics time-dependent ROM, it will struggle to describe the movement in time of

[1]Intuitively, when snapshot 1 is almost 0, snapshot 2 is greater than 0 and vice-versa. Therefore, the inner product will be very close to 0.

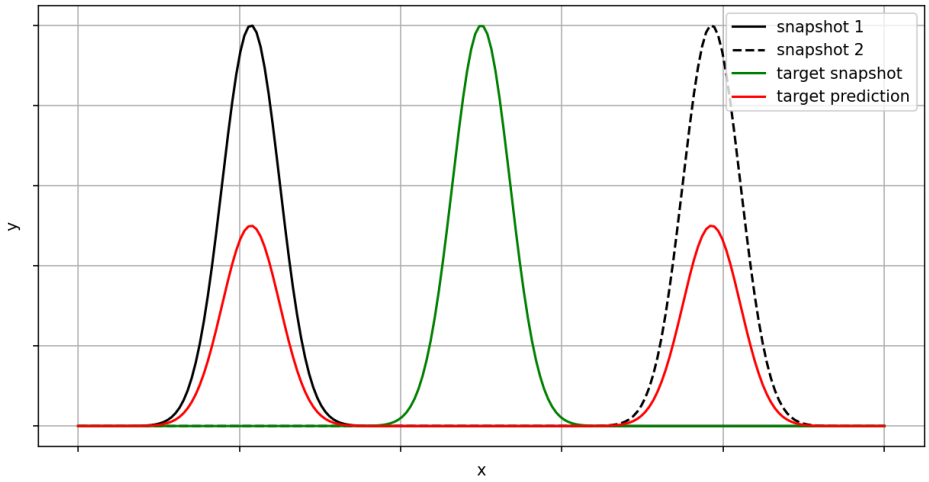


Figure 2.1: POD-based NI ROM example with two quasi-orthogonal snapshots . In black the training snapshots, in green the target snapshot to predict and in red the actual prediction.

the flow structures advected by the velocity field, similarly to what can be observed in the previous example. In general, to mitigate this problem, the only option is to increase the number of meaningful modes until the problem can be linearly approximated. However, this implies increasing the number of training snapshots, making this solution undesirable from a computational cost point of view.

2.3.3 Mapping heterogeneous snapshots

As previously mentioned in Section 2.3.1, when the snapshots are defined on different discretizations, it is not possible to encode the information with POD techniques, making the ROM unfeasible. From a practical standpoint, this usually happens when:

1. two snapshots are obtained from two different geometries;
2. the granularity of the discretization changes, e.g. due to mesh refinements, even if the domain shape remains untouched.

The last point is common when the snapshots are originated from different fidelity solvers, and it will be particularly relevant for the multi-fidelity ROM presented in

Chapter 4. As a consequence, to train a ROM with heterogeneous snapshots, we might need to harmonize their definition on the domain, satisfying hypotheses 1 and 2 of Section 2.3.1 regarding the snapshots' cardinality and coherence. In presence of geometrical deformations, it is possible to satisfy those hypotheses by mapping all the snapshots on a common reference discretization. If there is no geometrical deformation but the discretization cardinality changes, an interpolation of the snapshots on a common reference discretization is sufficient.
Here, mapping refers to the process of transferring a field defined on one discretized domain, i.e., the source, to a different discretized domain, i.e., the reference, which may differ in geometry, mesh topology, and resolution. This process ensures that the source field, associated with the source discretization, is consistently represented on the reference discretization, enabling comparison on a common reference domain. The mapping may involve interpolation or other techniques to account for differences between the source and reference discretizations, while preserving key properties of the source field. In general, the mapping operator should be reversible, meaning that a field defined on the reference discretization can be brought back to the source discretization.

Usually, mapping strategies introduce some error due to the required interpolations. Therefore, if the geometrical parameterization causes small enough deformations, it is possible to use a single reference computational grid, e.g., in the case of CFD or FEM problems, and apply mesh morphing techniques prior to the numerical solution to keep the topology of the mesh intact. However, this is not always possible, since too large deformations might result in invalid meshes. In these cases, the shape deformations should be modeled from a reference surface with a known nonrigid spatial transformation, or utilize registration techniques to learn the geometry's morphing. The nonrigid transformation can be of different types. As an example, in the applications presented in this work, we used both analytical, in Sections 5.2 and 5.4, and free-form deformations-based methods in Section 5.1, to obtain the transformations. The main idea is to exploit one's chosen technique to move the reference discretization to the source surface domain and then interpolate the data from the source node distribution to the morphed reference nodes. Keeping in mind that the morphed reference nodes should be coherent in terms of cardinality and topology with the original reference nodes, the interpolated field can be associated with the reference discretization, too. This can be done for both surfaces and volumes, and many different interpolation techniques exist. To cite a few of them, we have:

- barycentric interpolation;

- k-nearest interpolation;
- RBF interpolation.

The first should be preferred when the source discretization was obtained as a computational grid and the connectivity is available. However, this is not always possible, so the k-nearest or the RBF interpolation can be used to fast interpolate the source field from the source point cloud to the morphed reference one. Between the two of them, the RBF is usually more accurate, but for very large point clouds, the k-nearest is the fastest and most effective solution.

The same interpolation strategies remain valid when there is the need only to interpolate the data between different fidelity discretizations, i.e., from coarse to fine meshes, while not having any geometrical parameterization. The reference mesh can be both a low- or high- fidelity mesh; however, using a low-fidelity reference can lead to precious high-fidelity information loss.

The error introduced by the mapping is usually negligible if compared to the ROM reconstruction error, but should be taken into consideration for high-accuracy applications. Hence, the reconstruction error components in Equation 2.18 become

$$e_{rec} = e_{prj} + e_{int} + e_{map}\,, \tag{2.19}$$

where e_{map} represents the mapping error.

Since mapping techniques should be specifically tailored to each application, utilizing more general approaches can be compelling. For both intrusive and non-intrusive ROMs, registration methods showed promising results for geometrical problems [17, 18]. In Appendix A.2, we discuss a preliminary attempt to obtain a data-driven registration model based on the DeepSDF architecture [19].

2.4 A non-intrusive ROM example

In this section, we present an application of a single-fidelity ROM for a heat exchanger shape optimization. The aim is mostly pedagogical, showing how a ROM can be integrated into a real-life industrial workflow. This also allows us to highlight the main challenges for these kinds of problems. As we previously stated, ROMs can be an efficient solution when there is a need to perform a high number of numerical simulations, and optimization is a fitting example.

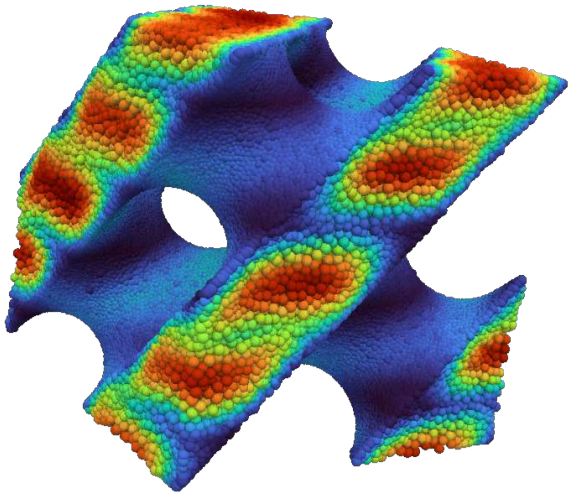


Figure 2.2: Qualitative example of the velocity magnitude field inside of a TPMS unit cell. Spheres represent the meshless solver computational nodes.

We consider a geometrical parameterization of a Triply Periodic Minimal Surface (TPMS), and we simulate its thermo-fluid dynamics with a meshless solver [20]. Recently, TPMS geometries are getting more interest due to the new additive manufacturing techniques, and their high surface-to-volume ratio, which is promising for heat exchanger applications. We combine non-intrusive ROMs with the meshless solver to enable efficient multi-objective shape optimization. Figure 2.2 shows an example of a TPMS velocity field on the node distribution of a meshless solver.

This work has been developed for the international conference EUROGEN 2025, on evolutionary and deterministic methods for design, optimization, and control with applications.

Problem description

We consider a Schwarz-D TPMS, defined as the level set of a trigonometric function, as in

$$s_x s_y s_z + s_x c_y c_z + c_x s_y c_z + c_x c_y s_z = 0\,, \tag{2.20}$$

with $s_t = \sin\left(\frac{\pi t}{L}\right)$, $c_t = \cos\left(\frac{\pi t}{L}\right)$, and L being the TPMS size. The geometrical parameterization has two parameters, and it is applied using the first two eigenfunc-

tion of the surface Laplacian, here noted as $\boldsymbol{\psi}_1, \boldsymbol{\psi}_2$. The surface points are displaced along the surface normals of a quantity $\delta(\mathrm{x}_1, \mathrm{x}_2)$ as in

$$\delta(\mathrm{x}_1, \mathrm{x}_2) = \sum_{i=1}^{2} \mathrm{x}_i \frac{\boldsymbol{\psi}_i}{\|\boldsymbol{\psi}_i\|_\infty}\,, \tag{2.21}$$

where $\mathrm{x}_1, \mathrm{x}_2$ are the actual geometrical parameters which scale the displacement contribution of each normalized eigenfunction. This parameterization will be convenient since the surface displacement is known and can be easily propagated to the internal points if needed.

The physical model is based on the incompressible Navier-Stokes and energy equations for an incompressible and laminar problem, as in Equations 2.22-2.24.

$$\nabla \cdot \mathbf{U} = 0\,, \tag{2.22}$$

$$\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \cdot \nabla)\mathbf{U} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{U}\,, \tag{2.23}$$

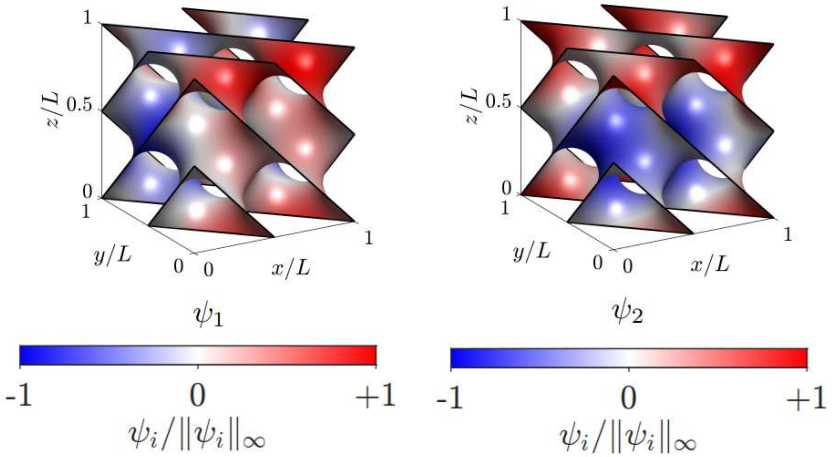
$$\frac{\partial T}{\partial t} + \mathbf{U} \cdot \nabla T = \alpha \nabla^2 T\,, \tag{2.24}$$

where $\mathbf{U}$ is the velocity field, p the pressure field, T the temperature field, ρ the fluid density, ν the kinematic viscosity, α the thermal diffusivity, and t the time. The Reynolds number and Prandtl number are considered constant, having $Re = 100$ and $Pr = 0.71$, respectively.

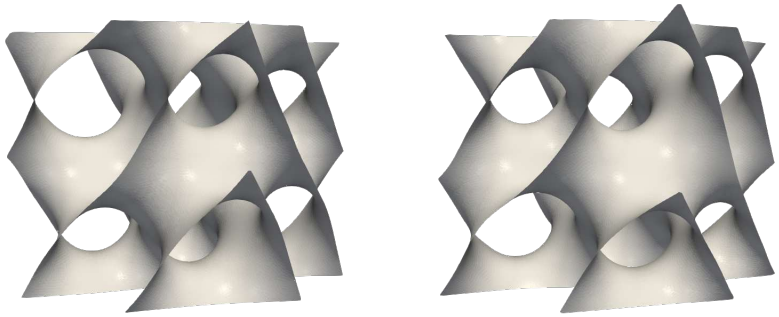
From the solution fields, we can compute two quantities of interest, namely the Fanning friction factor, or coefficient, f, and the Nusselt number Nu, which will be the objectives of the optimization. The Fanning coefficient is defined as

$$f = \frac{2\tau}{\rho U^2}\,, \tag{2.25}$$

where τ is the wall shear stress and this coefficient is a dimensionless indicator of the fluid resistance at the wall.



(a) First eigenfunction. (b) Second eigenfunction.



(c) Extreme design 1. (d) Extreme design 2.

Figure 2.3: Deformation's eigenfunction representations (a, b) and resulting deformations for two extreme designs in the input parameter space (c, d).

The Nusselt number quantifies the convective heat exchange when compared to the conductive exchange, and it is defined as

$$Nu = \frac{hL}{k}\,, \tag{2.26}$$

where h is the the convective heat transfer coefficient, L is the characteristic length and k is the thermal conductivity.

Snapshot generation

The principles of the Radial Basis Function - Finite Differences (RBF-FD) meshless solver used are discussed in [21, 20, 22], but the main idea is that the solution fields are defined on a set of nodes scattered inside the computational volume, without defining a mesh connectivity. Since the geometry is triply periodic, periodic boundary conditions are applied to every intersection with the bounding planes, alongside a pressure gradient in one of the directions to force the flow. This ensures that all snapshots have the same cardinality, regardless of the applied deformation. The input parameter space sampling for the ROM training database has been done with a Halton sequence of 150 designs. This allows us to evenly cover the input domain with designs, while having the possibility to consider meaningful subsets of designs from the original 150.

The main advantage of the meshless solver for geometrically parameterized problems lies in the flexibility when applying significant deformations. Since we need all dimensionally coherent snapshots for every design configuration, a reference node distribution is deformed according to the displacement of the boundary nodes. This is easily achieved by prolongation of the displacement field, solving the Laplacian of the displacements. This can be done with mesh-based solvers too, but it is not trivial to satisfy the quality criteria for the mesh, nor is it to manage the periodicity of the mesh after the deformation. Meshless approaches alleviate these problems, since there is no concept of node connectivity. An interested reader can find further details regarding the physical and meshless characterization of the considered Schwarz-D TPMS in [20].

Workflow

A typical non-intrusive ROM workflow is divided into two parts: the offline phase and the online phase. In the offline part, the training snapshots are generated, and the ROM is trained. CFD simulations can be time-consuming, and the creation of the database is the most computationally heavy task, but it needs to be done only once. In this experiment, we train two ROMs, one for the pressure solution p and one for the velocity solution U. The offline part of the workflow for this application is represented in Figure 2.4. Note that there is no intermediate mapping between the solver and the snapshots, since we preserve the solution cardinality for any given parameter.

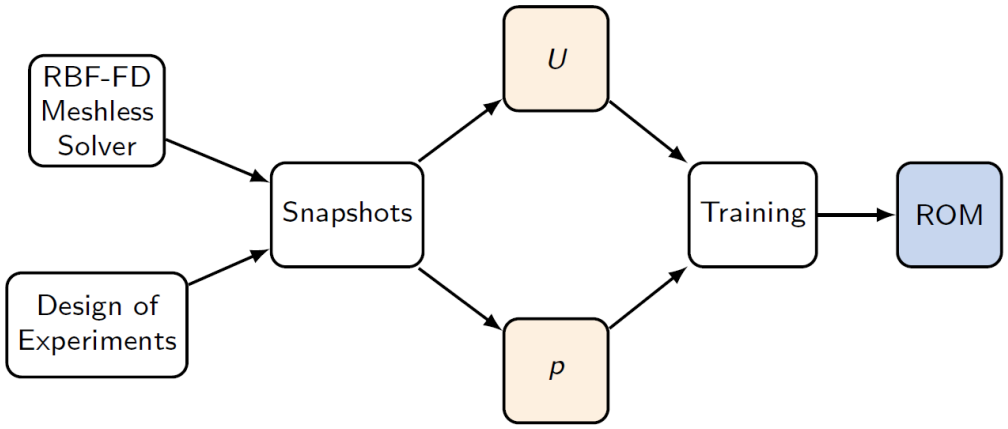


Figure 2.4: Offline phase in the ROM study for the TPMS optimization.

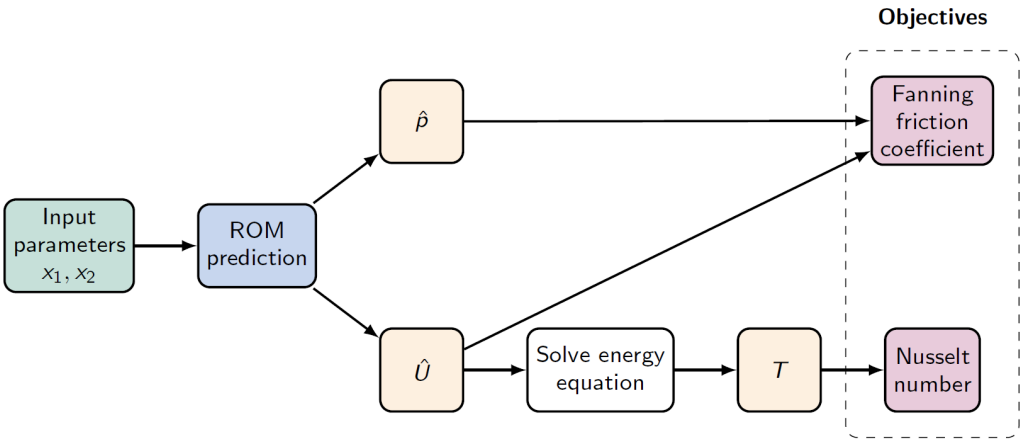


Figure 2.5: Online ROM phase for an optimizer call for the TPMS optimization problem.

Once the ROMs are trained, it is possible to query the models to predict an approximation of the fields of interest, $\hat{p}, \hat{\mathbf{U}}$ for any design configuration. This is done at the online phase, which consist in a linear combination of POD modes and some inference on the POD coefficient models. Thanks to this, the computation is particularly cheap and can be done in real-time. In Figure 2.5 we detail how the online phase is structured for an optimization workflow.

In Figure 2.5 there are extra steps and not just the ROM predictions. Often, the predictions of a ROM are successively post-processed to compute some quantities of interest. Here we use the predictions to compute f and Nu which are the optimization objectives. In this example, the solution of the energy equation, which requires the steady state solution for the velocity $\mathbf{U}$, is relatively cheap, so we did not surrogate the temperature field T, and, instead, we solve for the energy equation given the ROM approximation $\hat{\mathbf{U}}$.

Optimization

To perform a comparison, two multi-objective optimizations have been carried out: a classical CFD-based optimization and a ROM-based one. In the classical approach, the CFD solver is utilized for every design evaluation. On the other hand, the ROM-based optimization relies on a trained ROM for the fields of interest, having all the computational cost limited to the offline phase.

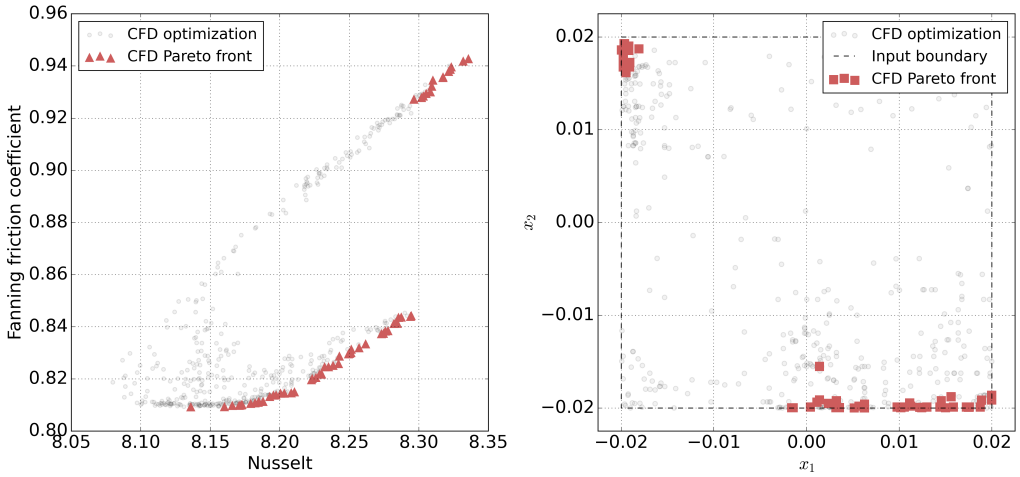
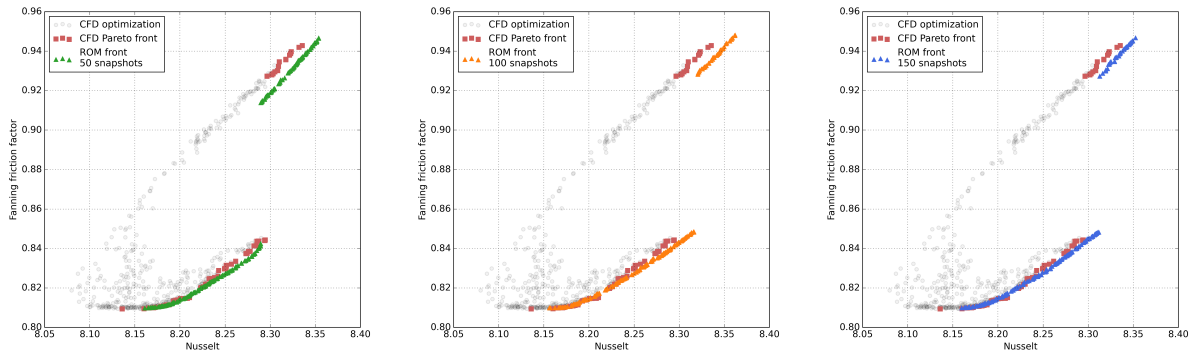


Figure 2.6: Pareto front (right) and relative input locations (left) for the benchmark CFD-based optimization.

For both optimizations the NSGA-II [23] multi-objective genetic algorithm has been utilized, aiming to minimize f while maximizing Nu. Here, the classical optimization is used as a benchmark for the ROM-based one. A total of 20 generations have been considered, each of 25 individuals, obtaining a final cost of 500 design evaluations for the benchmark optimization. In Figure 2.6, the optimal values belonging to the Pareto front and their design configurations are presented. The optimizer identifies a two-part front, which corresponds to two different input domain zones.

We then trained three ROMs: one with 50 training snapshots, one with 100 and one with 150 snapshots. The resulting Pareto fronts are represented in Figure 2.7. The ROM trained with the highest number of snapshots, 150 designs, is the closest one to the benchmark CFD results. There is a slight overestimation of the Nusselt number, having the ROM Pareto fronts consistently shifted to the right by a small amount, while the Fanning coefficient approximation are more in line with the CFD. If we compare the results in the front locations for the best ROM, all Nu relative errors are below 1%, and depending on the required accuracy it can be an acceptable approximation.



(a) 50 training snapshots. (b) 100 training snapshots. (c) 150 training snapshots.

Figure 2.7: Pareto fronts for different ROM training sizes compared to the benchmark CFD front.

In this pedagogical example, we did not augment the training dataset size further, since with 150 designs we were able to retrieve a close approximation of the optimal locations in the input parameter space. This can be seen in Figure 2.8, where we compare all the front locations in the design space for both the CFD optimization and the ROM-based optimizations.

If we put together both the CFD-based optimization and the best ROM-based optimization, we obtain the results in Figure 2.9. It is possible to notice a clear superposition in the optimal location prediction, and also a good agreement between the values in the output space. A single design in the CFD Pareto front does not belong to the optimal portion of the input domain identified by the ROM-based optimization. This is because that specific design was dominated by the ROM front by a small margin, due to the accuracy trade-off introduced with the ROM.

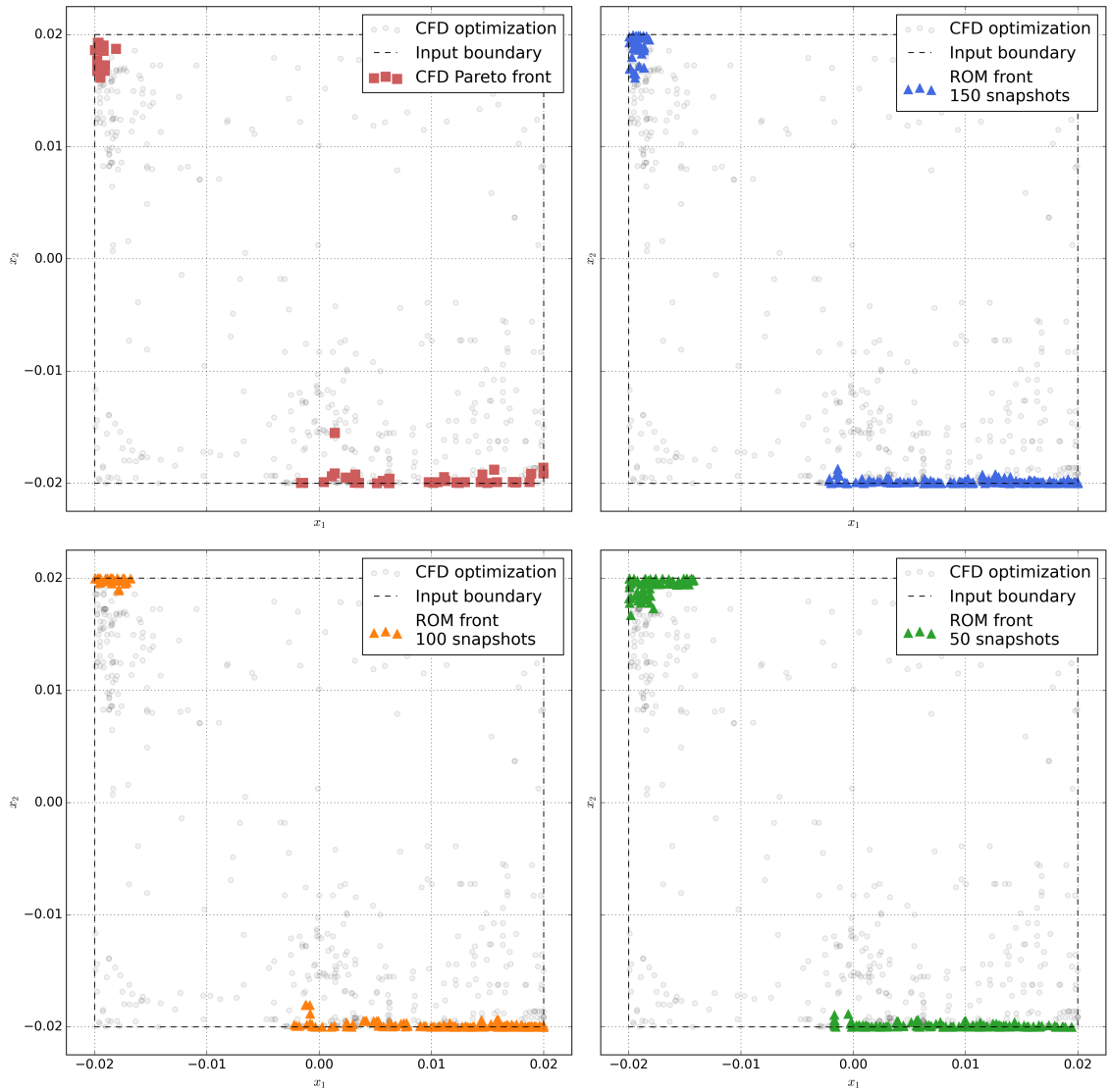


Figure 2.8: Pareto front locations in the input parameter space. From top left in clockwise order: benchmark CFD-based optimization, ROM-based optimization with 150, 50, 100 training snapshots, respectively.

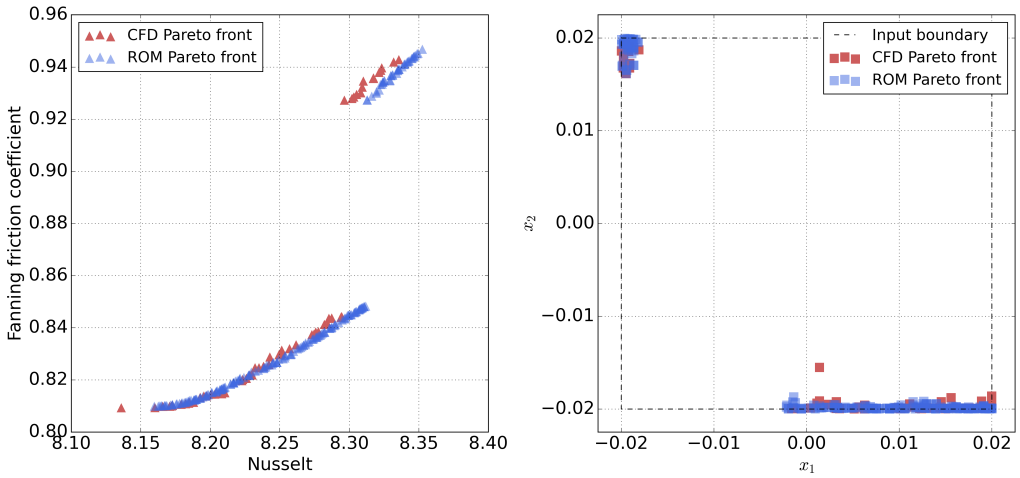


Figure 2.9: Pareto front comparison (left) of the CFD-based optimization and the ROM-based optimization, and respective input parameter locations (right).

Discussion

The ROM-based optimization has similar results to the classical CFD-optimization in this multi-objective shape optimization problem. Moreover, an acceptable approximation of the Pareto front required 150 FOM evaluations, instead of the 500 FOM evaluations for the CFD-based optimization. This translates to a 3 times computational cost reduction thanks to the ROM.

The chosen problem is very specific, but it offers an example of how ROMs can be helpful in a plausible industrial application. Indeed, optimization is a perfect example of an engineering problem that requires to perform multiple numerical simulations. Moreover, it highlights how the offline phase represents a bottleneck for ROMs, deeply affecting the overall computational cost.

Chapter 3

Regression models

Regression models aim to approximate an unknown function based on a finite set of observations, mapping a parameter $\mathbf{x} \in M \subseteq \mathbb{R}^d$ to a scalar value $y \in \mathbb{R}$. These models are typically applied to relatively small datasets, which makes them well-suited for estimating concentrated quantities. However, regression models tend to be less effective and/or not feasible when dealing with high-dimensional datasets, such as solution fields obtained from numerical simulations, where ROMs are often preferred. In particular, non-intrusive ROMs, described in Section 2.3, make extensive use of regression-based models to be able to approximate distributed quantities. Moreover, the ROM's performance and capabilities are influenced by the model choice and the underlying regression model configurations. In this thesis, various regression models will be studied for their application to ROMs, with a primary focus on distinguishing between single-fidelity and multi-fidelity models, and emphasizing GP-based regression models. Single-fidelity interpolation and regression models are the most common approximation model. Typical examples are Kriging, also known as Gaussian Process Regression [24], Radial Basis Function interpolations [25, 26], and artificial Neural Networks [27, 28]. When heterogeneous data sources are available, multi-fidelity regression models can be used. Most of them are based on GPs, as Cokriging, Hierarchical Kriging or NARGP [29, 30, 31, 32], while others make use of deep learning architectures, such as in [33, 34, 35].

This Chapter will present single-fidelity models in Section 3.1, specifically GP regression in Section 3.1.1, followed by alternative methods in Section 3.1.2. Then, Section 3.2 will cover multi-fidelity models, starting from the NARGP in 3.2.1 and presenting other multi-fidelity approaches in Section 3.2.2, including an original multi-fidelity model tailored for severe data scarcity problems. Finally, Section 3.3

outlines a possible preprocessing of the data before the regression.

3.1 Single-fidelity regression

Most of the more widely used regression models are single-fidelity models. Indeed, single-fidelity regression models leverage information from homogeneous sources, meaning that all the training data originates from the same model or experiment. As an example, if we take multiple experimental measurements of a quantity of interest, and the experiment remains the same for each sample, the obtained data has a single level of fidelity. Similarly, with a numerical problem, i.e., CFD, if all the simulations were computed with the same solver, the same mathematical modeling, and a comparable number of mesh nodes, the obtained data can be considered single-fidelity too. However, if we mix heterogeneous data sources, e.g., numerical simulations with fine and coarse meshes, the data has multiple fidelity levels, making it unsuitable for single-fidelity regression models.

Let us have an unknown function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, a set of n known input parameters $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, of which we know the outputs $y_i = f(\mathbf{x}_i)\ \forall i = 1, ..., n$. The set of pairs $D = \{(\mathbf{x}_i, y_i) : i = 1, ..., n\}$ is called the training dataset, and it collects the input-output couples used for the regression problem training. A regression model will be a map $\mathcal{M} : \mathbf{x} \to y$ that fits the training dataset D and approximates the unknown function f estimating the relationship between inputs and outputs

$$\mathcal{M}(\mathbf{x}) \approx f(\mathbf{x})\,. \tag{3.1}$$

Different approaches can be used to find $\mathcal{M}$, ranging from interpolation to Bayesian and deep learning methods. In this thesis, we will focus principally on GP regression, but other approaches such as RBF or NNs are widely used in the context of NI ROMs [16, 36]. In general, any regression model able to accurately approximate d-dimensional arbitrary continuous functions can be a viable solution. Therefore, this work will not cover simpler regression models, e.g., linear or polynomial regressions, or discontinuous approaches such as tree-based regression models.

To compare different regression models, the models' performance is evaluated through training and validation errors. The first one measures how accurate the model predictions are in the training locations. On the other hand, the validation error measures how accurate the model predictions are far away from the training locations, requiring a second dataset D_{val} of unseen input parameter configurations.

In general, the training error is expected to be lower than the validation error, and for noise-free training data, it should be close to 0. The training error is used mostly to monitor the training process. The validation error, instead, is the most relevant metric, especially when the cardinality $|D_{val}|$ is high enough to be statistically relevant. The lower the validation error, the better the model is expected to behave for out-of-sample predictions.

3.1.1 Gaussian process regression

Gaussian Processes

GP regression, also known as Kriging, is based upon GPs [24], a particular type of stochastic process [37]. Thus, the stochastic process and GP definitions become fundamental to understanding GP regression models.

Definition 3.1 – *A* stochastic process $\mathbf{A} = \{A_t : t \in T\}$ *is a collection random variables* A_t *indexed by a set of indices* T. □

Definition 3.2 – *A* Gaussian process *is a collection of random variables, any finite number of which have a joint Gaussian distribution.* □

A GP is uniquely defined by a mean function $m(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ and a covariance function $k(\mathbf{x}, \mathbf{x}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, also called a kernel, which determinates the GP's joint distribution. Similarly to how a random variable can be used to describe a distribution over scalar values, a GP can be used to describe a distribution over functions. Let's say that $\mathbf{f}$ is a GP with mean $m(\mathbf{x})$ and kernel $k(\mathbf{x}, \mathbf{x}')$; the GP can be written as

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \,, \tag{3.2}$$

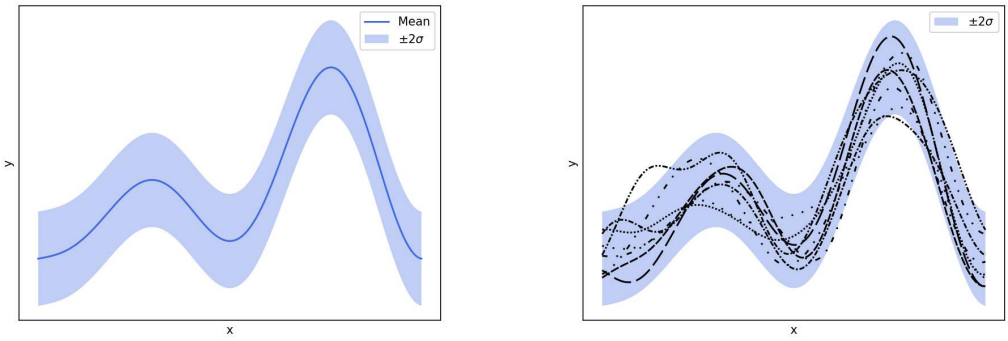
having

$$m(\mathbf{x}) = \mathbb{E}\left[\mathbf{f}(\mathbf{x})\right] \,, \tag{3.3}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[\left(\mathbf{f}(\mathbf{x}) - m(\mathbf{x})\right)\left(\mathbf{f}(\mathbf{x}') - m(\mathbf{x}')\right)\right] \,, \tag{3.4}$$

where $\mathbb{E}[\cdot]$ is the expected value. Figure 3.1 gives an intuition of the GP function space interpretation. From a GP, represented in Figure 3.1(a), it is possible to

sample from the GP any number of realizations, as in Figure 3.1(b), each one representing a different function. As the number of samples increases, the mean of the realizations approaches the GP mean, and similarly does the GP standard deviation, as illustrated in Figure 3.1 through the confidence interval.



(a) GP mean and confidence interval (b) GP random samples and confidence interval

Figure 3.1: GP function space interpretation. (a) GP mean and confidence interval as ± 2 the standard deviation and (b) GP confidence interval and 8 random realizations sampled from the GP.

If we consider a finite set of random variables $\mathbf{Y} = \{Y_1, ..., Y_m\}$ from the GP $f(\mathbf{x})$ corresponding to the input parameters $\{\mathbf{x}_1, ..., \mathbf{x}_m\}$, by definition they have a joint normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)\,, \tag{3.5}$$

where $\mathcal{N}$ indicates a multivariate normal distribution, and $\boldsymbol{\mu}, \Sigma$ are defined as

$$\boldsymbol{\mu} = \left[m(\mathbf{x}_1), ..., m(\mathbf{x}_m)\right]\,, \tag{3.6}$$

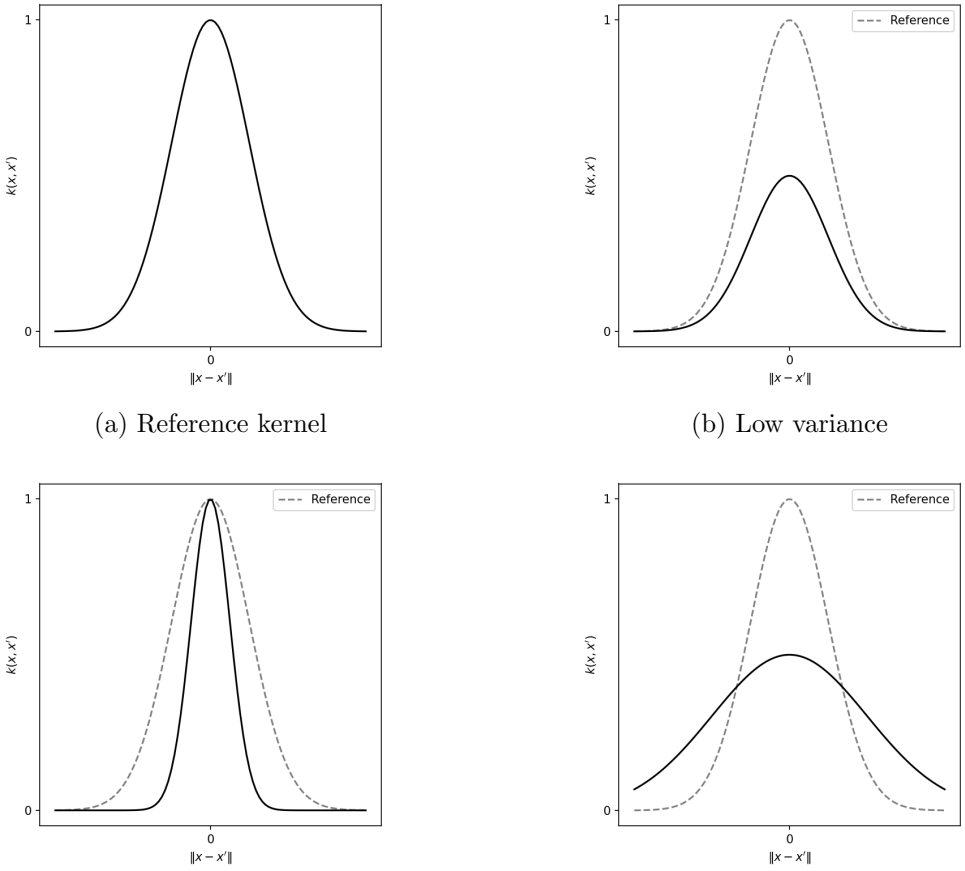
$$\Sigma = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & ... & k(\mathbf{x}_1, \mathbf{x}_m) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & ... & k(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_m, \mathbf{x}_1) & k(\mathbf{x}_m, \mathbf{x}_2) & ... & k(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}\,. \tag{3.7}$$

To have a valid kernel for a GP, it is required that the resulting covariance matrices Σ are semi-definite positive. A classical kernel function is the Squared Exponential

(SE), also known as the Gaussian or RBF kernel, and its formulation can be found in Equation 3.8.

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right), \tag{3.8}$$

where $\|\cdot\|$ is a norm e σ^2 and l are two GP hyperparameters defining its shape, respectively the variance and the lengthscale. The variance modulates the intensity of the kernel, while the lengthscale changes the width of the kernel, and their effect on the kernel shape can be seen in Figure 3.2.



(a) Reference kernel

(b) Low variance

(c) Low lengthscale

(d) Low variance, high lengthscale

Figure 3.2: Effect of variance and lengthscale hyperparameters on the kernel shape. (a) reference kernel shape, (b) low variance kernel, (c) low lenghtscale kernel and (d) low variance and high lengthscale.

According to Equation 3.8, the kernel assigns the same covariance to equally spaced points, disregarding their orientation if the input parameter space $\mathbb{R}^d$ is not one-dimensional. If this happens, the resulting kernel is called isotropic. However, in multi-dimensional problems, with $d > 1$, it is possible to assign different lengthscales to each input parameter dimension, making the kernel anisotropic. This can be achieved through automatic relevance determination [24] (ARD), which, instead of using the classical Euclidean norm $\|\mathbf{x} - \mathbf{x}'\|$ to compute the distance between two points $\mathbf{x}, \mathbf{x}'$, it uses a weighted Euclidean norm $\|\mathbf{x} - \mathbf{x}'\|_M$, where M is a semi-definite positive matrix in $\mathbb{R}^{d\times d}$. Specifically, for ARD M is a diagonal matrix, with non-negative values, and the distance between $\mathbf{x}, \mathbf{x}'$ in the weighted norm becomes

$$\|\mathbf{x} - \mathbf{x}'\|_M = \sqrt{(\mathbf{x} - \mathbf{x}')M(\mathbf{x} - \mathbf{x}')^T} = \sqrt{\sum_{i=1}^{d} l_i(\mathrm{x}_i - \mathrm{x}'_i)^2}\,, \tag{3.9}$$

where the matrix M diagonal elements depend on the lengthscales l_i corresponding to each input dimension

$$M = \begin{bmatrix} 1/l_1^2 & 0 & \dots & 0 \\ 0 & 1/l_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/l_d^2 \end{bmatrix}. \tag{3.10}$$

Using the norm defined in Equation 3.9 into the SE kernel of Equation 3.8, it becomes

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|_M^2\right). \tag{3.11}$$

Moreover, an isotropic kernel can be seen as a special case where $M = 1/l^2 I$, with I being the identity matrix in $\mathbb{R}^{d\times d}$.

One of the most commonly used kernels for GP regression is the SE; however, some others are frequently used. For this thesis, we will consider only the Matérn class of kernel (Matérn 3/2 and Matérn 5/2) and the rational quadratic (RQ) kernel, as in Equations 3.12 - 3.14, or their combinations. Indeed, sums, products, or convolutions of valid kernels are valid kernels too, allowing to obtain more complex kernels combining preexisting ones.

$$k_{Mat3/2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \sqrt{3} r_M\right) \exp\left(-\sqrt{3} r_M\right), \tag{3.12}$$

$$k_{Mat5/2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \sqrt{5} r_M + \frac{5 r_M^2}{3}\right) \exp\left(-\sqrt{5} r_M\right), \tag{3.13}$$

$$k_{RQ}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{r_M^2}{2\alpha}\right)^{-\alpha}, \tag{3.14}$$

where $r_M = \|\mathbf{x} - \mathbf{x}'\|_M$ for simplicity. All of these kernels are stationary, meaning that the kernel depends on the relative distance between two points only and not their position. Usually, non-stationary kernels are avoided for GP regression tasks in ROM applications.

Kriging model

A GP regression model [24], often called Kriging [38], is a Bayesian approach to modeling that aims to approximate an unknown function $f(\mathbf{x})$ with a GP. Specifically, the actual function estimate is given by the mean of the GP, while the GP variance is used to estimate the uncertainty associated with the mean prediction.

Training Previously, we said that a GP is fully defined by its mean and covariance functions; however, in a regression problem, they are not known. Therefore, with the training of the GP regression model, the aim is to find the best mean and kernel functions given the available training data. From a practical standpoint, the optimal mean and kernel function are obtained by solving a maximum likelihood estimation (MLE) problem, which will allow us to find the best GP's hyperparameters configuration. In particular, the hyperparameters are parameters that define the model's structure, and for GP regression models, they consist of the kernel ones (lengthscales, variance, ...) and other parameters associated with noise or the mean function. Hereafter, the set of GP hyperparameters will be denoted as $\boldsymbol{\theta}$.

First of all, without loss of generality, the GP prior mean function is assumed to be $m(\mathbf{x}) = \mathbf{0}$, making the kernel shape pivotal for the GP training. Successively, we need to formalize the MLE problem. Given the training dataset $D = \{(\mathbf{x}_i, y_i) : i = 1, ..., n\}$, the marginal likelihood $p(\mathbf{y}|X, \boldsymbol{\theta})$ measures the probability p of the observed $\mathbf{y} = \{y_i : i = 1, ..., n\}$, given the inputs $X = \{\mathbf{x}_i : i = 1, ..., n\}$ and a GP with hyperparameters $\boldsymbol{\theta}$. The marginal likelihood for a GP is known and has the following log formulation

$$\log p(y|X,\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K^{-1}(X,X)\mathbf{y} - \frac{1}{2}\log(\det K(X,X)) - \frac{1}{2}\log(2\pi)\,, \tag{3.15}$$

where $K(X,X)$ is the covariance matrix obtained from the kernel function $k(\mathbf{x},\mathbf{x}')$ computed for the $\mathbf{x} \in X$ and $\mathbf{x}' \in X$. The log marginal likelihood is particularly convenient since it is relatively easy to differentiate with respect to the hyperparameters $\boldsymbol{\theta}$. It should be kept in mind that even if it has an analytical solution, the derivative will still require computing at least once the inversion of the covariance matrix $K(X,X)$, which is $\mathcal{O}(n^3)$; therefore, it can be demanding for high cardinality training datasets. The differentiability of $\log p(y|X,\boldsymbol{\theta})$ is well suited for gradient-based optimization algorithms, which can be used to solve the MLE problem. The optimization becomes the minimization of the negative marginal log likelihood and allows for finding the best hyperparameters $\boldsymbol{\theta}^*$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left(-\log\left(\mathbf{y}|X,\boldsymbol{\theta}\right)\right). \tag{3.16}$$

Generally speaking, the optimization is relatively fast, however $\log\left(p(\mathbf{y}|X,\boldsymbol{\theta})\right)$ can have different local optima, and some robust approaches can be applied to mitigate their effect. Typically, random restarts of the gradient-based optimization represent a fast and effective solution.

Being a probabilistic approach that optimizes the data likelihood, it is fundamentally different from other ML models, such as NNs. In fact, the reader should keep in mind that the GP regression model "loss" does not represent how well the model fits the data, in contrast to NN regression models, but how likely it is that the data can be described by a GP. At the end of the model training, the kernel shape and, eventually, all other hyperparameters are known, and the model can be used to approximate the target function for unseen input parameters.

Prediction Given a set of unseen input parameters $X^* = \{\mathbf{x}_i^* : i = 1, ..., n^*\}$, the model aims to predict the corresponding unknown outputs $\mathbf{y}^* = \{y_i^* : i = 1, ..., n^*\}$. Given the training input parameters X, the training outputs $\mathbf{y}$, a zero-mean prior, and a kernel k defined by the optimal hyperparameters $\boldsymbol{\theta}^*$, if both $\mathbf{y}, \mathbf{y}^*$ belong to the same GP, they have by definition a joint Gaussian distribution

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{y}^*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X,X) & K(X,X^*)\\ K(X^*,X) & K(X^*,X^*)\end{bmatrix}\right), \tag{3.17}$$

being $K(X,X) \in \mathbb{R}^{n\times n}, K(X,X^*) \in \mathbb{R}^{n\times n^*}, K(X^*,X) \in \mathbb{R}^{n^*\times n}, K(X^*,X^*) \in \mathbb{R}^{n^*\times n^*}$ rectangular covariance matrices obtained with the kernel $k(\mathbf{x}.\mathbf{x}')$. However, the conditional distribution of $\mathbf{y}^*$ given $\mathbf{y}$ for a prior multivariate Gaussian distribution is known. Therefore the posterior $\mathbf{y}^*$ becomes

$$\mathbf{y}^*|X^*, X, \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}^*,\, \Sigma^*\right) , \tag{3.18}$$

where $\boldsymbol{\mu}^*$ is the posterior mean, as in Equation 3.19, and Σ^* is the posterior covariance matrix, as in Equation 3.20. In particular, the posterior mean $\boldsymbol{\mu}^*$ will be the actual model prediction for the unknown $\mathbf{y}^*$ values, and the diagonal of Σ^* stores the variances associated with each predicted value, necessary to assign a confidence interval to the prediction.

$$\boldsymbol{\mu}^* = K(X^*,X)K(X,X)^{-1}\mathbf{y} , \tag{3.19}$$
$$\Sigma^* = K(X^*,X^*) - K(X,X^*)K(X,X)^{-1}K(X^*,X) . \tag{3.20}$$

The results in Equations 3.18 - 3.20 come from a property of multivariate Gaussian distributions [24]. Given $\mathbf{a}, \mathbf{b}$ two jointly Gaussian vectors with the following distribution

$$\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}A & C\\ C^T & B\end{bmatrix}\right) , \tag{3.21}$$

then the conditional distribution of $\mathbf{a}$ given $\mathbf{b}$ is still a joint Gaussian distribution with the following mean and covariance matrix

$$\mathbf{a}|\mathbf{b} \sim \mathcal{N}\left(\boldsymbol{\mu}_a + CB^{-1}(\mathbf{b}-\boldsymbol{\mu}_b),\, A - CB^{-1}C^T\right) . \tag{3.22}$$

Thus, given the prior in Equation 3.17, the GP posterior in Equation 3.18 comes from Equations 3.22.

In Equation 3.17, there is a noise-free assumption; however, a GP regression model can handle noisy training data, meaning that the output $y_{noisy} = y + \varepsilon$, with ε here is a random variable with 0 mean representing the noise. Since this noise is not known, a possibility is to assume that it is distributed as a Gaussian, and the covariance function can be modeled as

$$k(\mathbf{x}_p, \mathbf{x}_q) \to k(\mathbf{x}_p, \mathbf{x}_q) + \delta_{pq}\sigma_n^2 , \tag{3.23}$$

where δ_{pq} is a Kronecker delta and σ_n^2 is the variance of the Gaussian noise, which will be an additional hyperparameter of the GP regression model. This implies that the GP prior of Equation 3.17 becomes

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right), \tag{3.24}$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix. This has an effect too on the posterior distribution, turning Equations 3.19 and 3.20 into

$$\boldsymbol{\mu}^* = K(X^*,X)[K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y}\,, \tag{3.25}$$

$$\Sigma^* = K(X^*,X^*) - K(X,X^*)[K(X,X) + \sigma_n^2 I]^{-1}K(X^*,X)\,. \tag{3.26}$$

Since the σ_n^2 hyperparameter is learned during training, the prediction in Equations 3.25 and 3.26 takes into account the noise effect and returns a de-noised mean with an increased variance.

In general, GP prediction is easy and fast to compute; however, when the prediction locations X^* become huge in size, some memory problems could arise. Indeed, the calculator needs to obtain and store at least the matrix $K(X^*,X) \in \mathbb{R}^{n^* \times n}$ to predict the posterior mean in Equation 3.19, and its dimension scales linearly with n^*. Therefore, GP models are not well-suited for large concurrent predictions, unlike ROMs, which are designed for such tasks.

3.1.2 Other single-fidelity models

Other single-fidelity regression models can be employed in POD-based NI ROMs, and the most frequent are RBF interpolation models and NN regression models.

Radial basis function interpolation

RBF interpolation models are often used to approximate continuous function $f(\mathbf{x})$ from an unstructured dataset $D = \{(\mathbf{x}_i, y_i) : i = 1, ..., n\}$ [26, 39, 21], which points will be also called RBF nodes. $f(\mathbf{x})$ is modeled as a sum of functions $\xi_i(\mathbf{x})$ with $i = 1, ..., n$, called RBFs. Each RBF depends on the distance between $\mathbf{x}$ and one of the RBF nodes, in particular

$$\xi_i(\mathbf{x}) = \eta(\|\mathbf{x} - \mathbf{x}_i\|_2) = \eta(r)\,, \tag{3.27}$$

where η is a RBF function, such as those presented in Table 3.1, and $r = \|\mathbf{x}-\mathbf{x}_i\|_2$ here is the distance between $\mathbf{x}, \mathbf{x}_i$ measured with the Euclidean norm $\|\cdot\|_2$. In the $\eta(r)$ formulations of some RBFs in Table 3.1, i.e., the multiquadric, the Gaussian, or the inverse multiquadric, ϵ is a model hyperparameter that controls the shape of the RBFs.

RBF name	**RBF function** $\eta(r)$
Linear	r
Multiquardic	$\sqrt{1+(\epsilon r)^2}$
Thin-plate spline	$r^2 \log r$
Cubic	r^3
Inverse multiquadric	$(1+(\epsilon r)^2)^{-1/2}$
Gaussian	$e^{-(\epsilon r)^2}$

Table 3.1: Most frequent radial basis functions formulations.

The RBF interpolation model will then be a linear combination of the radial basis functions

$$f(\mathbf{x}) \approx \sum_{j=1}^{n} \alpha_j \xi_j(\mathbf{x}) \,, \tag{3.28}$$

where α_i are the unknown combination coefficients. To compute α_i, for $i = 1, ..., n$, Equation 3.28 is enforced for each training point in D

$$\sum_{j=1}^{n} \alpha_j \xi_j(\mathbf{x}_i) = y_i \,, \tag{3.29}$$

leading to a linear system such as

$$\Xi \boldsymbol{\alpha} = \mathbf{y} \,, \tag{3.30}$$

where $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_n\}$, $\mathbf{y} = \{y_1, ..., y_n\}$ and the RBF kernel matrix $\Xi \in \mathbb{R}^{n \times n}$ is obtained computing the RBFs $\eta(\mathbf{x})$, centered in each training node, with respect to each other training node

$$\Xi = \begin{bmatrix} \xi_1(\mathbf{x}_1) & \xi_2(\mathbf{x}_1) & ... & \xi_n(\mathbf{x}_1) \\ \xi_1(\mathbf{x}_2) & \xi_2(\mathbf{x}_2) & ... & \xi_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \xi_1(\mathbf{x}_n) & \xi_2(\mathbf{x}_n) & ... & \xi_n(\mathbf{x}_n) \end{bmatrix} . \tag{3.31}$$

An appropriate RBF $\eta(\mathbf{x})$ ensures that the linear system in Equation 3.30 has a unique solution if and only if the RBF kernel matrix is definite positive for any set of distinct training node locations $\{\mathbf{x}_i : i = 1, ..., n\}$. The RBF formulations in Table 3.1 are all valid RBFs and are the most utilized ones. In particular, when using those functions characterized by the shape parameter ϵ, special care should be taken, since improper values of ϵ can make Ξ ill-conditioned. Also, interpolations on too large datasets can lead to an ill-conditioned Ξ matrix. Finally, the system in Equation 3.30 can be solved for $\boldsymbol{\alpha}$, finalizing the training of the RBF interpolation model.

From a practical standpoint, the RBF model in Equation 3.28 is augmented with a polynomial term, to improve accuracy and stability

$$f(\mathbf{x}) \approx \sum_{j=1}^{n} \alpha_j \xi_j(\mathbf{x}) + \sum_{k=1}^{m} \beta_k p_k(\mathbf{x}) \,, \tag{3.32}$$

with $m = \binom{d+P}{P}$, where P is the polynomial degree hyperparameter, β_k are the unknown polynomial coefficients and $p_k(\mathbf{x})$ are all the polynomial's monomials up to degree P. Under the hypothesis that the RBF coefficients $\boldsymbol{\alpha}$ are orthogonal to each $p_k(\mathbf{x})$ in the training nodes

$$p_k(\mathbf{x}_i) = \boldsymbol{\alpha} \cdot [p_k(\mathbf{x}_1), ..., p_k(\mathbf{x}_n)] = 0 \quad \forall k = 1, ..., m \,, \tag{3.33}$$

it is possible to rewrite the linear system of Equation 3.30 taking into account the polynomial augmentation

$$\begin{bmatrix} \Xi & \Pi \\ \Pi^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \,, \tag{3.34}$$

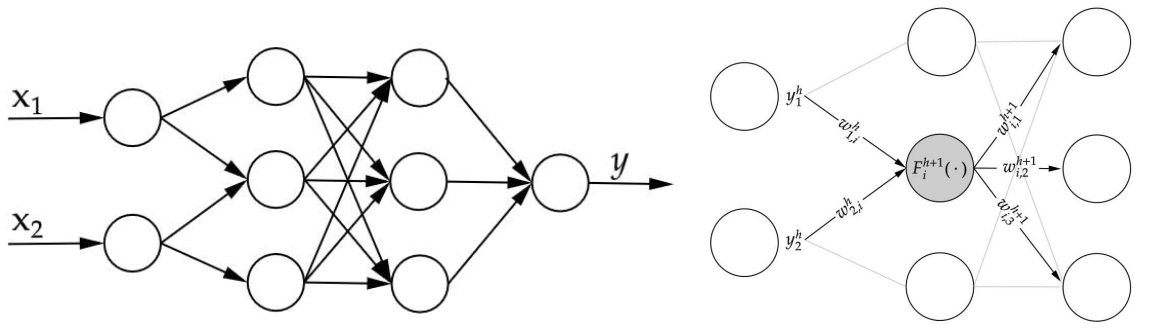
where 0 here is a $m \times m$ null matrix, $\mathbf{0} \in \mathbb{R}^m$ and the matrix $\Pi \in \mathbb{R}^{n \times m}$ derives from the polynomial part of Equation 3.34, being

$$\Pi = \begin{bmatrix} p_1(\mathbf{x}_1) & p_2(\mathbf{x}_1) & ... & p_m(\mathbf{x}_1) \\ p_1(\mathbf{x}_2) & p_2(\mathbf{x}_2) & ... & p_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\mathbf{x}_n) & p_2(\mathbf{x}_n) & ... & p_m(\mathbf{x}_n) \end{bmatrix} \,, \tag{3.35}$$

and simultaneously determine the RBF coefficients $\boldsymbol{\alpha}$ and the polynomial coefficients $\boldsymbol{\beta}$ solving the system in Equation 3.34. Once $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are known, Equation 3.32 can be used to approximate $f(\mathbf{x})$ in any point $\mathbf{x}$ of the domain.

Neural network regression

Similar to a GP regression or an RBF interpolation, an NN model [28] aims to approximate an unknown function $f(\mathbf{x})$, modeling a nonlinear relationship between the input variables and the output variables. The model structure is inspired by the human brain, and it is composed of neurons and weighted connections between those nodes, similar to the synapses in the nervous system. This surrogate model has been extended to non-intrusive POD-based ROMs too [36].



(a) Neural network example. (b) Focus on a single neuron connections.

Figure 3.3: (a) Example of a simple NN for a 2D regression; arrows and circles represents respectively the connections and the neurons. (b) Information flow from the h-th layer to the i-th neuron (in grey) of the $h+1$-th layer of a NN.

Figure 3.3 illustrates the main components of an NN model. A NN is composed of neurons and connections, and the neurons are organized in layers. There are 3 types of layers: input, hidden, and output. The input layer, as the name suggests, receives the input information. In a regression problem, the input consists of the vector of independent variables $\mathbf{x} \in \mathbb{R}^d$, and to each dimension $i = 1, ..., d$ there is associated one neuron. Successively, there might be one or more "hidden" layers, whose number and dimension are model's hyperparameter, and they connect the input and the output layers. Finally, there is the output layer, composed of as many neurons as the dimension of the output y. Without loss of generality, here we consider only one-dimensional scalar outputs $y \in \mathbb{R}$; therefore, the output layer will have a single

neuron.

Each neuron has some connections that can be directed to the neuron, or from the neuron itself, eventually in a recursive manner, carrying the NN information. The neuron acts as an aggregation operator that manipulates and propagates the information. Typically, all the information directed to the neuron is aggregated by sum, and then used as an input for an activation function $F(x) : \mathbb{R} \to \mathbb{R}$. This activation function introduces a certain degree of nonlinearity, and many functions can be used. The most common activation functions can be found in Table 3.2. If we consider the i-th neuron of the $h+1$-th layer of a NN, as in Figure 3.3 (b), the information y_i^{h+1} exiting the neuron is obtained as

$$y_i^{h+1} = F_i^{h+1}\left(\sum_{j=1}^{n_h} w_{j,i}^h y_j^h\right), \tag{3.36}$$

where F_i^{h+1} is the neuron's activation function, $w_{j,i}^h \in \mathbb{R}$ is the weight associated to the connection between the j-th neuron of the h-th layer to the i-th neuron of the $h+1$-th layer, and y_j^h is the output of the j-th neuron of the h-th layer. Successively, y_i^{h+1} will become the input to another neuron and so on. This process, starting from the input layer, is called feed-forward propagation, and it is used to evaluate the output of the NN model. However, the weights are not known a priori; thus, before utilizing the model, it is necessary to determine their optimal values.

Name	**Activation function** $F(x)$
Linear	x
Rectified linear unit	$\max(0, x)$
Hyperbolic tangent	$\tanh(x)$
Sigmoid	$(1 + e^{-x})^{-1}$

Table 3.2: Most frequent neural network activation functions.

Once the number and the dimension of the hidden layers have been decided, as well as the activation function, it is possible to train the NN. The training process consists of finding the best set of weights w, and this is done with a metric called the loss function. The loss function measures how close the NN output is to the actual true value in the training points, i.e., if we call $\hat{f}(\mathbf{x})$ the NN predicted output and $f(\mathbf{x})$ the ground truth, the loss function L will be in the form

$$L(X) = \|\{f(\mathbf{x}) - \hat{f}(\mathbf{x}) : \mathbf{x} \in X\}\|, \tag{3.37}$$

where $\|\cdot\|$ is a suitable norm and X is the set of the training input parameters. Table 3.3 shows some of the most common loss functions used in NN regression models.

Name	**Loss function** $L(X)$
Mean absolute error	$n^{-1}\sum_{i=1}^{n}\lvert f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i)\rvert$
Mean squared error	$n^{-1}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i))^2$
Root mean squared error	$(n^{-1}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i))^2)^{-1/2}$

Table 3.3: Most frequent neural network loss function for regression.

The training starts with a random initialization of the weights, and each weight $w_{i,j}^h$ has an initial value $w_{i,j}^h(0)$. Then, with an iterative procedure, the weight is updated

$$w_{i,j}^h(t+1) = w_{i,j}^h(t) + \Delta w_{i,j}^h(t). \tag{3.38}$$

The update $\Delta w_{i,j}^h(t)$ is determined with a technique called backpropagation of error. Indeed, with the backpropagation it is possible to compute the gradient of the loss function $\frac{\partial L}{\partial w_{i,j}^h}$ with respect to the weights and allow to find the best $\Delta w_{i,j}^h(t)$ $\forall i, j, h$ through a gradient-based optimization algorithm. After some iterations t, also known as epochs, the loss L is expected to decrease, and the training is stopped according to a preselected criterion, such as the loss reaching a threshold, a small relative variation, or a certain number of epochs. In the end, the weights $w_{i,j}^h$ should be the optimal weights for the NN, and they can be used to approximate y for unknown values of the input parameter $\mathbf{x}$.

NNs tend to be more expensive to train compared to RBF or even GP regression models. Similar to GPs, NNs are very sensitive to hyperparameters, considering both the hidden layers' hyperparameters and the choice of the activation and loss function. Often, NNs suffer from overfitting if the choice of the hyperparameters is poor. Therefore, during training, the loss of an out-of-sample set of points is monitored to avoid such problems.

3.2 Multi-fidelity regression

In the previous sections, it emerges that a single-fidelity model's accuracy is strictly dependent on which and how many data points belong to the training dataset. In general, the greater the cardinality of the training dataset, the higher the accuracy of the model. As a consequence, if each data point has a cost associated with it, whether it is time, computational resources, or an economic cost, a more accurate regression model is more expensive. Moreover, if regression models are used to approximate unknown functions, it is not uncommon for the available training data to be scarce. This is particularly important when the data has been obtained from a high-fidelity (HF) data source. An information source, such as a computational model or an experimental measurement, can have a higher or a lower fidelity when compared to another information source, where the fidelity refers to how close the source is to a reference ground truth, and, typically, HF representations are more costly than equivalent low-fidelity (LF) ones. Thus, when training a regression model, the aim is to obtain the most accurate results without sacrificing too many resources for HF data.

First of all, let us consider a CFD example to understand how to determine fidelity levels. Intuitively, a Reynolds-averaged Navier-Stokes (RANS) solution on a coarse mesh will provide a less accurate representation of the flow if compared to a large eddy simulation (LES). One would conclude that RANS simulations are LF, while LES are HF. However, a LES solution will be less accurate than a Direct Numerical Simulation (DNS), therefore making the DNS the HF and the LES a LF model this time. Here, LES can be at the same time a LF and HF model, and it follows that the low- and high-fidelity concept is always relative to which models are considered. Moreover, in the case of CFD, the fidelity hierarchy is not always that clear. Sometimes a cheap potential flow simulation can outperform in accuracy a way more expensive RANS one, while most of the time this does not happen. Or, sometimes, a DNS simulation can be considered more adherent to reality than an experimental result due to measurement error. Hence, a model can be both HF and LF even when compared to the same counterpart, and this is extendable to many other fields and applications. Therefore, it is crucial to understand that an HF model exists only when compared to a LF one, and special care should be used when choosing which model is closer to what we consider the ground truth.

Multi-fidelity models aim to reduce the overall training cost by leveraging information from heterogeneous sources and provide a framework for integrating models

and/or experimental data with different fidelity levels. By doing so, a multi-fidelity model reduces the amount of HF data necessary for the training by exploiting inexpensive LF information.

Multi-fidelity models can take advantage of LF data in many different ways. For simplicity, we will consider only 2 fidelity levels; however, many multi-fidelity models can make use of 3 or more fidelity levels. There are two main approaches to multi-fidelity [40]:

1. Corrective;

2. Space mapping and non-linear models.

A corrective multi-fidelity model is a rather linear one, since it assumes that the HF function to be a scaled and/or corrected version of the LF one. Called $y_{HF}(\mathbf{x}), y_{LF}(\mathbf{x})$ the target HF function and the corresponding LF function respectively, a comprehensive corrective approach is structured as

$$y_{HF}(\mathbf{x}) \approx \rho(\mathbf{x}) y_{LF}(\mathbf{x}) + \delta(\mathbf{x}) \,, \tag{3.39}$$

where $\rho(\mathbf{x})$ represents a multiplicative correction factor, and $\delta(\mathbf{x})$ is the additive correction factor, sometimes referred to as discrepancy. If $\rho(\mathbf{x}) = 1$, the correction will be purely additive, leading to the simpler model

$$y_{HF}(\mathbf{x}) \approx y_{LF}(\mathbf{x}) + \delta(\mathbf{x}) \,, \tag{3.40}$$

or, if $\delta(\mathbf{x}) = 0$ the correction is purely multiplicative

$$y_{HF}(\mathbf{x}) \approx \rho(\mathbf{x}) y_{LF}(\mathbf{x}) \,. \tag{3.41}$$

The model structures presented in Equations 3.39 - 3.41 can be particularly performing when the correlation between LF and HF data is close to linear, or it can be described with a simple function. In general, this structure is used for relatively simpler models, and one of the most widely used ones is the Cokriging model [29, 30].

However, when the correlation between LF and HF data is more complex, the corrective multi-fidelity models can struggle. To overcome this limitation, more nonlinear approaches can be used. In [40] models that operate a transformation of the $y_{LF}(\mathbf{x})$ as in

$$y_{HF}(\mathbf{x}) \approx F(y_{LF}(\mathbf{x})) \,, \tag{3.42}$$

are called space mappings, where F here is a transformation between the two fidelity output spaces. However this structure can be generalized even further obtaining a more nonlinear model

$$y_{HF}(\mathbf{x}) \approx F(\mathbf{x}, y_{LF}(\mathbf{x}))\,, \tag{3.43}$$

where the transformation F is a mapping between the input space domain, augmented by the LF output, to the HF output space. This mapping can be modeled in many different ways, where the most common ones are based on GPs or artificial NNs. One example is the Nonlinear AutoRegressive multi-fidelity Gaussian Process (NARGP) regression model [32]. Nonlinear approaches can be applied to linear problems too; however, they are usually heavier models and with more hyperparameters, making them less effective in extreme HF data scarcity.

In this work, we will utilize the NARGP model for multi-fidelity regressions. In the multi-fidelity problem of POD coefficients, it is not straightforward to know what the type of correlation between fidelities is, as it will be better discussed in Chapter 4; thus, the NARGP can be more effective than other multi-fidelity models due to its flexibility.

3.2.1 NARGP

NARGP [32] can be seen as an evolution of Cokriging models able to handle nonlinear relationships between low- and high-fidelity data. This model has a recursive structure, similar to the recursive Cokriging formulation in [30], and, eventually, can be used with any number of fidelity levels. Without loss of generality, we will consider only 2 fidelity levels, LF and HF, respectively. To simplify further, we will consider the HF training points to be nested to the LF ones, even though it is not mandatory

$$X_{LF} = \{\mathbf{x}_i : i = 1, ..., n_{LF}\}\,, \tag{3.44}$$
$$X_{HF} = \{\mathbf{x}_j : j = 1, ..., n_{HF}\} \subset X_{LF}\,, \tag{3.45}$$
$$n_{LF} \gg n_{HF}\,, \tag{3.46}$$

where X_{LF}, X_{HF} are the input parameters for the LF and HF training sets, and n_{LF}, n_{HF} are the respective cardinalities.

Training

The first step to train a NARGP model is to fit a single-fidelity regression model onto the LF data, which will be noted as $\hat{y}_{LF}$, with a GP regression

$$\hat{y}_{LF}(\mathbf{x}) \approx y_{LF}(\mathbf{x})\,. \tag{3.47}$$

Successively, the LF GP model can be evaluated in the HF training locations, obtaining the LF process posterior mean

$$\boldsymbol{\mu}_{LF}^{HF} = \hat{y}_{LF}(X_{HF})\,, \tag{3.48}$$

and $\boldsymbol{\mu}_{LF}^{HF}$ is used to enrich the input to the second GP model that composes the NARGP. This second model will be used to predict the HF output function and will take as input

$$XX_{HF} = \left[X_{HF},\, \boldsymbol{\mu}_{LF}^{HF}\right] \in \mathbb{R}^{n_{HF}\times(d+1)}\,. \tag{3.49}$$

As a consequence, the HF GP trained on the database $D = \{(\mathbf{xx}_j, y_{HF}(\mathbf{x}_j)) : j = 1, ..., n_{HF}\}$ will handle the multi-fidelity data with a special kernel. In particular, the kernel k_{HF} will be a combination of kernels, each one active on part of the input $\mathbf{xx}$, as in

$$k_{HF}\left((\mathbf{x}, \mu_{LF}), (\mathbf{x}', \mu'_{LF})\right) = k_\rho(\mathbf{x}, \mathbf{x}') \cdot k_f(\mu_{LF}, \mu'_{LF}) + k_\delta(\mathbf{x}, \mathbf{x}')\,, \tag{3.50}$$

where μ_{LF}, μ'_{LF} are the LF posterior means in the respective locations $\mathbf{x}, \mathbf{x}'$, and k_ρ, k_f, k_δ are valid kernel functions with dimension d for k_ρ, k_δ and dimension 1 for k_f. Here, the LF information has its own kernel k_f, while the other two kernels depend only on the independent variables. As already mentioned in Section 3.1.1, sums and products of valid kernels are valid kernels too.

Once the kernel formulation k_{HF}, the input data XX_{HF}, and the output data y_{HF} are determined, it is possible to train the HF NARGP model as any other GP regression model, solving a MLE problem. Even though the kernel in Equation 3.50 resembles the corrective approach formulation of Equation 3.39, the structure of the model is not a corrective one. During the MLE, the model will find the optimal hyperparameters for the NARGP without any prior assumption on the HF-LF correlation relationship, unlike what happens for the corrective approaches. The multiplicative correction factor $\rho(\mathbf{x})$ in Equation 3.39 imposes a predetermined structure on the HF-LF correlation. In general, this allows for reducing the number of hyperparameters; however, it reduces the capabilities of the model. Hence, the NARGP is

inherently a flexible model in terms of multi-fidelity correlations, and, in addition, it has a convenient property related to its kernel. In general, a kernel has a variance hyperparameter σ^2 and a lengthscale l. If, during the MLE, the variance of k_ρ or k_f goes to zero, i.e. $\sigma_\rho^2 \to 0 \vee \sigma_f^2 \to 0$, the HF NARGP kernel becomes

$$k_{HF}\left((\mathbf{x}, \mu_{LF}), (\mathbf{x}', \mu'_{LF})\right) = \cancelto{0}{k_\rho \cdot k_f} + k_\delta = k_\delta(\mathbf{x}, \mathbf{x}')\,, \tag{3.51}$$

and now the HF NARGP model is independent to the contribution μ_{LF} of the LF model and the kernel is equal to k_δ, thus making the HF model equivalent to a single-fidelity GP on the HF dataset. This happens when there is no relationship between HF and LF information, or the likelihood gradients show that the contribution of $k_\rho k_f$ is not beneficial to the MLE. This property becomes relevant when there is no certainty that there is a LF-HF relation, and, as it will be discussed in Chapter 4, it is particularly convenient for the POD coefficient regression problem.

To understand the high complexity of the NARGP model, it can be compared to an equivalent single-fidelity GP model. If we suppose that k_ρ, k_f, k_δ are all anisotropic SE kernels with ARD, as in Equation 3.11, and we consider a noise-free problem, the number of hyperparameters of the NARGP is up to $d+1$ for the NARGP's LF model (d lengthscales, 1 variance), and $2d+4$ for the NARGP's HF ($2d+1$ lengthscales, 3 variances). Since the first $d+1$ LF model hyperparameters should be straightforward to obtain, the actual hyperparameters of interest are the NARGP HF model $2d+4$ ones. On the other hand, the equivalent single-fidelity GP model will have $d+1$ hyperparameters. And given that the number of HF data points n_{HF} is usually low, optimizing through MLE a higher number of hyperparameters can be more difficult. Sometimes constraining the hyperparameters, especially the lengthscale ones, can help the MLE solution to converge to the target optimum, avoiding unwanted local optima during the optimization. However, the higher complexity is what enables this multi-fidelity model, together with the LF data, to improve the HF model accuracy.

Prediction

Once the NARGP has been trained, 2 models are available: the LF and the HF GP models. The advantage of the NARGP is that it can exploit at the prediction stage the LF GP predictions. Indeed, it is possible to effortlessly evaluate the LF GP model for a set of unknown input values X^* and obtain the LF output prediction $\hat{y}_{LF}(\mathbf{x}^*)\ \forall \mathbf{x}^* \in X^*$, or in a more compact notation $\hat{y}_{LF}(X^*)$. Similarly to what has been done during the training phase, a composed input vector is obtained as

$$XX^* = [X^*, \hat{y}_{LF}(X^*)] \,, \tag{3.52}$$

the input vector XX^* is now compatible with the kernel k_{HF} of Equation 3.50 and the GP prediction can be evaluated. If the training was successful and there is a learnable relationship between LF and HF data, the addition of the LF prediction to the NARGP HF model should improve the accuracy of the model, if compared to a single-fidelity one. However, since there could be some uncertainty associated with the LF model, the prediction should take it into account. To do so, we first sample over the LF GP posterior and, instead of using the mean as the GP prediction, we consider a set of realizations from the multivariate Gaussian distribution. Then, each of the LF realizations is used as the LF component of XX^* in Equation 3.52 and the final prediction is obtained as a statistic over those HF predictions. By doing so, the uncertainty from the LF level can be propagated to the HF prediction. In particular, the HF stochastic process mean and the variance, according to the law of total variance, become

$$\boldsymbol{\mu}_{HF} = \mathbb{E}[\hat{y}_{HF}(X^*)] \,, \tag{3.53}$$

$$\boldsymbol{\sigma}^2_{HF} = \mathbb{E}[\hat{\sigma}^2_{HF}(X^*)] + \mathbb{V}[\hat{y}_{HF}(X^*)] \,, \tag{3.54}$$

where the expected value $\mathbb{E}$ and the second order statistical moment $\mathbb{V}$ are considered over the LF realizations, $\hat{y}_{HF}(\mathbf{x})$ and $\hat{\sigma}_{HF}(\mathbf{x})$ are the NARGP models for the HF GP mean and the variance, and $\boldsymbol{\mu}_{HF}$ will be the actual NARGP prediction, with the associated uncertainty in form of variance $\boldsymbol{\sigma}^2_{HF}$. It should be noted that the NARGP HF mean and variance posteriors are not necessarily multivariate Gaussian distributions [32]. Moreover, this Monte Carlo approach used at the prediction stage has a limited impact on the overall computational cost if and only if the number of fidelity levels remains modest. Indeed, the total number of samples drawn from the various GPs is exponential: if q is the number of realizations sampled per fidelity level, and s is the number of fidelity levels, then the total number of samples will be q^s.

3.2.2 Other multi-fidelity models

In general, the most widely used multi-fidelity model is the Cokriging model [29, 30]. This model is well established and tested, showing good results, especially for close to linear correlation between fidelities and requiring modest computational resources. On the other hand, more complex and non-linear LF-HF relationships can

be learned with more advanced models, such as multi-fidelity NNs or the previously described NARGP [32] in Section 3.2.1. NNs are, in general, more demanding and complex to train; therefore, they are not ideal for multi-fidelity ROMs, which require the training of many regression models.

This thesis studied simpler approaches to multi-fidelity modeling, too. Indeed, all the previously mentioned multi-fidelity models have a relatively high number of hyperparameters to determine, and they need to be found with a limited number of HF training points, making multi-fidelity regression with very few HF training points almost impossible. To address this problem, we developed a simplified multi-fidelity model, specifically designed to deal with HF data scarcity and high-dimensional input spaces.

Cokriging

The cokriging model was first introduced in [29], and successfully modified in a recursive manner in [30]. In its first formulation, $y_{HF}(\mathbf{x}) \sim \mathcal{GP}(0, k_{HF}(\mathbf{x}, \mathbf{x}'))$ and $y_{LF}(\mathbf{x}) \sim \mathcal{GP}(0, k_{LF}(\mathbf{x}, \mathbf{x}'))$ are supposed to be two GPs, with a zero mean prior. In particular

$$y_{HF}(\mathbf{x}) = \rho y_{LF}(\mathbf{x}) + \delta(\mathbf{x}) \,, \tag{3.55}$$

$$y_{LF}(\mathbf{x}) \perp \delta(\mathbf{x}) \,, \tag{3.56}$$

where $\rho \in \mathbb{R}$ here is the correlation hyperparameter, $\delta(\mathbf{x}) \sim \mathcal{GP}(0, k_\delta(\mathbf{x}, \mathbf{x}'))$ and $\perp$ indicates that two GPs are independent. Keeping in mind Equations 3.55 and 3.56, the HF kernel becomes

$$\begin{aligned} k_{HF}(\mathbf{x}, \mathbf{x}') &= \mathrm{Cov}\left[y_{HF}(\mathbf{x}), y_{HF}(\mathbf{x}')\right] && (3.57)\\ &= \mathrm{Cov}\left[\rho y_{LF}(\mathbf{x}) + \delta(\mathbf{x}), \rho y_{LF}(\mathbf{x}') + \delta(\mathbf{x}')\right] && (3.58)\\ &= \rho^2 k_{LF}(\mathbf{x}, \mathbf{x}') + k_\delta(\mathbf{x}, \mathbf{x}') \,. && (3.59) \end{aligned}$$

Given the HF input parameters X_{HF} and the LF ones X_{LF}, the Cokriging model assumes that $y_{HF}(X_{HF}), y_{LF}(X_{LF})$ are jointly Gaussian with the following prior distribution

$$\begin{bmatrix} y_{LF}(X_{LF}) \\ y_{HF}(X_{HF}) \end{bmatrix} \sim \left(\mathbf{0}, \begin{bmatrix} k_{LF}(X_{LF}, X_{LF}) & \rho k_{LF}(X_{LF}, X_{HF}) \\ \rho k_{LF}(X_{HF}, X_{LF}) & \rho^2 k_{LF}(X_{HF}, X_{HF}) + k_\delta(X_{HF}, X_{HF}) \end{bmatrix} \right) . \tag{3.60}$$

Then, the hyperparameters can be estimated as any other GP regression model solving an MLE problem, see Section 3.1.1, and the Cokriging can be utilized to predict the HF output in unseen input locations X^*.

Recalling Equation 3.55 and Equation 3.39, the Cokriging model is a corrective approach. In particular, this formulation allows us to learn only linear relationships between fidelities since ρ is a constant. More recent Cokriging formulations assume that $\rho = \rho(\mathbf{x})$ [30], with the function being a low-order polynomial. The most relevant difference with the first formulation of [29] remains the recursive approach that significantly reduces the computational cost when more than 2 fidelity levels are considered.

Multi-fidelity neural networks

There exist a few different approaches to multi-fidelity modeling with NNs, and usually the multi-fidelity model is composed of multiple NNs [33, 34, 35], trained simultaneously or in a segregated way. The general methodology consists of training a LF NN, or eventually any other regression model, in order to approximate $y_{LF}(\mathbf{x})$ and then train one more NN for each fidelity level. Once the LF approximation $\hat{y}_{LF}(\mathbf{x})$ is known, it is possible to use it to train a second NN, congruently to Section 3.1.2, which will predict the HF outputs with a nonlinear operator $\mathcal{F}$

$$\mathcal{F} : (\mathbf{x}, \hat{y}_{LF}(\mathbf{x})) \rightarrow y_{HF}(\mathbf{x}) \,, \tag{3.61}$$

representing the NN mapping from the augmented input parameter space to the HF output space.

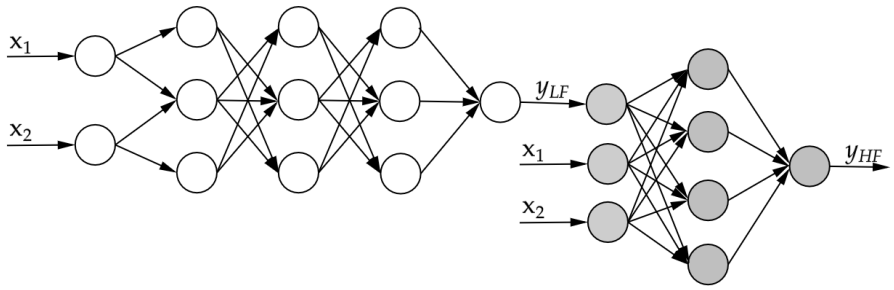
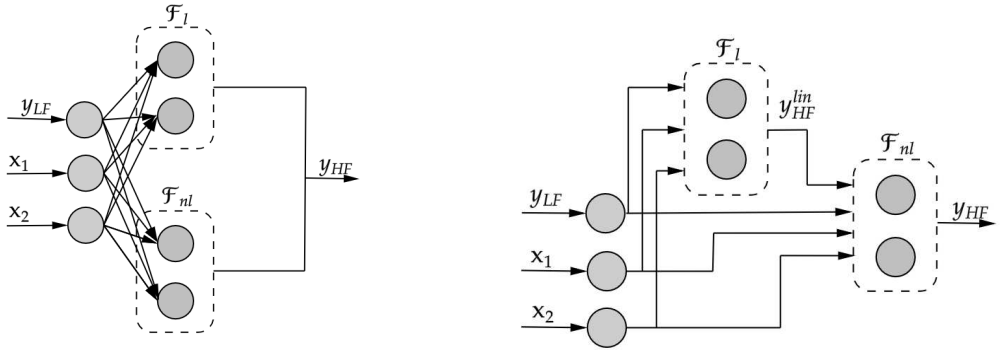


Figure 3.4: Multi-fidelity multi-level NN architecture example. The white NN represents the LF model, while the gray one represents the HF model.

Eventually, the HF NN operator $\mathcal{F}$ can be split into two components: a linear $\mathcal{F}_l$ and a nonlinear $\mathcal{F}_{nl}$ one [33], and then combined as in Equation 3.62. Moreover, a single loss function can aggregate all the different NN losses, allowing for training all the models together.

$$\mathcal{F} = \mathcal{F}_l \circ \mathcal{F}_{nl} \,. \tag{3.62}$$

The structure of the HF NN of Figure 3.4 becomes the one presented in Figure 3.5 (a) or (b), where the outputs of two NNs are combined to predict the HF function. In Figure 3.5 (a), the output of the 2 NNs is summed to obtain the HF output $y_{HF}(\mathbf{x}) \approx \mathcal{F}_l(\mathbf{x}, y_{LF}) + \mathcal{F}_{nl}(\mathbf{x}, y_{LF})$ [33]. On the other hand, in Figure 3.5 (b), the output $y_{HF}^{lin} = \mathcal{F}_l(\mathbf{x}, y_{LF})$ of the linear NN augments the input of the nonlinear NN, thus having $y_{HF}(\mathbf{x}) \approx \mathcal{F}_{nl}(\mathbf{x}, y_{LF}, y_{HF}^{lin})$ [34].



(a) Architecture according to [33]. (b) Architecture according to [34].

Figure 3.5: Linear and nonlinear splitting of the HF NN of a multi-level multi-fidelity NN, two different approaches.

In both cases, the linear NN $\mathcal{F}_l$ is characterized by neurons with no nonlinear activation functions, e.g., with a linear activation function (see Table 3.2), allowing to capture only linear correlations between fidelities. Instead, $\mathcal{F}_{nl}$ can have neurons with nonlinear activation functions, e.g. the hyperbolic tangent function. Since most of the time there is no information regarding the LF-HF correlation, this approach is rather flexible, allowing the handling of both linear and nonlinear correlations.

A simplified multi-fidelity model

The relationship between LF and HF data can be very close to a linear one in real-life applications. Hence, the popularity of Cokriging-like models, being them most

effective for simple linear LF-HF correlations. However, these models suffer from the curse of dimensionality when the input parameter space dimension is high, requiring more HF information. As a consequence, Cokriging models are not that straightforward to train with few HF data points for high-dimensional problems, due to the high number of hyperparameters to optimize.

In this thesis, we developed a simplified multi-fidelity model, tailored to simple LF-HF correlations problems [41]. This model aims to be less dependent on the dimensionality of the input parameter space and should be able to provide acceptable HF approximations of y_{HF} in extreme HF data scarcity conditions. The main hypothesis is that there is a sufficiently high linear correlation between y_{LF} and y_{HF}, which can be estimated with tools such as Pearson's R correlation coefficient. Under this hypothesis, it is possible to assume that

$$y_{HF} \approx c_1 \cdot y_{LF} + c_0 \,, \tag{3.63}$$

where $c_0, c_1 \in \mathbb{R}$ are the coefficients that represent the linear relationship between the outputs. Let us consider the training input set for the HF problem X_{HF} with cardinality n, and a LF set X_{LF} with an higher cardinality, and let us assume that the first n input parameters are the same, so that $\mathbf{x}^i_{HF} = \mathbf{x}^i_{LF}\ \forall i = 1, ..., n$. The corresponding outputs will be $y^1_{HF}, ..., y^n_{HF}$ and $y^1_{LF}, ..., y^n_{LF}$. If we want Equation 3.63 to be satisfied, the corresponding linear system of equations becomes

$$\begin{bmatrix} y^1_{LF} & 1 \\ \vdots & \vdots \\ y^n_{LF} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} y^1_{HF} & 1 \\ \vdots & \vdots \\ y^n_{HF} & 1 \end{bmatrix} \,, \tag{3.64}$$

which can be solved to find the optimal c_0, c_1 through least squares. It is possible to notice that the system does not depend on the input parameter $\mathbf{x}$ directly, and since the system is overdetermined when $n > 2$, we can compute c_0, c_1 with few HF data points, regardless of the input dimensionality. With c_0, c_1 we obtain a first linear approximation y^*_{HF} of the HF output

$$y^*_{HF}(\mathbf{x}) = c_1 \cdot y_{LF}(\mathbf{x}) + c_0 \,, \tag{3.65}$$

where $y_{LF}(\mathbf{x})$ can be substituted with a model $\hat{y}_{LF}(\mathbf{x}) \approx y_{LF}(\mathbf{x})$ trained on a large enough LF database if the LF problem cannot be solved in real time. In general, $y^*_{HF}(\mathbf{x})$ will not fit the HF training data, since c_0, c_1 were obtained with a least square solution of Equation 3.64. This behavior is not desirable, wasting crucial HF information; therefore, we compute the discrepancies δ^i between the first

approximation $y^*_{HF}(\mathbf{x})$ and the actual HF outputs corresponding to the training set X_{HF}, having

$$\delta^i = y^i_{HF} - y^*_{HF}(\mathbf{x}^i_{HF}) \quad \forall i = 1, ..., n\,. \tag{3.66}$$

The corresponding input parameter values can be mapped to these discrepancies δ^i and can be approximated with a regression model, such as an RBF or a GP, in every part of the input domain, obtaining the model $\hat{\delta}(\mathbf{x})$. This allows us to approximate the HF output function as

$$y_{HF}(\mathbf{x}) \approx y^*_{HF}(\mathbf{x}) + \hat{\delta}(\mathbf{x})\,. \tag{3.67}$$

However, since we are considering a situation of HF data scarcity, the model $\hat{\delta}(\mathbf{x})$ will still suffer from the lack of HF data points; hence, we do not expect $\hat{\delta}(\mathbf{x})$ to be accurate away from the training points. A possible solution is to introduce a weight function $w(\mathbf{x})$ which scales the contribution of $\hat{\delta}(\mathbf{x})$ in Equation 3.67 accordingly. There are different possible solutions for the shape of $w(\mathbf{x})$, introducing a certain bias from the user, and we decided to have the weights depending of the distance from the training locations X_{HF}, so that $w(\mathbf{x}) \to 0$ if $\mathbf{x}$ is too far from any training point. The model's prediction becomes

$$y_{HF}(\mathbf{x}) \approx y^*_{HF}(\mathbf{x}) + \hat{\delta}(\mathbf{x}) \cdot w(\mathbf{x})\,, \tag{3.68}$$

with $w(\mathbf{x})$ being in [41]

$$w(\mathbf{x}) = \frac{1}{1 + d_{\min}(\mathbf{x})}\,, \tag{3.69}$$

where $d_{\min}(\mathbf{x})$ is the minimum Euclidean distance of the point $\mathbf{x}$ from the HF training locations X_{HF} in the input parameter space. Since $d_{\min}(\mathbf{x})$ is inexpensive to compute, and $w(\mathbf{x})$ has an analytical formulation, the weights do not impact significantly the overall training time.

This model can be used with 3 or more HF training points, since the effect of the input dimensionality of $\mathbf{x}$ is limited to the discrepancy model $\hat{\delta}(\mathbf{x})$, but it is mitigated by the weight function $w(\mathbf{x})$. When $w(\mathbf{x}) \to 0$, the model assumes that the $\hat{\delta}(\mathbf{x})$ does not provide any more meaningful information and most of the prediction depends on $y^*_{HF}(\mathbf{x})$, which becomes the safest approximation in those zones of the domain where few HF points are available.

In Appendix A.1, an engineering test case is proposed, studying a multi-fidelity regression problem for the flame position inside a hydrogen burner under parametrized operative conditions. This work shows that even with few HF points, it is possible to outperform Cokriging-like methods if the correlation hypothesis is satisfied.

3.3 Data preprocessing

The models presented in the previous sections are meant to be used for POD coefficients regression in the context of parameterized NI ROMs. Thus, the parameterizations can be very different from each other, and the POD coefficients can have various orders of magnitude of difference, considering a different mode number. To control the range of variation of the input and output domain dimensions, in this thesis, we apply a normalization transformation in $[0, 1]$. This is done for each input dimension and for the output values too. Furthermore, regression models in general usually perform better with normalized data.

To do so, given a vector $\mathbf{v} \in \mathbb{R}^n$, e.g., one obtained from one of the input dimensions of the training database parameters, we can evaluate its minima and maxima $v_{\min} = \min \mathbf{v}$, $v_{\max} = \max \mathbf{v}$. Then the vector $\mathbf{v}$ can be normalized in $[0, 1]$

$$\tilde{v_i} = (v_i - v_{\min})(v_{\max} - v_{\min}) \quad \text{for } i = 1, ..., n\,, \tag{3.70}$$

obtaining the normalized vector $\tilde{\mathbf{v}}$. In the context of multi-fidelity data in this thesis, specifically for LF and HF outputs, the LF outputs are normalized with the same maximum and minimum values of the HF training outputs.

Since the regression model predictions will be coherent with the normalized outputs, they need to be de-normalized. The inverse transformation of Equation 3.70 becomes

$$v_i = \tilde{v_i}(v_{\max} - v_{\min}) + v_{\min}\,. \tag{3.71}$$

Chapter 4

Non-intrusive multi-fidelity reduced order model

The adoption of multi-fidelity approaches for surrogate models of parametrized fields is a fairly new trend. In [42], a multi-fidelity non-intrusive POD methodology is presented, appending the POD basis with LF modes projected on the complementary space spanned by the HF data available. In [43, 44], inconsistent HF and LF solutions are projected on a common latent space exploiting manifold alignment techniques. The multi-fidelity modeling is successively done with a hierarchical Kriging model. In [45, 46, 47, 48], multiple multi-fidelity Kriging meta-models are utilized in the POD coefficient approximation, with different representations of the variable fidelity POD coefficients. In [49], a non-intrusive reduced basis method for parametrized PDEs is introduced, and approximates the reduced basis coefficients with a multi-fidelity GP regression model. Artificial NNs map the relationship between variable fidelity POD coefficients in [50]. A comparison between Kriging and artificial NNs as surrogate models for the multi-fidelity approximation of the POD coefficients is presented in [51].

Furthermore, multi-fidelity deep-learning approaches have been developed. In [52], a multi-fidelity data fusion algorithm based on autoencoders is presented. In [53, 54], transfer learning enhances the multi-fidelity modeling, or variable fidelity information can be used for the pretraining of the deep learning model in the POD-based ROM, as in [55]. An intrusive Galerkin-POD surrogate model integrates multi-fidelity modeling in [56]. Here, the closure problem to compensate for the contribution of the truncated scales onto the resolved ones is interpreted as a multi-fidelity problem, using a multi-fidelity deep operator network (DeepONet) framework.

In general, there is not a fixed multi-fidelity characterization of the POD coef-

ficients in the POD-based ROMs, often involving multiple POD basis for different fidelity levels. In our original multi-fidelity ROM approach, we discuss the LF and HF representations of the POD coefficients. Moreover, we motivate the choice of the multi-fidelity regression model for the latent space interpolations, keeping in mind the industrial context and the limited knowledge of a priori correlation between fidelities.

In this Chapter, an original multi-fidelity extension of the NI ROMs of Chapter 2 will be presented, exploiting the NARGP multi-fidelity model illustrated in Section 3.2.1. In Section 4.1.1, the encoding strategy is discussed. Section 4.1.2 introduces the multi-fidelity regression of POD coefficients, where Section 4.1.2 focuses on the prediction and Section 4.1.2 motivates the choices in terms of multi-fidelity regression models.

4.1 Model structure

The proposed multi-fidelity ROM is a multi-fidelity extension of the NI POD-based ROMs of Chapter 2, presented in our previous work [57]. In this model, multi-fidelity methodologies have an effect on both the encoding, here done with the POD, and the regression models used to approximate the POD coefficients. Figure 4.1 illustrates the model structure, highlighting these specific behaviors. Here, on the top, both LF and HF snapshots feed the POD conceptual block, while, on the bottom, both the LF and HF POD coefficients become inputs for the multi-fidelity regression model conceptual block.

The first step of Figure 4.1 workflow is to have a Design of Experiments (DoE) to train the ROM. The kind of problems studied with ROMs are parametric; therefore, each snapshot is paired with an input parameter, representing the design configuration. DoE techniques consist of choosing the best design configurations to optimally explore the input parameter space when the quantity of interest function is unknown. Usually, DoE methodologies rely on spatial, random, or quasi-random sampling strategies and are often utilized together with regression models. Some of the most common techniques are the Latin Hypercube Sampling (LHS), uniform random sampling, Halton sequences, and factorial methods, such as full and reduced factorial. Being NI ROMs, and multi-fidelity NI-ROMs by extension, based upon regression models, a well-thought-out DoE can positively affect the ROM performance. From a multi-fidelity perspective, both LF and HF DoEs need to be considered. If the LF snapshots' computational cost is negligible, the DoE has less of an impact,

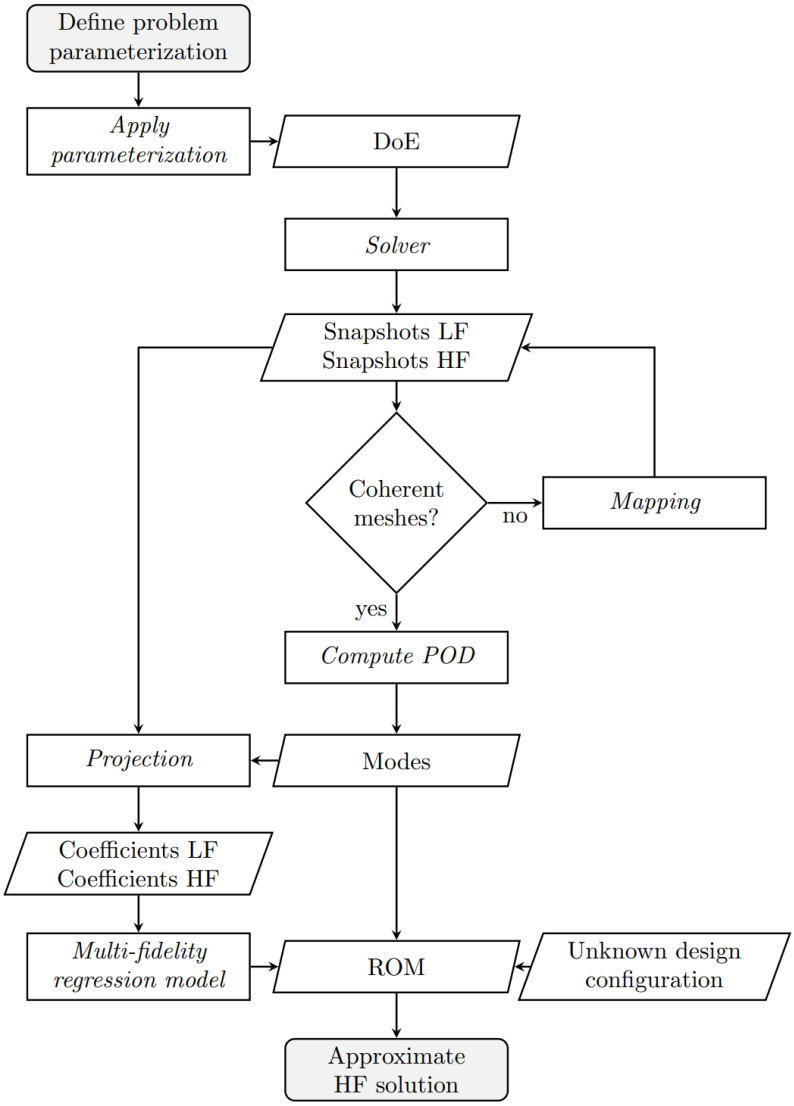


Figure 4.1: Flowchart representing the structure of the proposed POD multi-fidelity ROM.

while if its cost becomes significant, the LF sampling should cover well and densely the input parameter space. On the other hand, special care should be considered for the HF snapshots. In general, it is better to have nested HF and LF snapshots, so that the HF training input parameters are a subset of the LF ones. This is preferable

since nested LF-HF data ease learning the relationship between low- and high-fidelity information when multi-fidelity regression models, as those in Section 3.2, are used in the proposed multi-fidelity ROM. Nevertheless, the multi-fidelity ROM will still be trainable when LF and HF designs are not nested, especially if there are enough LF snapshots to obtain a good LF surrogate to replace the LF data, performing similarly to the nested case.

Subsequently, once both LF and HF solutions have been computed according to the respective DoE, we have to ensure that all snapshots are coherent. This is especially important for multi-fidelity ROMs since the LF snapshots are often obtained with coarser discretizations than HF ones. Section 2.3.3 discusses this topic in detail, and all the considerations relative to geometrical parameterization remain valid for the multi-fidelity ROM too. For all further discussions, we assume that all snapshots are defined on the same reference discretizations.

4.1.1 Mixing heterogeneous snapshots

When dealing with multiple fidelity levels, there are different approaches to the POD encoding of the snapshots, and the three most intuitive ones are:

- POD only of LF snapshots;
- POD only of HF snapshots;
- POD of both LF and HF snapshots.

In this work, we focus on the latter, where the POD is computed mixing both LF and HF snapshots.

From a conceptual point of view, the LF-only POD utilizes a great number of LF snapshots obtained from all parts of the input domain. If the LF snapshots are informative enough, the LF basis should be meaningful also for the HF problem and encode well unseen HF snapshots, even in remote parts of the input parameter domain. However, LF solvers often rely on some simplifications, whether it is in the mathematical modeling or in the discretization, and the LF POD basis might not be able to describe certain behaviors. As an example, if we consider a CFD problem where the LF solver uses a coarse mesh and the HF solver uses a finer one, all the smaller-scale dynamics can not be represented at all by the LF snapshots; hence, it is impossible to have LF POD modes associated with those behaviors. Moreover, LF information might be noisier than HF information, and the POD will try to encode

noise that might later pollute the ROM.

On the other hand, an HF-only POD should find the best possible basis that represents the HF training snapshots, with minimal noise impact and with few POD modes. However, since multi-fidelity problems are characterized by HF data scarcity, the resulting HF-only POD basis is likely to poorly represent out-of-sample snapshots. This can be easily observed by evaluating the projection error, as in Equation 2.6, on a validation set of snapshots: if the error is close to 0, the basis can represent unseen snapshots well; otherwise, it needs more training snapshots to achieve lower errors. Remembering the reconstruction error e_{rec} components, as in Equation 2.18, if, hypothetically, the interpolation error e_{int}, due to the POD coefficient regression models, goes to zero

$$e_{rec} = e_{prj} + \cancelto{0}{e_{int}} = e_{prj} \,, \tag{4.1}$$

then the projection error e_{prj} becomes the reconstruction error lower bound. As a consequence, lowering the out-of-sample projection error is always beneficial to the ROM.

Finally, if LF and HF snapshots are mixed at the POD level, it is possible to mitigate the disadvantages of both the previous approaches while keeping most of the beneficial aspects. Indeed, since the mixed fidelity POD modes are a basis for both LF and HF snapshots, it means that we are still retrieving the precious HF information otherwise lost by the LF-only POD basis, and at the same time, the representability of the basis of unseen HF snapshots is higher than the HF-only POD basis. In general, the effect of the noise that might have been introduced with the LF snapshots at the encoding stage is mitigated by the POD energy threshold. The main drawback of this approach is that the number of POD modes is the highest of the three possibilities: the higher the cardinality of the snapshot dataset, the higher the number of POD basis functions for the same energy threshold. Thus, the ROM will require more memory, it will occupy more storage space on the machine, and it will need more POD regression models to be trained, increasing the overall computational cost. On the other hand, if this higher ROM computational cost is negligible compared to more HF simulations, it does not represent a problem.

From a practical standpoint, we observed in our applications that the projection error improves sensitively when utilizing the mixed fidelity POD. Moreover, we observed that this improvement translates positively onto the reconstruction error too, if compared to the HF-only POD. These topics are discussed in further detail

in Chapter 5, where the multi-fidelity ROM results are presented.

4.1.2 Multi-fidelity POD coefficient approximation

Once the POD has been computed and the POD modes $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_r\}$ are available, it is possible to evaluate the POD coefficients corresponding to the training designs. Since there are both LF and HF snapshots, we will obtain both a LF representation of the POD coefficients a^{LF} and an HF representation a^{HF}. According to the input notation of Section 3.2.1, let $X_{LF} = \{\mathbf{x}_i : i = 1, ..., n_{LF}\}, X_{HF} = \{\mathbf{x}_i : i = 1, ..., n_{HF}\} \subset X_{LF}$ be the LF input set and the HF input set, respectively. Therefore, if we consider the generic input $\mathbf{x}_i$, the j-th mode POD coefficient representations are obtained by projection

$$a_j^{HF}(\mathbf{x}_i) = \mathbf{u}^{HF}(\mathbf{x}_i) \cdot \boldsymbol{\psi}_j \,, \tag{4.2}$$

$$a_j^{LF}(\mathbf{x}_i) = \mathbf{u}^{LF}(\mathbf{x}_i) \cdot \boldsymbol{\psi}_j \,, \tag{4.3}$$

for every POD mode $\boldsymbol{\psi}_j$ with $j = 1, ..., r$.

Depending on which snapshots have been used when computing the POD basis $\boldsymbol{\Psi}$, the LF and HF representation will change accordingly. Thus, the choice of the training snapshots utilized for the encoding has not only an effect on the representability of unseen HF snapshots, but changes the POD coefficients themselves, affecting the regression models too. As an example, the HF coefficients $a_1^{HF}(\mathbf{x}_i)$ will be different for the first mode whether we evaluate the POD from LF snapshots only, HF snapshots only, or a mix of both, and this will be true for all other modes. This is due to the modes obtained from different snapshot matrices being different, even for the first ones, which generally are associated with the most energetic features. As stated in Section 4.1.1, we focus on the mixed fidelity snapshots POD approach, meaning that the POD basis is a system of generators for both LF and HF training snapshots, instead of just one of the fidelity levels.

For each j-th mode in $1, ..., r$, we train a single-fidelity regression model, $\hat{a}^{LF}(\mathbf{x}) \approx a^{LF}(\mathbf{x})$, with the LF training database $D_j^{LF} = \{(\mathbf{x}_i, a_j^{LF}(\mathbf{x}_i)) : i = 1, ..., n_{LF}\}$. In principle, any single-fidelity regression model presented in Section 3.1 can be used to approximate $a^{LF}(\mathbf{x})$, but we will consider the GP regression model since it will enable the NARGP modeling for the multi-fidelity approximation of the HF POD coefficients. Other models, such as RBF, can be a good alternative, especially if the cardinality of D_j^{LF} increases, making the GP regression intractable.

Similarly, for each j-th mode in $1, ..., r$, we can train a multi-fidelity approximation $\hat{a}^{HF}(\mathbf{x}) \approx a^{HF}(\mathbf{x})$ of the HF POD coefficient with the NARGP model. To do so, we need to create a map between the input parameters $\mathbf{x}$, augmented by the corresponding LF POD coefficient approximations $\hat{a}^{LF}(\mathbf{x})$, previously trained, and the HF POD coefficient a^{HF}. The training database for the NARGP approximation of the j-th HF POD coefficients becomes $D_j^{HF} = \{((\mathbf{x}_i, \hat{a}_j^{LF}(\mathbf{x}_i)), a_j^{HF}(\mathbf{x}_i)) : i = 1, ..., n_{HF}\}$. Defined the multi-fidelity problem, each HF POD coefficient is modeled according to Section 3.2.1, which describes in more detail the NARGP model.

Prediction

Supposing that all low- and high-fidelity POD coefficients models have been trained, the multi-fidelity ROM prediction becomes

$$\mathbf{u}^{HF}(\mathbf{x}) \approx \sum_{j=1}^{r} \hat{a}_j^{HF}(\mathbf{x}, \hat{a}^{LF}(\mathbf{x}))\boldsymbol{\psi}_j \,. \tag{4.4}$$

Another possibility to take into consideration is to limit the utilization of the multi-fidelity model, i.e., the NARGP, only to specific mode numbers and resort to faster single-fidelity approximations, such as GP regression or RBF interpolation, for the remaining POD coefficients. This only helps with the training time and generally does not improve the accuracy. In this work, we have identified two reasonable solutions, namely approximating with a multi-fidelity model only:

1. the first POD coefficients;
2. the POD coefficients that shows a Pearson's R correlation coefficient, computed with nested LF and HF POD coefficients, above a user-defined threshold.

In the first case the rationale is to exploit the accuracy gain only for the most energetic modes, which contribute the most to the overall model representability of the problem, thus its accuracy. Indeed, most of the time LF solvers are able to capture only macroscopic behaviors of the problems, e.g., due to a coarser discretization for the numerical solution. These behaviors carry most of the problem energy and, consequently, they are often captured by the first modes when we perform the POD.

On the other hand, the second approach focuses the computational resources allocated for the ROM training to the most likely POD modes which could benefit from a multi-fidelity model. Even though Pearson's R only highlights linear correlations between fidelities, it is a cheap yet effective indicator with an appropriate threshold. Naturally, a linear correlation measure can not be trusted completely for

non-linear relationships, even with a loose threshold, so it is a sensitive approach only if the enough POD coefficients show high enough Pearson's R coefficients between fidelities.

Either choices will identify two sets of indices I_S and I_M, corresponding to the POD mode numbers, which will refer to the POD coefficients to be modeled with a multi-fidelity or a single-fidelity model respectively. Therefore, the multi-fidelity ROM prediction of Equation 4.4 becomes

$$\mathbf{u}^{HF}(\mathbf{x}) \approx \sum_{j \in I_M} \hat{a}_j^{HF}(\mathbf{x}, \hat{a}^{LF}(\mathbf{x}))\boldsymbol{\psi}_j + \sum_{j \in I_S} \hat{a}_j(\mathbf{x})\boldsymbol{\psi}_j \,, \tag{4.5}$$

where here $\hat{a}_j^{HF}$ is again the multi-fidelity model of the j-th POD coefficient, while $\hat{a}_j$ is the single-fidelity approximation of the HF POD coefficient.

Choice of the multi-fidelity regression model

It appears evident that almost any multi-fidelity model can be used to approximate $\hat{a}_j^{HF}$. However, NARGP is particularly suitable for this problem, especially for industrial use cases. Indeed, NARGP can handle a wide variety of LF-HF relationships due to its intrinsic nonlinear nature, while Cokriging-like models are limited to linear or polynomial correlations. When dealing with real-life problems, in general, there is no prior knowledge regarding the type of correlation between the LF and HF POD coefficients associated with each of the different POD modes. As a consequence, a nonlinear model able to capture both linear and nonlinear multi-fidelity relationships can be more effective and flexible, with minimal user intervention. Another NARGP advantage is the capability to act as a single-fidelity model when the MLE hyperparameter optimization does not reward the multi-fidelity component of the NARGP's HF composite kernel formulation, as Section 3.2.1 illustrates. This might happen when the LF and HF representations of a POD coefficient are not correlated, and a multi-fidelity model might not improve the accuracy of the regression. If the HF NARGP kernel can ignore all the LF information contribution, the model safely approximate the HF POD coefficients with a GP regression in a completely autonomous way, again improving the multi-fidelity ROM ease of use. In general, NARGP training is less resource-intensive than an equivalent multi-fidelity NN, making it preferable when dealing with an high number of POD modes, requiring training a multi-fidelity regression model for each one of them. This happens frequently due to the addition of many LF snapshots to the snapshot matrix in the mixed fidelity POD, as the results presented in Chapter 5 show. The main drawback of NARGP as the POD coefficient multi-fidelity model, if compared to other GP-based models,

is its complexity, which increases the number of hyperparameters, thus potentially hindering the training process and negatively affecting the training time. Nonetheless, its flexibility generally overcomes these limitations, which often does not pose a real problem.

In summary, the proposed multi-fidelity ROM starts by mixing the different fidelity snapshots in a unique matrix of snapshots, encodes the solution with POD, and approximates the latent snapshot representation with a nonlinear multi-fidelity model. In particular, NARGP has been identified as the optimal candidate for the multi-fidelity regression task, especially in terms of automation and flexibility. The main improvements expected to be achieved, compared to equivalent single-fidelity ROMs, are primarily related to the representability of the multi-fidelity ROM and its accuracy.

Chapter 5

Application to industrial problems

Different industrial problems might benefit from ROMs, such as design, optimization, uncertainty quantification, multi-physics simulations or control. All these are many-query applications, where numerous and expensive numerical simulations need to be performed. ROMs have low online computational cost, with almost real time performance. However, offline computational cost is heavily affected by the training database generation. Multi-fidelity ROMs aim to tackle this problem, showing that inaccurate but cheap LF information can be exploited in a non-intrusive ROM framework. Moreover, in an industrial context, geometrical parameterization plays a key role in the design process, and it requires particular care when utilizing ROMs.

This Chapter is structured as follows. The first three sections report the core results for this thesis where the multi-fidelity ROM is applied to relevant industrial problems. In detail, Section 5.1 presents an external aerodynamic automotive problem with an high-dimensional geometrical parameterization, and Section 5.1.2 deeply discusses the ad-hoc geometrical mapping strategy. Section 5.2 focuses on the internal flow and thermal characterization inside a pipe with wavy surfaces, having both geometrical and operational parameters. In Section 5.3 there is an uncertainty quantification study on the performance of a vertical axis wind turbine under uncertain rotational speed. The details of the uncertainty quantification techniques are in Section 5.3.2. Finally, Section 5.4 presents an additional application of the proposed methodology. While the outcomes are not as conclusive as in the previous three core case studies, this test remains valuable as it illustrates the versatility of the approach, highlights its current limitations, and provides further context to the overall contribution of this thesis.

5.1 DrivAer

External aerodynamic problems, i.e., automotive ones, are mainly interested in the aerodynamic forces acting on the body's external surfaces. One of the main contributors to the aerodynamic forces is the wall's pressure field, making it critical in the design process. This application aims to compare the capabilities of a multi-fidelity ROM with those of an equivalent single-fidelity ROM. The ROMs approximate the pressure field acting on a car boundary, given a geometrical parameterization.

This section builds upon our earlier study, published in [57]. While the main findings remain unchanged, certain adjustments have been made to fit the context and objectives of the present thesis.

Figure 5.1: DrivAer in fastback configuration: on top - side view; on bottom from left to right - front and back views.

5.1.1 Problem Description

We considered the DrivAer test case geometry in the fastback configuration [58, 59] as our baseline geometry, which can be seen in Figure 5.1. The baseline geometry is then deformed according to a geometrical parameterization, and the new geometry is used to perform a CFD simulation. From the CFD results, the pressure field acting

on the car body is kept to successively train the ROMs. An example of the boundary pressure field can be found in Figure 5.2.

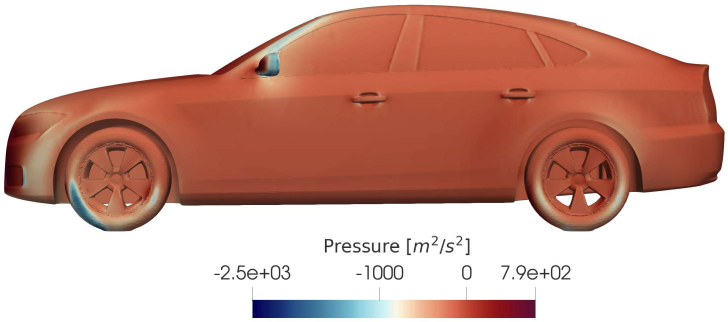


Figure 5.2: High-fidelity pressure field on the car body - side view.

Parameterization

All the problem parameters are geometrical, keeping constant all other operational and physical set-ups. In particular, there are a total of 6 parameters, each corresponding to a different geometrical deformation of the baseline fastback car. These deformations have been obtained by manipulating the car's geometry through a free-form deformation-based technique [60], with the software `mimic` [61]. Indeed, the software is able to move the surface features and propagate the deformation field smoothly around the features themselves, while preserving imposed geometrical constraints. To do so, a Laplacian equation of the deformation is solved, keeping unaltered the deformed designs' manufacturability. This allows for parameterizing the geometry without the utilization of a Computer-Aided Design (CAD) software. Each deformation is then associated with a separate scalar value. Figure 5.3 illustrates the effect of each of the 6 geometrical deformations, affecting the vehicle's:

1. front window angle;
2. rear window angle;
3. roof drop;
4. greenhouse angle;
5. bumper nose extrusion;
6. bumper nose drop.

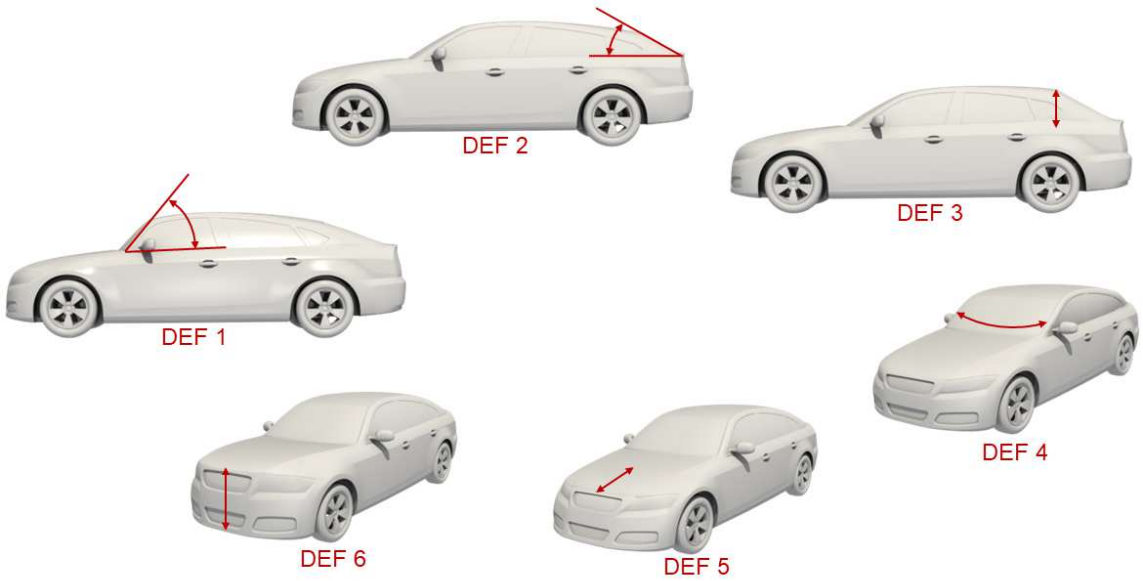


Figure 5.3: Representation of car deformations, in clock-wise order from the left: *DEF 1* Front window angle; *DEF 2* Rear window angle, *DEF 3* Roof drop; *DEF 4* Greenhouse angle; *DEF 5* Bumper nose extrusion; *DEF 6* Bumper nose drop.

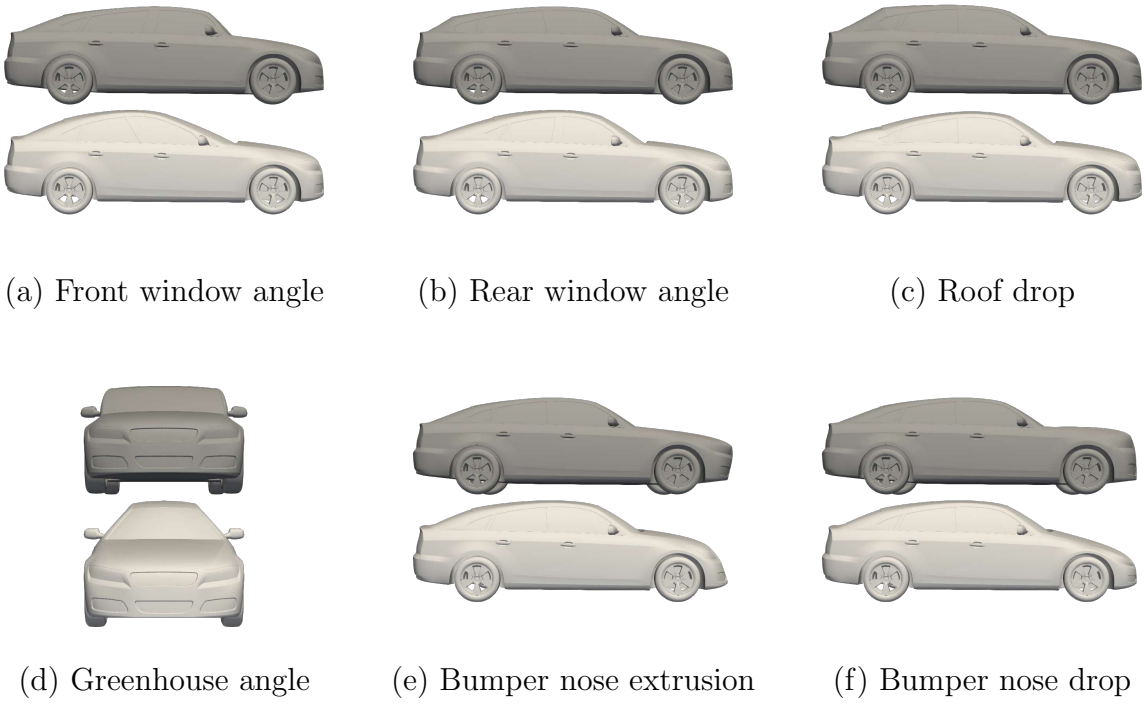
The magnitude of each deformation can be appreciated in Figure 5.4, where the minimum and maximum deformations are presented.

Simulation set-up

To obtain the pressure field, a CFD simulation of the flow around the vehicle is mandatory. The problem was simplified by exploiting the car's symmetry by its length; thus, we considered only half of the car. Around the halved car, we defined a box domain of dimension $10.0L \times 2.2L \times 2.5L$, with L being the car length.

Since the aim of this experiment is to compare multi-fidelity and single-fidelity ROMs, two types of solvers were considered: a LF solver and a HF solver. From a practical standpoint, the main difference between the two lies in the mesh discretization. The LF computational grid is a hex-dominant mesh with ≈ 1.8 million cells, being coarser than the finer HF mesh. The HF mesh has ≈ 6.0 million cells, with most of the refinements being near the car surface. Figure 5.5 illustrates a longitudinal slice of a LF mesh, showing two of the three main dimensions of the computational domain and the grid refinement locations.

All the remaining simulation parameters and choices are shared between both LF and HF solvers.



(a) Front window angle (b) Rear window angle (c) Roof drop

(d) Greenhouse angle (e) Bumper nose extrusion (f) Bumper nose drop

Figure 5.4: Extreme deformations for all six deformation parameters applied separately to the baseline geometry.

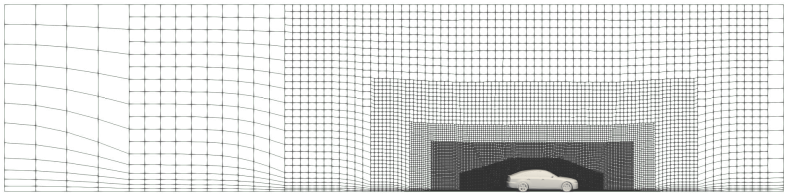


Figure 5.5: LF mesh slice and car geometry, side view.

The incompressible steady-state RANS equations are solved together with the Spalart-Allmaras turbulence model [60, 62, 63]. This turbulence model is routinely employed in similar industrial designs thanks to its simplicity and ease of use. However, some of the considered vehicle designs reflect the behavior typical of a bluff body; therefore, the Spalart-Allmaras model is not always reliable enough to adequately capture the flow sensitivity due to some geometrical changes. In the context of preliminary design choices, exploring many different geometrical configurations is usually more important than a high accuracy of the CFD simulations, which is inevitably expensive to achieve. We decided to accept this accuracy-cost compromise, valuing the capability to discern the general performances of multiple designs.
The solver `simpleFoam`, embedded in the open-source CFD software `OpenFoam` [64], has been employed to solve the set of governing equations. The far field air velocity was set to $U_0 = 38.89$ m/s, leading to a Reynolds number of $\approx 12 \cdot 10^6$, based on the length of the car. The ground is modeled as a moving wall to match the tangential velocity of the wheels, which, in their turn, are modeled with a moving reference frame.

Concerning the differences between LF and HF pressure solutions on the vehicle boundary, Figure 5.6 shows the actual differences between two pressure snapshots, and Figure 5.16 compares LF and HF drag coefficients C_D, defined in Equation 5.7, for different designs. Indeed, C_D depends on the pressure field integral on the car boundary.

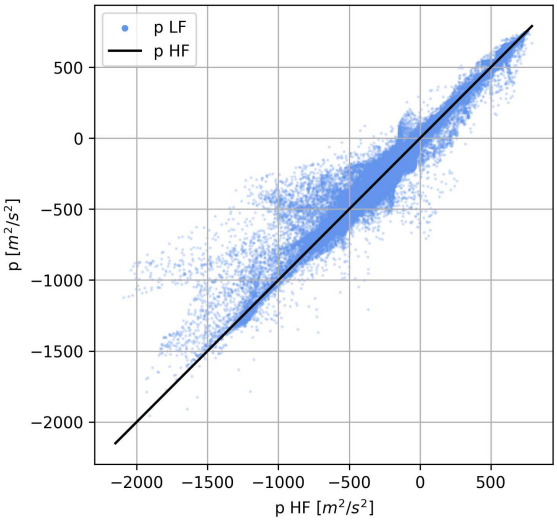


Figure 5.6: Scatter plot between HF and LF pressure solutions associated to the same design; each point correspond to the same location on the vehicle's surface.

Design of experiments

The training DoE is based upon a LHS where the input parameter values correspond to the 6 geometrical deformations, and are normalized in $[0, 1]$. The LF training DoE was obtained a priori with the LHS strategy, consisting of 160 HF designs. Part of the LF designs were evaluated with the HF solver to create the HF training DoE. Since the results will be evaluated for different HF dataset dimensions, the original LHS designs were opportunely ordered. In particular, starting from a random design, the next design is added recursively so that for any design, the last one is at maximum distance from the previous ones. This is achieved by evaluating the minimal Euclidean distance in $\mathbb{R}^6$ from all previous designs. As a consequence, in the result Section, the first 20 HF training snapshots will be a subset of the first 40 HF snapshots, and so on. The HF snapshot training set dimensions will range from 20 HF snapshots to 120 HF snapshots. Thus, the first 120 snapshots of the 160 LF training snapshots will correspond to the 120 HF snapshots in the HF training set. The HF validation DoE was sampled from a random uniform distribution, counting 70 HF snapshots.

The LHS choice for the training DoE represents a good balance between an even coverage of the input parameter domain, while not needing too many designs [65]. This is made possible by the pseudo-random nature of the LHS strategy. Other techniques, such as full and reduced factorials, would not be a feasible choice for high-dimensional input parameter spaces, as in this problem.

ROM setup

In this experiment, we compared two ROMs: a single-fidelity ROM and a multi-fidelity ROM. The first ROM has been trained with HF data only, while the multi-fidelity ROM will use the same HF information, with additional LF data. To make all ROMs comparable, the POD energy ε, as in Equation 2.3, is set to $\varepsilon = 0.999$.

For each multi-fidelity ROM, we utilized multi-fidelity regression models only on the first 4 modes' POD coefficients. As stated in Section 4.1.2, the first POD modes capture most of the meaningful information; therefore, improving the regression model accuracy of the first POD coefficient modes with a multi-fidelity regression has the greatest impact. In this work, we used the NARGP model to perform the multi-fidelity regressions of the POD coefficients. Another reason to focus on the first POD modes for the multi-fidelity modeling is that we are likely to find better LF-HF correlations for the first POD coefficients, improving the effectiveness of the

multi-fidelity model. This stems from two factors: firstly, the LF solver struggles to capture high-frequency features, due to the coarser computational grid with respect to the HF case; secondly, the last and less energetic POD modes are often associated with these high-frequency behaviors. Consequently, the projection of LF snapshots on the last part of the POD basis will not necessarily be helpful for the multi-fidelity modeling. On the other hand, each of the less energetic POD modes' coefficients has been approximated with a single-fidelity GP regression, following Equation 4.5 for the multi-fidelity ROM prediction. In general, using the multi-fidelity approach for more POD coefficients should not hinder the ROM capabilities, especially with the NARGP model, but it can increase the ROM training time.

Concerning the hyperparameter settings of the GP-based regression models, we chose to use Matérn 3/2 kernels with ARD and a 0 mean prior, and we constrained only the lengthscale hyperparameters. The details regarding the hyperparameter constraints and other GP options are presented in Table 5.1.

Table 5.1: GP-based regression models configurations for the ROMs.

	NARGP		
	LF model	HF model	Single-fidelity GP
Optimization algorithm	BFGS*	BFGS	BFGS
Max iterations	2500	2500	2500
Optimization restarts**	10	8	10
Kernel	Matérn 3/2	k_ρ : Matérn 3/2 k_f : Matérn 3/2 k_δ : Matérn 3/2	Matérn 3/2
Lenghtscale bounds***	$0.5 \leq l_i/d_i \leq 30$	k_ρ : $0.5 \leq l_i/d_i \leq 10$ k_f : free k_δ : $0.5 \leq l_i/d_i \leq 10$	$0.5 \leq l_i/d_i \leq 10$
Gaussian noise variance	free	free	free

* Broyden–Fletcher–Goldfarb–Shanno algorithm.

** Random restarts of the optimization to increase robustness.

*** The i-th lengthscale is called l_i and the i-th input variable range of variation's amplitude called d_i; different lengthscales are applied only to anisotropic kernels.

The single-fidelity GP column in Table 5.1 refers to the GP training configurations used for the less energetic POD mode coefficient approximations. Nonetheless, the same setups have been employed for the single-fidelity GP regression models of the single-fidelity ROM used for comparison.

5.1.2 Mapping

Algorithm 1 Overview of the full mapping workflow.

1: Create a database of deformed car geometries (`mimic`)
2: Solve the CFD problem for the deformed cars (`OpenFOAM`)
3: Define a reference surface mesh from the undeformed vehicle
4: Deform the reference surface mesh
5: Interpolate the results from the CFD solution to the deformed reference mesh
6: Map the solution on the deformed reference mesh back to the undeformed reference mesh

The adoption of a mapping strategy is pursued to have all snapshots with a coherent discretization, as discussed in Section 2.3.3. To perform the POD, the cardinality of each snapshot must be identical. In addition, the indexing must be done so that the connectivity is preserved between different snapshots. Here we will discuss the mapping approach employed in this experiment, and an overview of the process can be found in Algorithm 1. The mapping is necessary since all of the considered designs will undergo a specific CFD solution, having all the computational meshes different from each other. This has been necessary due to the great deformations involved, making it impossible to guarantee an adequate mesh quality if deforming the computational grid prior to the CFD solution.

Keeping in mind that we are interested in only the car surface mesh and the relative pressure distribution, the mapping consists of three main steps:

1. the generation of a common reference mesh;

2. the morphing of an undeformed triangulated car geometry, according to the geometrical parameterization;

3. the interpolation of the solution field onto the reference nodes.

For the reference mesh, we computed a fine computational mesh for the baseline car, whose external surface we will call $\mathcal{S}_{car} \subset \mathbb{R}^3$. Let us call Φ the transformation responsible for the geometrical deformation of the car, and suppose it to be known. $\Phi(x_P; \mathbf{x})$ transforms a point $P \in \mathcal{S}_{car}$ of coordinates $x_P \in \mathbb{R}^3$ into $P' = \Phi(P)$, of coordinates $x_{P'} \in \mathbb{R}^3$, according to the known geometrical parameters $\mathbf{x} \in \mathbb{R}^6$. If we apply the transformation to each point belonging to the baseline car, we obtain the deformed car $\mathcal{S}'_{car}$ for the given parameter vector $\mathbf{x}$, as in

$$\mathcal{S}'_{car} = \{P' : x_{P'} = \Phi(x_P; \mathbf{x}), \quad \forall P \in \mathcal{S}_{car}\} \ . \tag{5.1}$$

In this experiment, Φ is known and it is a byproduct of the free-form deformation of the baseline car; however, the suggested mapping approach remains valid for any known geometrical transformation Φ, regardless of how it originated, and can be extended to similar problems. Figure 5.7 offers a schematic representation of the effect of a generic transformation Φ when applied to a mesh.

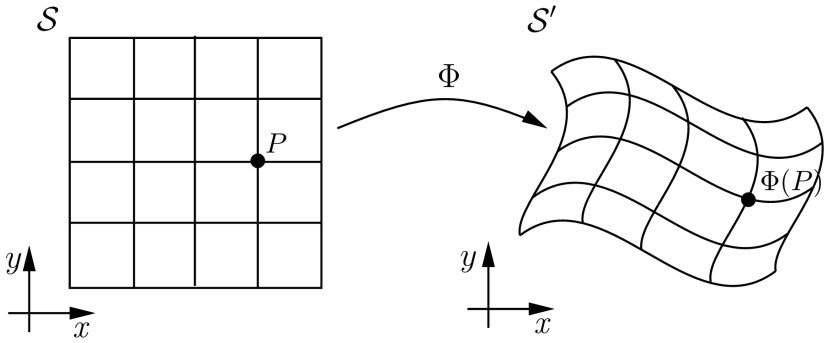


Figure 5.7: Schematic representation of the geometrical deformation's effect on a mesh. Φ is the transformation function, $\mathcal{S}$ is the original grid, P is a point of $\mathcal{S}$, and $\mathcal{S}', P$ are their respective transformations.

For a given parameter $\mathbf{x}$, the points belonging to the reference car $\mathcal{S}_{car}$ tesselation are projected onto the deformed car $\mathcal{S}'_{car}$, of which the CFD solution is known on the computational mesh. Consequently, both morphed reference points and the CFD results are living on the same locus of points, $\mathcal{S}'_{car}$, similarly to what Figure 5.8 shows. Each morphed reference point will now belong to a single face of the deformed car's solution tessellation. Thus, the point's barycentric coordinates with respect to this face's vertices can be evaluated as

$$\left[w_P^{V_i}\right]_{i=1,\dots,N_V} = \left[\frac{1}{\|x_{V_i} - x_{\Phi(P)}\|_2}\right]_{i=1,\dots,N_V} , \tag{5.2}$$

with V_i being the i-th vertex of the face, N_V the number of vertices of the face, x being the spatial coordinates of a point, and the $\left[w_P^{V_i}\right]_i$ are the barycentric coordinates of P. With the barycentric coordinates of P with respect to the face's vertices, it is possible to accurately interpolate the deformed car target fields values, i.e., the boundary pressure field, onto the morphed reference mesh following Equation 5.3. Since the morphed reference tesselation preserved the indexing and the connectivity of the original reference mesh, the interpolated field on a morphed point can be assigned to the corresponding original reference mesh point.

$$p^{ref}(x_P; \mathbf{x}) = \sum_{i=1}^{N_V} w_P^{V_i} \cdot p^{def}(x_{V_i}; \mathbf{x}) \,, \tag{5.3}$$

where p^{ref}, p^{def} are respectively the target fields, i.e., pressure, on the reference mesh and the deformed mesh obtained from the CFD calculations. By doing so, homogeneous snapshots of the target fields can be collected, even if the number of mesh cells differs or the grid connectivity changes. Figure 5.8 illustrates the mapping of a node from the reference mesh to the deformed geometry, with different discretizations.

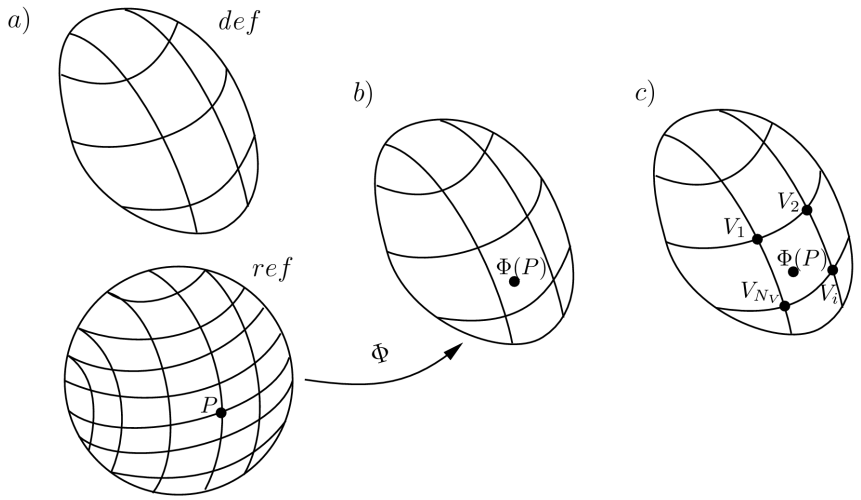


Figure 5.8: Schematic representation of the mapping of a reference (*ref*) mesh node to the deformed (*def*) geometry. (*a*) the *ref* and *def* geometries with meshes; (*b*) the transformation of the point P on the *ref* mesh to the *def* geometry; (*c*) vertices of the face to which $\Phi(P)$ belongs to the *def* geometry's mesh.

Both the barycentric interpolation and the geometry deformation techniques are

well known and have been used in many applications. In particular, the geometry deformation is based on free form deformations, which, in the context of ROMs, are used to move the computational mesh nodes according to a deformation field applied to the geometry itself [60]. However, to our knowledge, using these techniques to map surface fields from non-coherent boundary tessellations is novel. Therefore, this approach allows the use of computational meshes that were generated separately for each design configuration. This enables POD-based workflows to handle larger geometric deformations, overcoming the intrinsic limitations posed by the mesh elements' ordering and cardinality.

5.1.3 Results

As Section 5.1.1 introduced, the GP-based single-fidelity ROM and the multi-fidelity ROM have been trained for different dataset sizes, specifically with 20, 40, 60, 80, 100, and 120 HF snapshots. The multi-fidelity ROM training set is always enhanced with 160 LF snapshots. Both ROMs are tested on 70 HF snapshots reserved for validation.

The first metric we consider is the projection error, defined in Equation 2.6. In particular, we consider the mean projection error $\overline{e_{prj}}$ over the validation set, having

$$\overline{e_{prj}} = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \frac{\left\| \sum_{j=1}^{r} \left((\mathbf{u}_i \cdot \boldsymbol{\psi}_j) \boldsymbol{\psi}_j \right) - \mathbf{u}_i \right\|_2}{\|\mathbf{u}_i\|_2}, \tag{5.4}$$

where n_{val} is the number of validation snapshots, $\mathbf{u}$ represents a snapshot, $\boldsymbol{\psi}$ is a POD mode and r is the ROM's number of POD modes. For our purposes, $\mathbf{u}$ is the pressure distribution mapped on the reference car. This error quantifies the capability of the POD reduced basis to represent an out-of-sample snapshots, i.e., the ones reserved for validation. The projection error is independent of the POD coefficients regression models and measures only the encoding capabilities of the decomposition operator. Figures 5.9 and 5.10 show the projection error trends for different amounts of HF training snapshots.

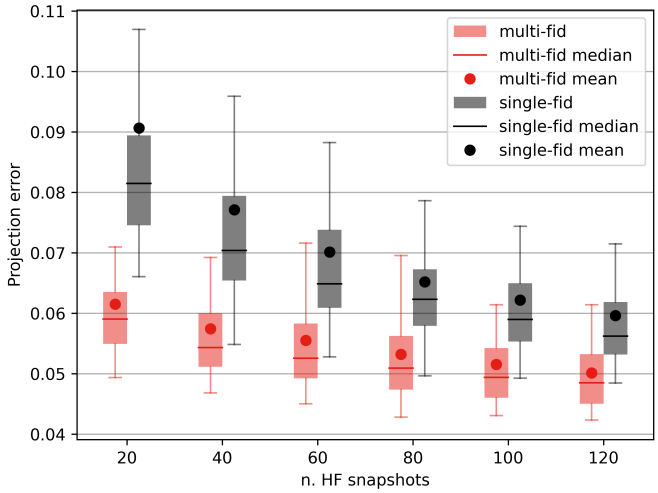


Figure 5.9: Projection error boxplots comparison between single and multi-fidelity PODs for different numbers of HF snapshots. Statistics on 70 HF validation snapshots - outliers omitted.

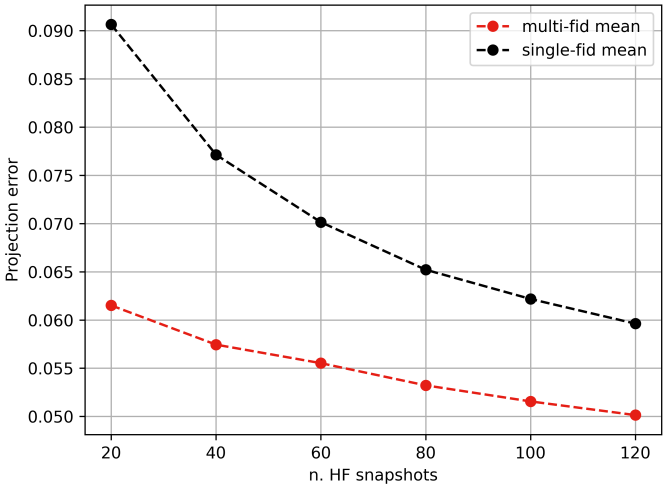


Figure 5.10: Mean projection error comparison between single and multi-fidelity PODs for different numbers of HF snapshots. Statistic on 70 HF validation snapshots.

The influence of the POD coefficient regression models is then considered in the reconstruction error e_{rec}, as in Equation 2.17. Similarly to the projection error, we consider the mean reconstruction error over the validation set, which becomes

$$\overline{e_{rec}} = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \frac{\left\| \left(\sum_{j=1}^{r} \left(\hat{a}_j(\mathbf{x}_i) \boldsymbol{\psi}_j \right) - \mathbf{u}_i \right) \right\|_2}{\|\mathbf{u}_i\|_2}, \tag{5.5}$$

where $\hat{a}_j(\mathbf{x})$ is the j-th POD coefficient's regression model, and $\mathbf{x}_i$ is the input parameter vector relative to the i-th validation snapshot $\mathbf{u}_i$. Figures 5.11 and 5.12 show the reconstruction error trends for different amounts of HF training snapshots. These figures illustrate the capabilities of both single and multi-fidelity ROMs to approximate unknown snapshots.

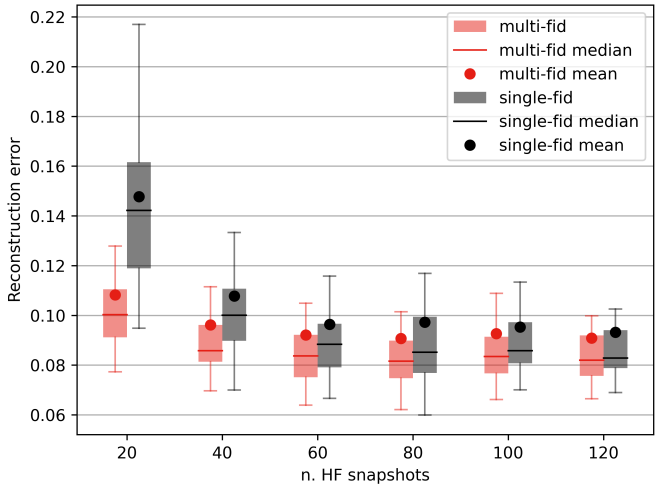


Figure 5.11: Reconstruction error boxplots comparison between single-fidelity ROM and multi-fidelity ROM for different numbers of HF snapshots. Statistics on 70 HF validation snapshots - outliers omitted.

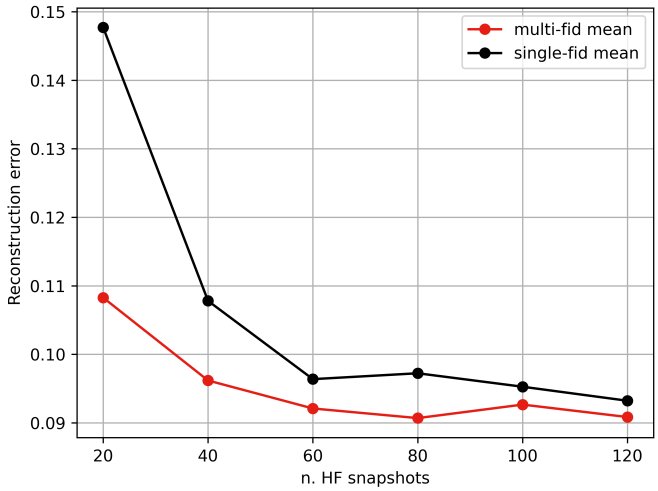
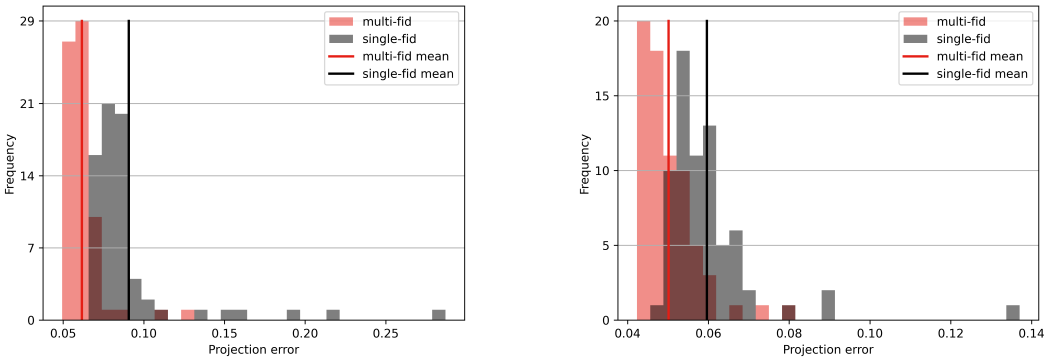


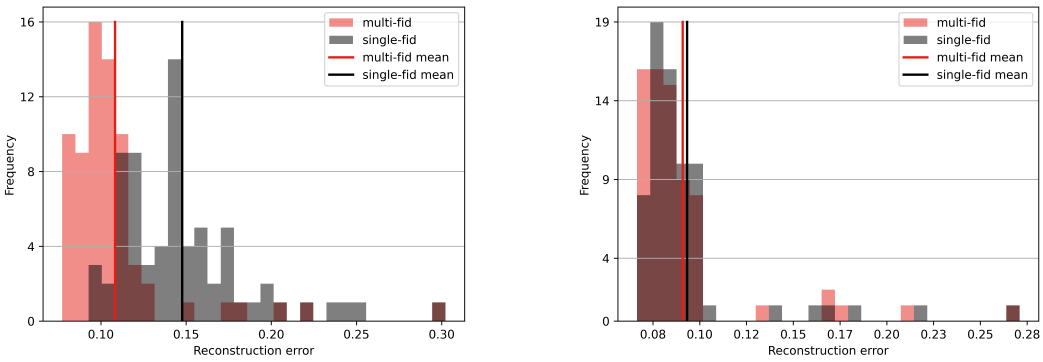
Figure 5.12: Mean reconstruction error comparison between single-fidelity ROM and multi-fidelity ROM for different numbers of HF snapshots. Statistics on 70 HF validation snapshots.

In Figures 5.13 and 5.14, the detailed distributions of projection and reconstruction errors have been represented, respectively, for both the extreme configurations with 20 and 120 HF training snapshots. Conversely to Figures 5.9 and 5.11, outliers are represented too.



(a) 20 HF training snapshots (b) 120 HF training snapshots

Figure 5.13: Comparison of single/multi-fidelity projection error distributions on 70 HF validation snapshots.



(a) 20 HF training snapshots (b) 120 HF training snapshots

Figure 5.14: Comparison of single/multi-fidelity ROM reconstruction error distributions on 70 HF validation snapshots.

The ROM reconstruction errors in Figures 5.11, 5.12 and 5.14 allow for comparing the differences between the single and multi-fidelity ROM results. However, to assess whether they provide a good approximation of the HF CFD solution, namely the full-order model, the results can be compared directly with the LF CFD solutions for the validation set, presented in Figure 5.15.

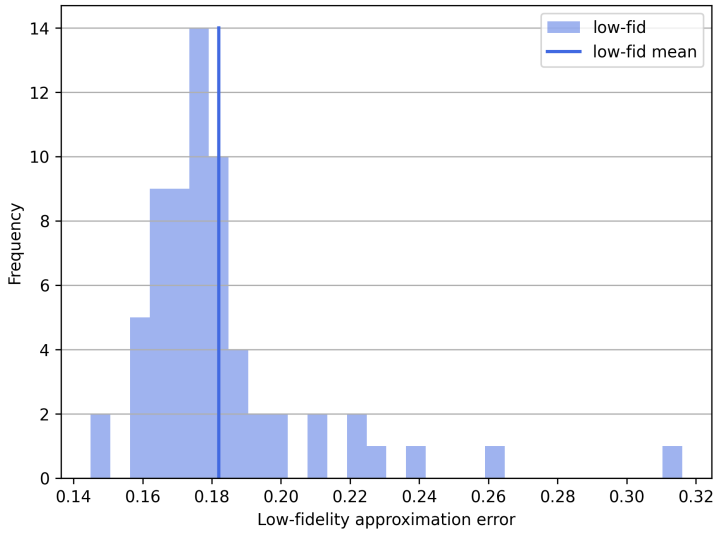


Figure 5.15: LF approximation of 70 HF validation snapshots; mean error of 18.2%.

Here, the LF solutions are used to directly approximate the HF snapshots reserved for validation, and it can be seen that both single- and multi-fidelity ROMs are better HF surrogates than the LF solver. To measure the mean LF approximation error $\overline{e_{lf}}$ in Figure 5.15, the error has been computed as in Equation 5.6, consistently with the metrics of Equations 5.4 and 5.5.

$$\overline{e_{lf}} = \frac{1}{n_{val}} \sum_{i=1} n_{val} \frac{\left\| \mathbf{u}_i^{LF} - \mathbf{u}_i \right\|_2}{\left\| \mathbf{u}_i \right\|_2} , \tag{5.6}$$

where $\mathbf{u}_i$ is a validation HF snapshot, and $\mathbf{u}_i^{LF}$ is the respective LF snapshots. For the same validation designs, in Figure 5.16, it is possible to observe the differences in the drag coefficient C_D relative to the LF and the HF CFD results. The C_D, defined in Equation 5.7, is an important quantity of interest in external aerodynamics, strongly connected to the pressure distribution on the body's boundary.

$$C_D = \frac{2F_D}{A_f \rho U_0^2} , \tag{5.7}$$

where F_D is the drag force, A_f is the frontal area of the body and ρ is the fluid density. Together with the results of Figure 5.15, it is evident the LF solver has difficulty matching the HF solver's performances.

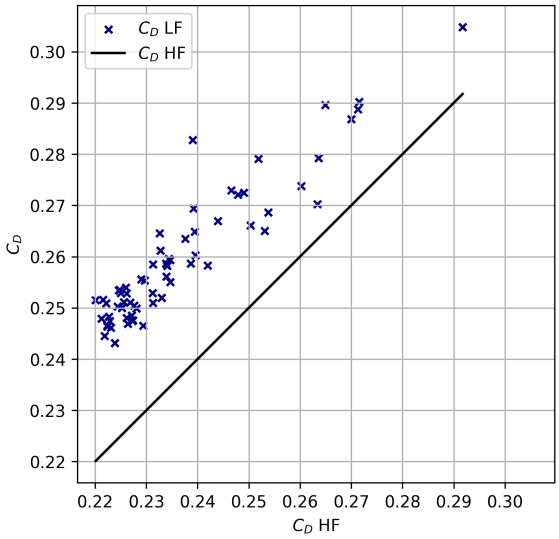


Figure 5.16: HF and LF C_D values corresponding to the validation designs. The car's frontal area is kept constant, equal to the reference vehicle frontal area.

Tables 5.2 and 5.3 report the time needed for the CFD simulations and the ROMs training, respectively. Each HF CFD solution is 4 times more expensive than the LF one. On the other hand, the ROMs training is at least 1 or 2 orders of magnitude less time-consuming than a single CFD simulation, without taking into consideration the necessary offline CFD evaluations. Being in the order of milliseconds, the prediction time is negligible for both the single and the multi-fidelity ROM, enabling almost real-time capabilities.

Table 5.2: CFD solvers solution time. All calculation on 96 CPU cores (AMD EPYC 7413).

	Low-fidelity		High-fidelity	
	Wall time	Core hours	Wall time	Core hours
Average solution time	24m59s	39.97h	1h39m09s	158.64h
Time ratio respect LF	1		3.97	

Table 5.3: ROMs wall time - 20 HF snapshots (plus 160 LF snapshots for the multi-fidelity ROM).

	Single-fidelity	Multi-fidelity
Decomposition time	1.2s	42.1s
Training time	24.2s	2m45s
Number of modes	15	99
Prediction time (1 snapshot)	$5.3 \cdot 10^{-4}$s	$2.4 \cdot 10^{-3}$s
Total time	25.4s	3m27s

Table 5.2 shows that the major contributor to the ROM computational time cost is the offline CFD phase. If we look at the projection and reconstruction errors with respect to the core hours needed for the offline phase, the results of Figures 5.10 and 5.12 become like those presented in Figures 5.17 and 5.18.

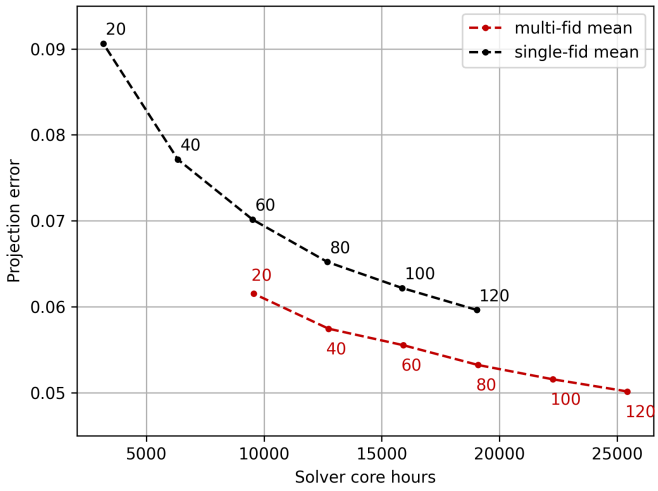


Figure 5.17: Mean projection error on the validation set versus the off-line core hours needed for the CFD solutions. Numbers in the plot indicate how many HF snapshots are used. All multi-fidelity errors are obtained with 160 LF snapshots.

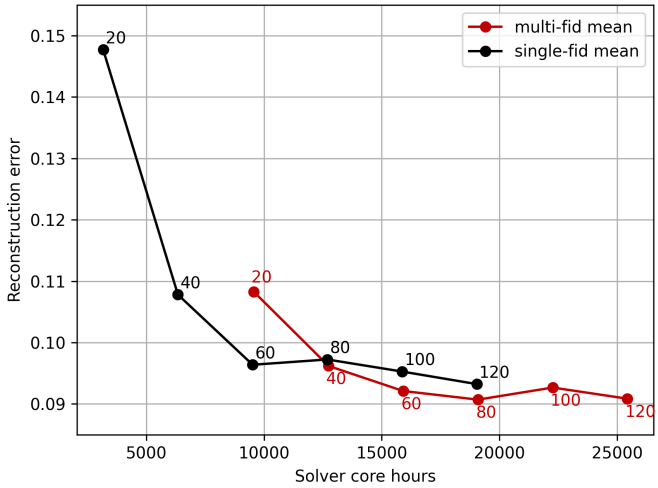


Figure 5.18: Mean reconstruction error on the validation set versus the off-line core hours needed for the CFD solutions. Numbers in the plot indicate how many HF snapshots are used. All multi-fidelity errors are obtained with 160 LF snapshots.

In terms of projection error, Figure 5.17 shows that the mixing of LF and HF snapshots at the POD stage is beneficial from an efficiency standpoint. On the other hand, the reconstruction error in Figure 5.18 shows that the multi-fidelity approach improves its efficiency with respect to the single-fidelity ROM with 40 HF snapshots or more. This is due to the cost of computing 160 LF solutions, and it is specific to this test case. Other LF formulations for different test cases can lead to different efficiencies; however, we want to focus on the fact that the poor quality LF information can improve the HF representation of the problem. This can be seen in Figure 5.18 and in Figure 5.12, where it is even more evident.

Keeping in mind the multi-fidelity ROM setup, we decided to utilize multi-fidelity regression models only for the first 4 POD modes' coefficient models. The main motivation was that the LF snapshots carried useful information mainly with respect to the first POD modes coefficients, as discussed in Section 5.1.1. Indeed, Figure 5.19 shows what happens to the reconstruction error when increasing the number of POD coefficients treated with a multi-fidelity regression model for the 20 HF snapshots case.

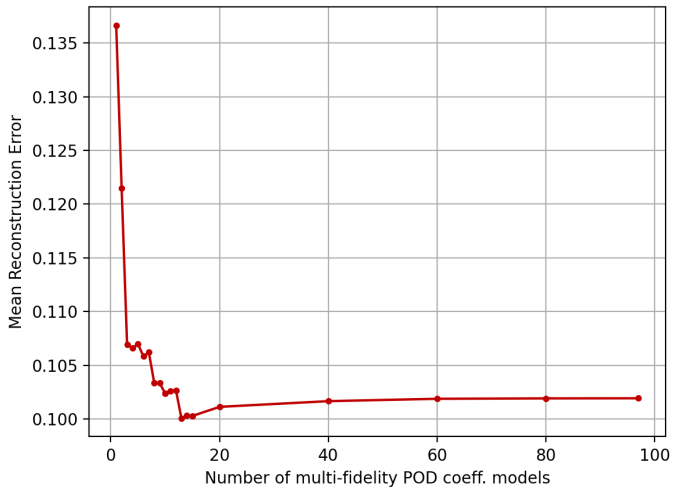


Figure 5.19: Mean reconstruction error on validation for different number of POD coefficients modeled with the a NARGP multi-fidelity regression model. Using 20 HF snapshots and 160 LF snapshots.

Here, it is clear that the first POD modes are the ones that greatly benefit

from the improved accuracy of the NARGP. After, the positive returns in accuracy are minimal, or even slightly worse performing. From 15 to 20 multi-fidelity POD coefficient models, there is an almost negligible 0.16% increase in the reconstruction error, and after 20 multi-fidelity models, it remains constant. A possible reason is that the less energetic POD modes might be representative of high-frequency behaviors, and the consequent LF POD coefficients, obtained by projection, result to be noisy. Therefore, the NARGP is trying to leverage noise instead of meaningful LF information, performing at most as a single-fidelity GP regression model. The choice to use only the first 4 modes is mostly didactic, since it emphasizes where and how multi-fidelity regression models have the maximum effect. Even though there is a reduction in the offline ROM training time, if the computational cost of training more multi-fidelity models is compared to the snapshot generation, the relative advantage is modest.

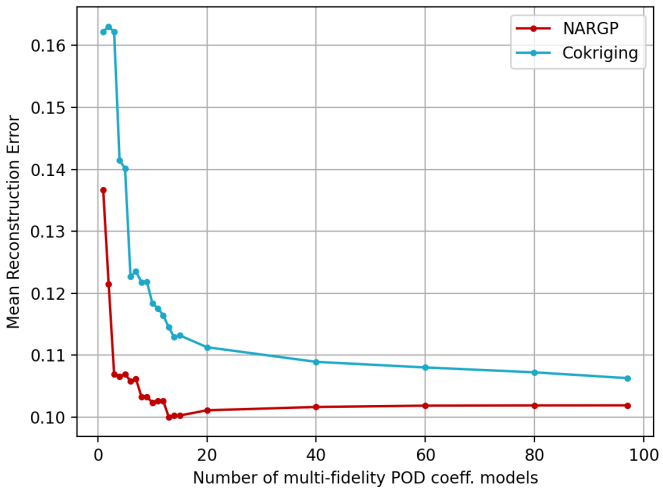


Figure 5.20: Mean reconstruction error on validation for different number of POD coefficients modeled with NARGP and Cokriging multi-fidelity regression models. Using 20 HF snapshots and 160 LF snapshots.

Another decision was to have NARGP as the multi-fidelity regression model of choice. In Section 4.1.2, we discuss the matter, and in summary, NARGP was chosen due to its intrinsic flexibility and ease of use, which was perfect for real-world industrial problems. Figure 5.20 supports this thesis, showing what happens to the mean reconstruction error when using the Cokriging model, a well-established and

comparable alternative to NARGP, instead of the multi-fidelity model of choice. In Figure 5.20, we considered the 20 HF snapshot case, similarly to Figure 5.19, since the multi-fidelity methodologies are designed to work in HF data scarcity conditions.

Discussion

In this experiment, a multi-fidelity approach to reduced order modeling was applied to an industrial problem. The aim of the work is to determine whether the multi-fidelity ROM can improve in accuracy over an equivalent single-fidelity ROM, while reducing the amount of HF information needed to train the ROM. Due to the test case's nature, these objectives were pursued while handling non-coherent information sources and a wide range of geometrical deformations. In light of the previously presented results, several conclusions can be inferred.

First of all, the addition of LF snapshots at the decomposition stage has a great impact on the POD basis representability of unseen snapshots. This is easily noticeable in Figure 5.10, where the single-fidelity modes produce projections between 1% to 3% less accurate than the multi-fidelity POD, where LF snapshots were mixed with the HF ones.

Secondly, not only did the overall representation capabilities of the basis improve, but the multi-fidelity POD also reduces the projection error in the outlier configurations. Being the configurations with the highest errors, it is advantageous to enhance the modes' capability to represent them.

Finally, there is an improvement too in the reconstruction error, especially with fewer HF snapshots, which was the actual purpose of the multi-fidelity ROM. The increased accuracy can be appreciated in Figure 5.12, where the multi-fidelity approach reaches lower reconstruction errors with fewer HF snapshots than the single-fidelity equivalent. It is important to observe that the HF training snapshots are nested in the LF training snapshots to ease the NARGP models' training. As mentioned in Section 5.1.1, 160 LF snapshots were added to the n_{HF} HF training snapshots; therefore, only $160 - n_{HF}$ LF snapshots actively contributed to better exploring the design space. This justifies the superior improvements with fewer HF training snapshots and, consequently, the lesser ones when approaching 120 HF training snapshots. Furthermore, this behavior suggests a certain degree of robustness of the POD to the LF information, since it does not deteriorate the quality of the HF representation.

Given all these considerations, the proposed multi-fidelity POD ROM was able to augment the capabilities of an analogue single-fidelity NI ROM, both in terms of representability and accuracy. The advantages in terms of computational costs are inevitably related to the choice of the LF model, and, as a downside, the snapshots

mapping strategy has to be tailored to the specific test case. In this industrial problem, the computational cost-accuracy ratio breaks even around 40 HF snapshots, while adding more HF information diminishes the returns of the multi-fidelity model with the given LF snapshots. Nevertheless, with this experiment, we proved that the HF approximation of a ROM could benefit from otherwise useless LF information.

5.2 Annular pipe with wavy surfaces

Heat exchangers are employed in numerous industrial applications, and the performance of these devices is deeply connected to their geometries. Annular wavy pipes [66, 67, 68] became of interest for compact heat exchangers, nuclear reactors, microfluidic devices, etc. In this experiment, we performed a parametric study of the flow and temperature fields inside a 3D annular pipe with wavy surfaces. In particular, we consider a problem with 5 parameters, 4 of which are used to parameterize the pipe surface geometry, and the remaining one is the Reynolds number. The main goal is to obtain an automatic workflow, from snapshot generation to the ROM training. This is achieved by leveraging the inherent characteristics of a meshless CFD solver [20] and NI ROMs.

This section is based on our earlier work published in [22]. While the principal findings remain consistent, the presentation has been adjusted to suit the context and objectives of the present thesis.

5.2.1 Problem Description

We considered an annular pipe with parameterized wavy surfaces. The pipe's walls have an analytical definition, dependent on the 4 geometrical problem parameters, and we assume them to be symmetric with respect to two orthogonal vertical planes. Let us consider the cylindrical coordinates r, θ, z, respectively, the radius, the angular position, and the height. A point inside the pipe satisfies Equations 5.8-5.10.

$$r_{\min} \leq r \leq r_{\max}\,, \tag{5.8}$$

$$0 \leq \theta \leq \frac{\pi}{2}\,, \tag{5.9}$$

$$0 \leq z \leq H\,, \tag{5.10}$$

where

$$r_{\min} = R_1 + A_1\delta(\theta, z, \omega_z, \omega_\theta)\,, \tag{5.11}$$

$$r_{\max} = R_2 + A_2\delta(\theta, z, \omega_z, \omega_\theta)\,, \tag{5.12}$$

being R_1 and R_2 the internal and external radii and A_1 and A_2 the internal and external multiplier factors for the wavy function $\delta = \cos(2\omega_\theta\theta - 1)\cos(2\omega_z z/H - 1)$, which is dependent from the other two cylindrical coordinates θ, z and from the

two frequency parameters ω_z, ω_θ; finally, H is the height of the considered pipe segment. All together, $A_1, A_2, \omega_z, \omega_\theta$ are the problem's geometrical parameters, while $R_1 = 0.4, R_2 = 0.6$ and $H = 1$ are fixed.

If we plot radial and horizontal slices of the wall pipes it is possible to observe the effect of the wavy functions, as in Figures 5.21 and 5.22 .

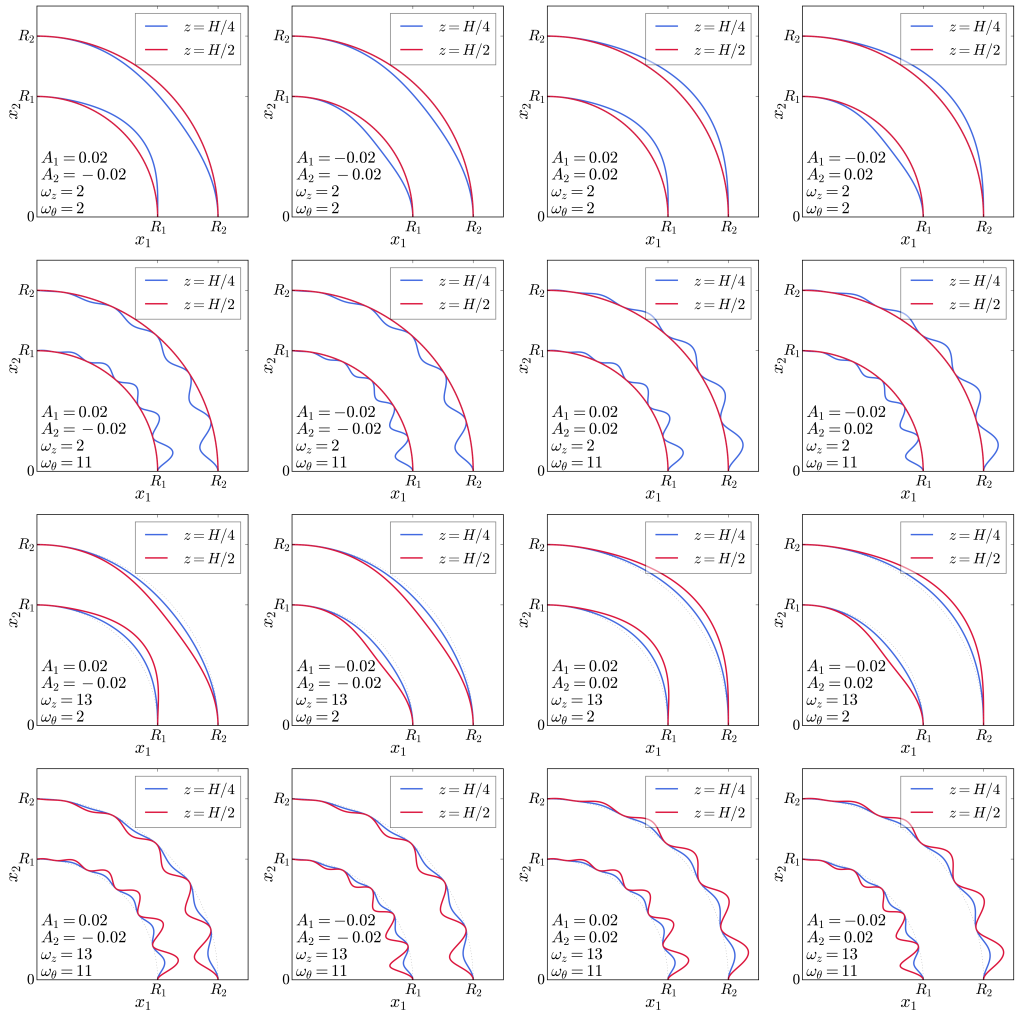


Figure 5.21: Horizontal slices of the computational domain at $z = H/4$ and $z = H/2$, with $z = x_3$ coordinate in the cartesian reference frame. All 16 extreme combinations of the geometrical parameters.

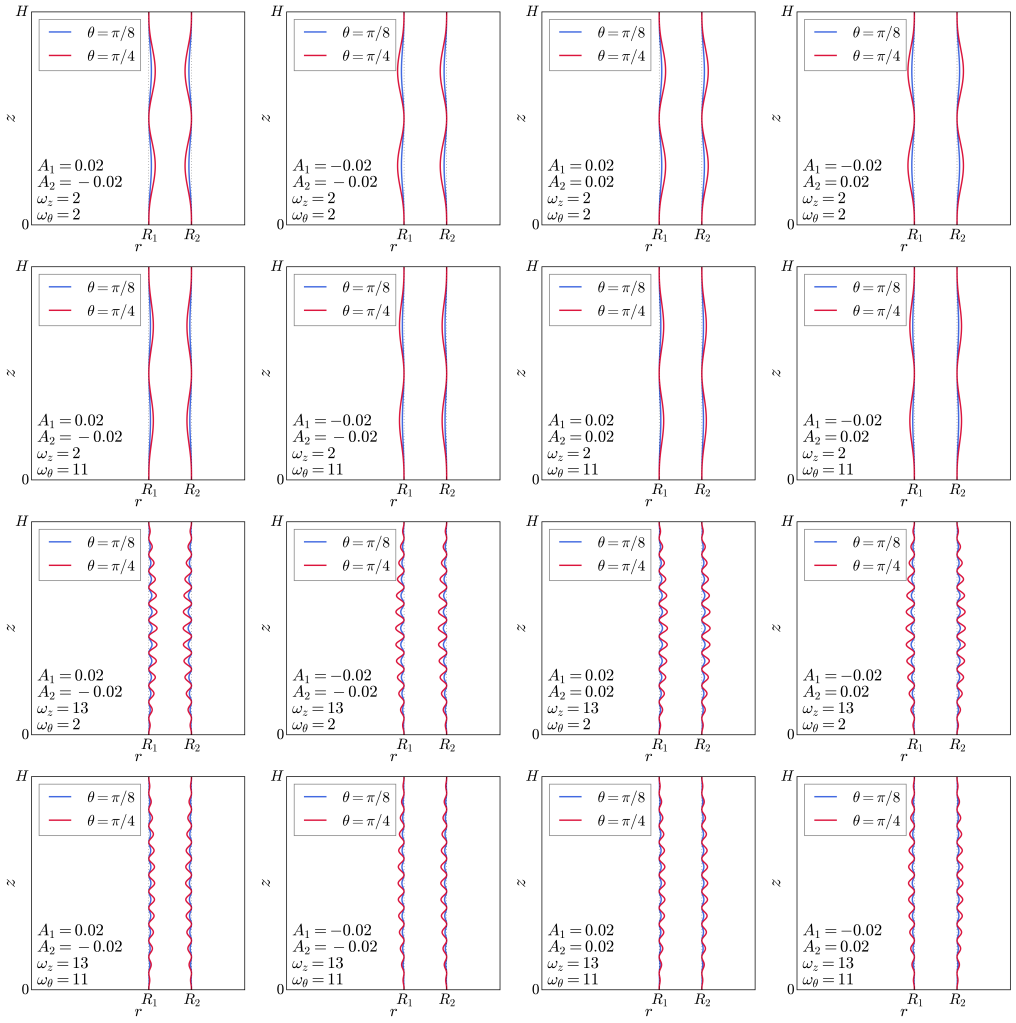


Figure 5.22: Radial slices of the computational domain at $\theta = \pi/8$ and $\theta = \pi/4$. All 16 extreme combinations of the geometrical parameters.

Parameterization

The problem has been parameterized with four geometric parameters, namely A_1, A_2, ω_z, ω_θ, as described in Section 5.2.1, and one operational parameter, the Reynolds number Re. Table 5.4 shows the ranges of variation for all the input parameters that have been considered for this study.

Table 5.4: Input parameters' ranges of variation.

Parameter	Type	Lower bound	Upper bound
A_1	geometrical	-0.02	0.02
A_2	geometrical	-0.02	0.02
ω_z	geometrical	2	11
ω_θ	geometrical	2	13
Re	operational	100	350

The effect of the geometrical parameters is visible in Figures 5.21 and 5.22, leading to complex wall shapes. To see the parameterization effect in 3D, we can observe Figure 5.23, where internal and external walls are colored according to the radial coordinate. The Reynolds number has no impact on the geometry and is limited to 350 to ensure steady and laminar solutions for all possible geometrical configurations.

Meshless solver setup

In this work, we use a meshless solver; thus, the solution is based on a distribution of computational nodes inside the fluid domain, avoiding the need for a grid. Indeed, the nodes do not have connectivity, contrary to traditional approaches, nor the strict quality requirements typical of computational meshes. The considered meshless methodology is Eulerian, and it is based on RBF finite differences (FD). Here, it is utilized to solve an incompressible, laminar, and steady-state forced convection problem inside the wavy pipe. In essence, the RBF-FD is composed of three steps:

1. node generation;
2. RBF-FD collocation;
3. solution.

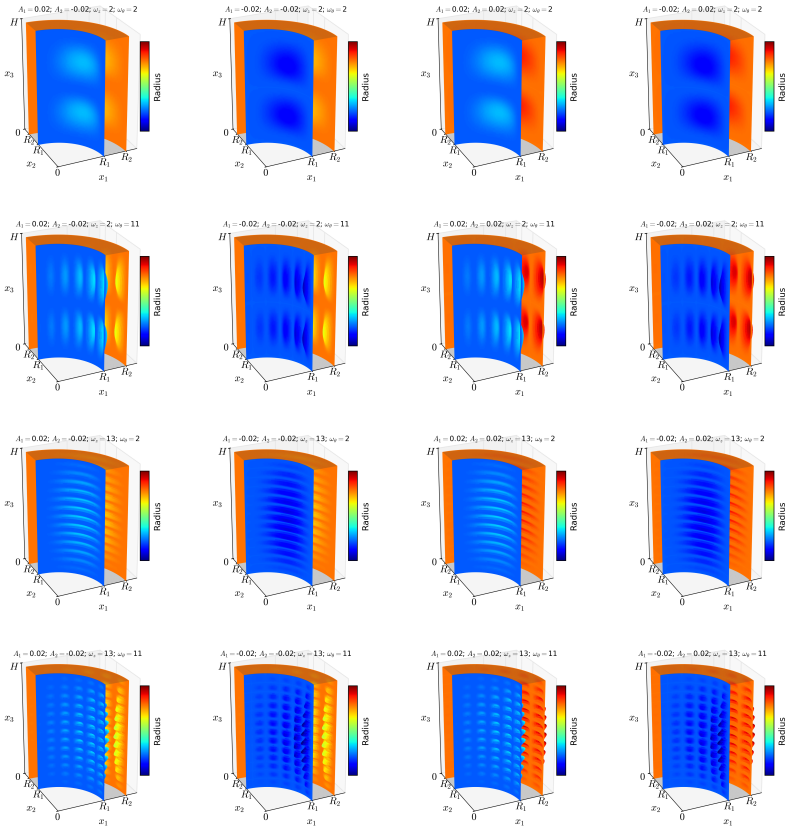


Figure 5.23: View from the inside of the quarter pipe's internal and external walls, for all 16 extreme combinations of the geometrical parameters. Colors represent the radial coordinates of the wall's points.

The node generation creates a non-uniform distribution of nodes exploiting a specific spacing function. The RBF-FD collocation method enables the discretization of the PDEs of interest, and it requires constructing local RBF expansions around each node. Finally, the solution procedure is iterative and continues until the steady-state convergence is met. The principles of the RBF-FD meshless methods are out of the scope of this thesis, and are deeply discussed in [21, 20, 22].

Similarly to Section 5.1, we will consider 2 solvers, a LF one and a HF one. The main difference stands in the geometrical domain discretization: the LF solver counts $150k$ nodes, while the HF solver needs $750k$ nodes. All the other specifications are common between the two solvers. For both LF and HF problems, we are interested in the velocity, pressure, and temperature fields, here named U, p, T, solving the conservation equations of mass, momentum, and energy, respectively:

$$\nabla \cdot \mathbf{U} = 0\,, \tag{5.13}$$

$$(\mathbf{U} \cdot \nabla)\mathbf{U} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{U}\,, \tag{5.14}$$

$$\mathbf{U} \cdot \nabla T = \frac{1}{Re\,Pr}\nabla^2 T\,, \tag{5.15}$$

with Pr being the Prandtl number, here constant $Pr = 0.71$. For the solution, U, p, T are made nondimensional by taking $L, U_0, \rho U_0^2$ and ΔT as reference quantities, being respectively the pipe width, the maximum inlet velocity, the reference kinetic energy, and the temperature difference between the inlet and the walls. As previously mentioned, we consider a quarter of the geometry, supposing that the geometry is symmetrical with two radial planes placed at $\theta = 0$ and $\theta = \pi/2$. The two walls are modeled with no-slip conditions and $T = 0$. The inlet velocity profile is non-zero velocity only along the z direction, having

$$U_z(r) = \frac{q_2 + q_1 \ln r - r^2}{q_3}\,, \tag{5.16}$$

with q_1, q_2, q_3 depending only on the pipe dimensions

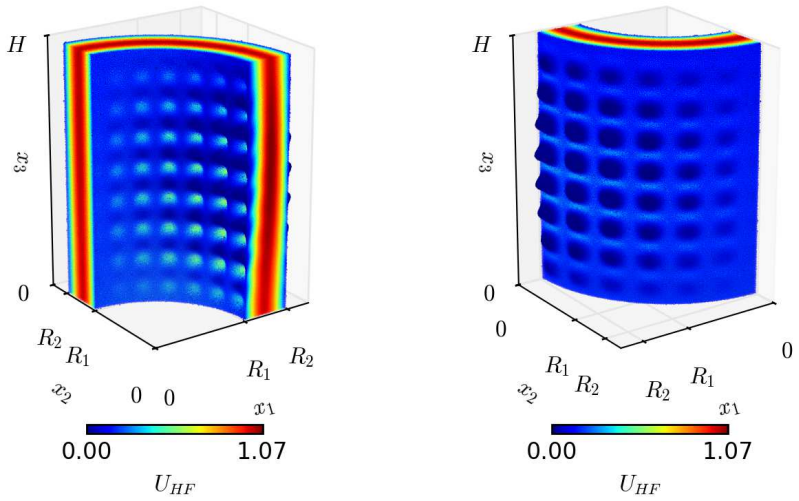
$$q_1 = \frac{R_2^2 - R_1^2}{\ln R_2/R1}\,, \tag{5.17}$$

$$q_2 = \frac{R_1^2 \ln R_2 - R_2^2 \ln R_1}{\ln R_2/R1}\,, \tag{5.18}$$

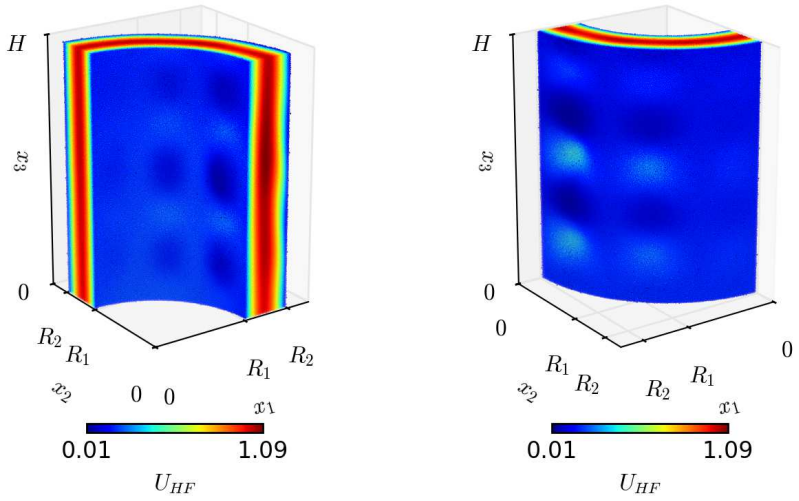
$$q_3 = \frac{q_1}{2}\left(\ln \frac{q_1}{2} - 1\right) + q_2\,, \tag{5.19}$$

and for the temperature profile at the inlet, $T(r) = U_z(r)$, as in Equation 5.16. The outlet is at $z = H$, where we consider $p = 0$ and $\partial T/\partial n = 0$.

Figures 5.24 (a) and (b) show examples of the solution on the internal nodes for two different designs. The respective temperature fields are presented in Figures 5.25 (a) and (b).

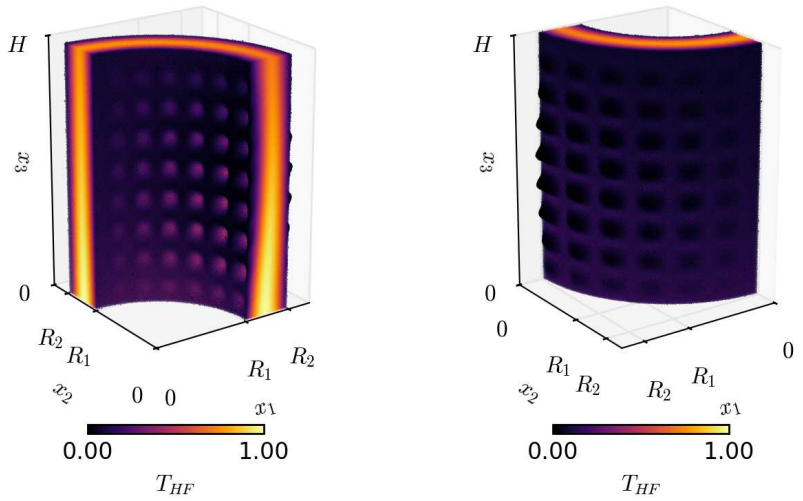


(a) $Re = 256.4, A_1 = 0.0182, A_2 = 0.0186, \omega_z = 8.3, \omega_\theta = 13.0$

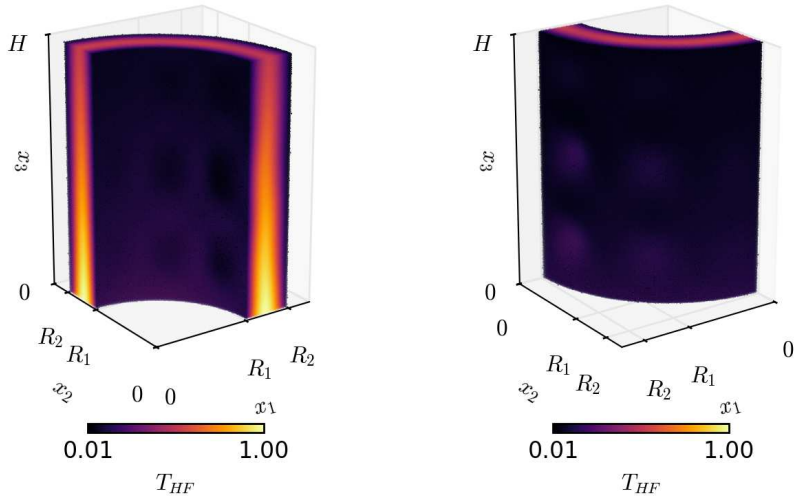


(a) $Re = 115.8, A_1 = -0.0124, A_2 = 0.0185, \omega_z = 2.7, \omega_\theta = 5.4$

Figure 5.24: HF velocity magnitude fields on RBF-FD meshless internal nodes for two different designs. On the left, the internal wall view, on the right, the external wall view.



(a) $Re = 256.4, A_1 = 0.0182, A_2 = 0.0186, \omega_z = 8.3, \omega_\theta = 13.0$



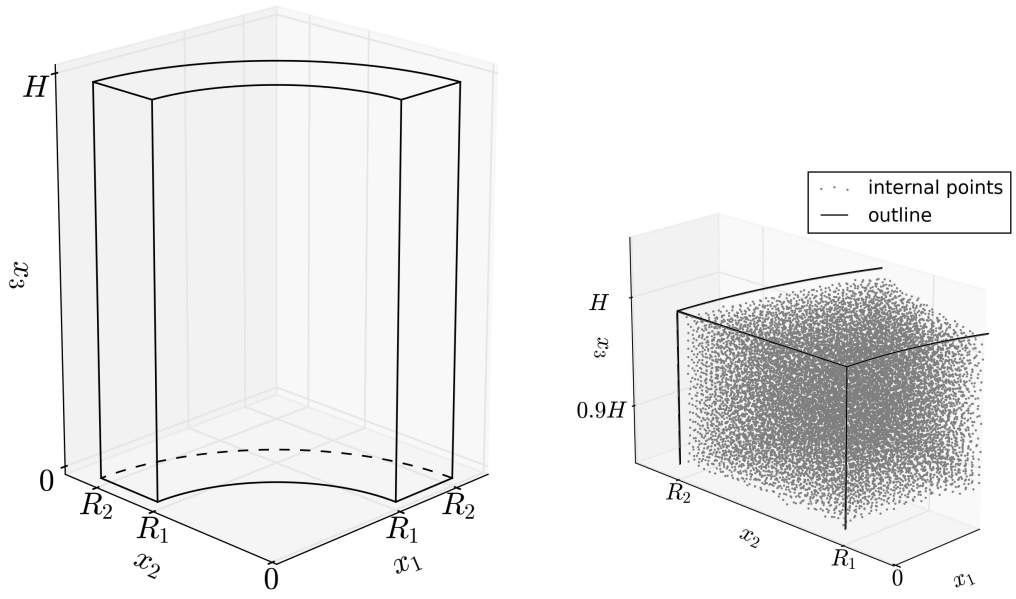
(a) $Re = 115.8, A_1 = -0.0124, A_2 = 0.0185, \omega_z = 2.7, \omega_\theta = 5.4$

Figure 5.25: HF temperature fields on RBF-FD meshless internal nodes for two different designs. On the left, the internal wall view, on the right, the external wall view.

Design of experiments

The parameter space has been sampled with a Halton sequence for the 5 input parameters, identifying the 230 experiments. In the results section, we will consider different HF training set sizes, ranging from 20 to 170 designs. The last 30 HF designs from the sequence are reserved for validation to assess the ROM's performance. For the LF training set, we used all 230 designs.

Mapping



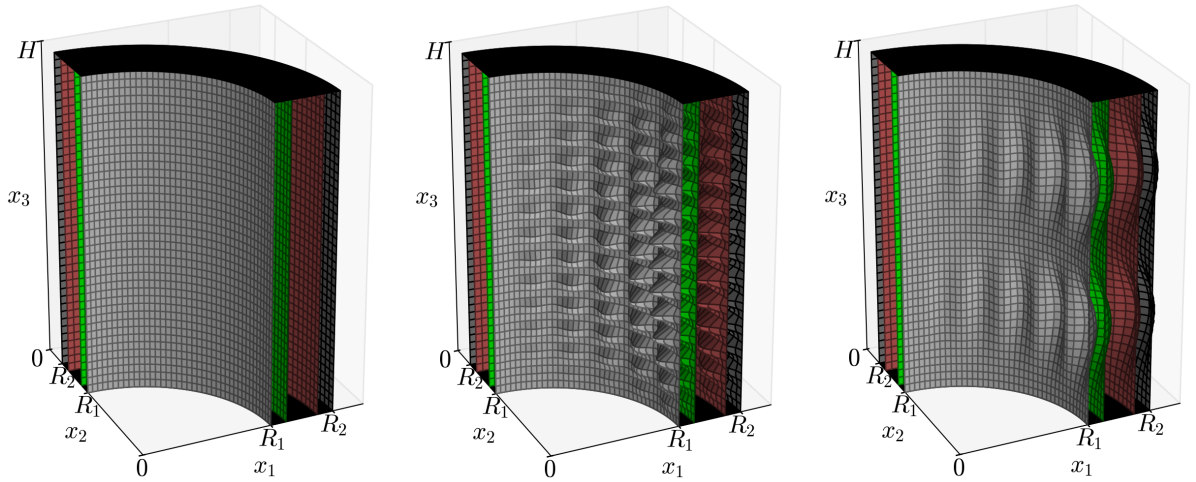
(a) Outline. (b) Close-up with nodes.

Figure 5.26: Reference cylinder Ω_{ref}.

Since the LF and HF solvers work with different discretizations, the latter having 5 times the number of computational nodes, all solutions need to be mapped to a reference discretization to enable POD. Moreover, four out of the five problem parameters are geometrical, adding even more variability to the discretization. Since the geometrical description of the domain is analytical, specifically to this problem, it is possible to perform a mapping even though we do not know the actual deformation field. Indeed, we obtained a valid transformation that can move a node from a reference discretization Ω_{ref} to the locus of points of the deformed geometry.

The first step is to define the reference nodes. To do so, we used a fine discretization of an undeformed pipe, where the amplitude parameters $A_1 = A_2 = 0$. Figure 5.26 shows the outline of the reference cylinder and gives an idea of the node distribution.

Successively, we can exploit the definition of the internal and external wall radial coordinates of Equations 5.11 and 5.12. The idea is to find a function $A(r) : r \to A$ that associates an amplitude value A to any $r \in [R_1, R_2]$ intermediate radial coordinate. A possibility is to define $A(r)$ as a linear interpolation between the points (R_1, A_1) and (R_2, A_2). Intuitively, $A(r)$, together with the other parameters ω_z, ω_θ, identifies an intermediate surface between the internal and external wall, and Figure 5.27 shows how 2 intermediate surfaces are transformed according to the parameterization.



(a) Reference geometry

(b) $A_1 = -0.02$, $A_2 = -0.02$, $\omega_z = 13$, $\omega_\theta = 11$

(c) $A_1 = 0.02$, $A_2 = 0.02$, $\omega_z = 2$, $\omega_\theta = 11$

Figure 5.27: . Mapping example for two different combinations of geometric parameters. The transformation of two internal level sets (red and green), which correspond to cylinders in the reference geometry, are shown. The mesh-like features on the surfaces are purely illustrative.

By doing so, a point $\mathbf{P} = (r, z, \theta) \in \Omega_{ref}$ can be transformed, obtaining $\mathbf{P}' = (r', z', \theta') \in \Omega$, where Ω is the deformed domain, with a smooth and unique transformation, here noted as $\mathcal{M}$

$$\mathcal{M}(\mathbf{P}) = \mathbf{P}' = \begin{Bmatrix} r + A(r)\delta(\theta, z, \omega_z, \omega_\theta) \\ \theta \\ z \end{Bmatrix}, \tag{5.20}$$

having, in practice, a radial displacement of the reference points. Once all nodes belonging to the reference discretization are transformed into their deformed equivalent, it is possible to interpolate the CFD solutions from the computational nodes to the transformed reference ones. This can be done with different interpolation techniques, and we opted for a k-nearest neighbor interpolation [69] since we do not have a node triangulation, and this technique can handle large cloud points with ease.

ROM setup

We recall that the aim of the work is to approximate a total of three fields, U, p, T. The velocity field is considered as a single field even though it is a vector field with three components. This can be achieved by considering all three components of a velocity field as a single snapshot of three times the dimension. Therefore, there will be a total of three single-fidelity ROMs and three multi-fidelity ROMs.

The decomposition has been done with a POD energy $\varepsilon = 0.9999$. We observed, as Figure 5.31 in the results section will show, that considering more POD modes does not bring significant advantages in terms of the basis representability.

The GP regression models have been trained with squared-exponential kernels, as in Equation 3.11. For the multi-fidelity ROM, we used NARGP as the multi-fidelity regressor, again with squared-exponential kernels. POD coefficients are assumed to be slightly noisy, and for all GP-based models, the noise standard deviation hyperparameter is kept in the order of $10^{-2}/10^{-3}$ of the normalized output. To improve the robustness of the hyperparameter optimization during training, we utilized a low-memory BFGS with 15 random restarts.

We mixed together LF and HF snapshots to perform the decomposition for the multi-fidelity ROM, as described in Chapter 4, enhancing the representability of the POD basis. Figure 5.28 shows the singular values decay for different POD basis approaches, ranging from HF-only snapshots, LF-only snapshots, and a mixed fidelity matrix of snapshots. Indeed, the larger the singular value, the more important the corresponding dimension is for capturing the variability of the data. Here, the multi-fidelity ROM decomposition counts many more modes compared to the HF one.

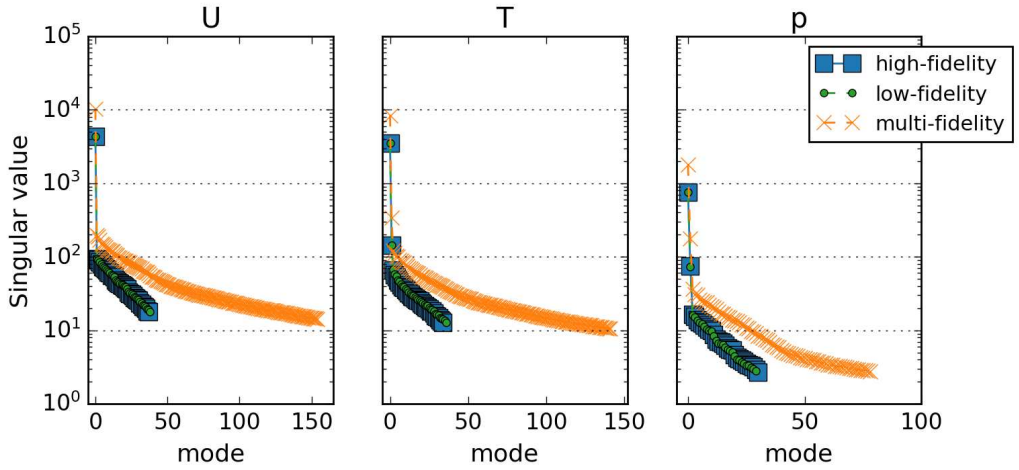


Figure 5.28: Singular values for HF, LF and multi-fidelity PODs, from 50 HF snapshots, 50 LF snapshots and 50 HF + 230 LF snapshots, respectively.

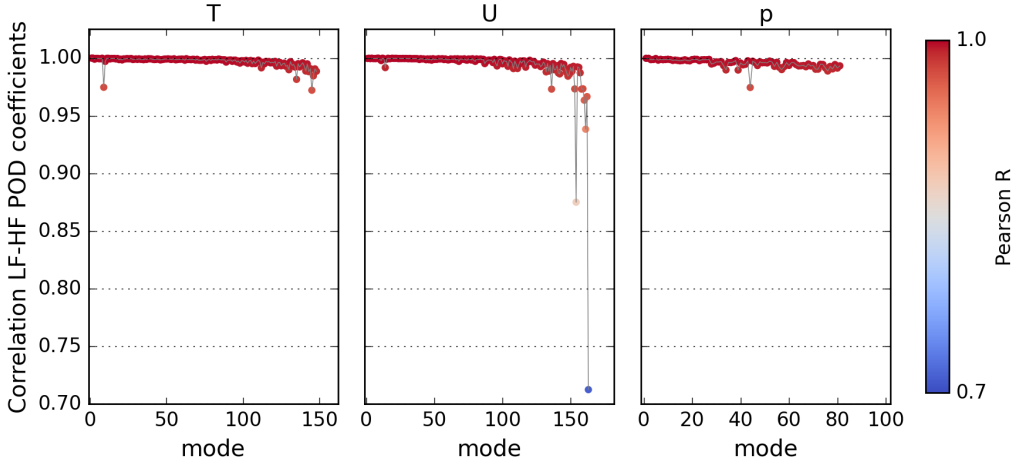


Figure 5.29: Correlation analysis with Pearson's R test between LF and HF representation of the POD coefficients for the multi-fidelity ROM obtained with 170 HF plus 230 LF snapshots.

If we compute Pearson's R correlation coefficient between the LF and HF POD coefficients relative to the mixed fidelity basis, a high linear correlation is observed. Figure 5.29 shows the Pearson's R correlation coefficient trend for all three fields of interest: temperature, velocity, and pressure. It is possible to notice that the

correlation is rather high, being close to 1 for most of the POD modes, with an almost 0 p-value, close to machine error, for all of the linear correlation coefficients. Consequently, we choose to model all POD coefficients with multi-fidelity regression models.

Out of curiosity, Figure 5.30 offers a visual representation of the first 6 POD modes for the HF POD of the velocity field.

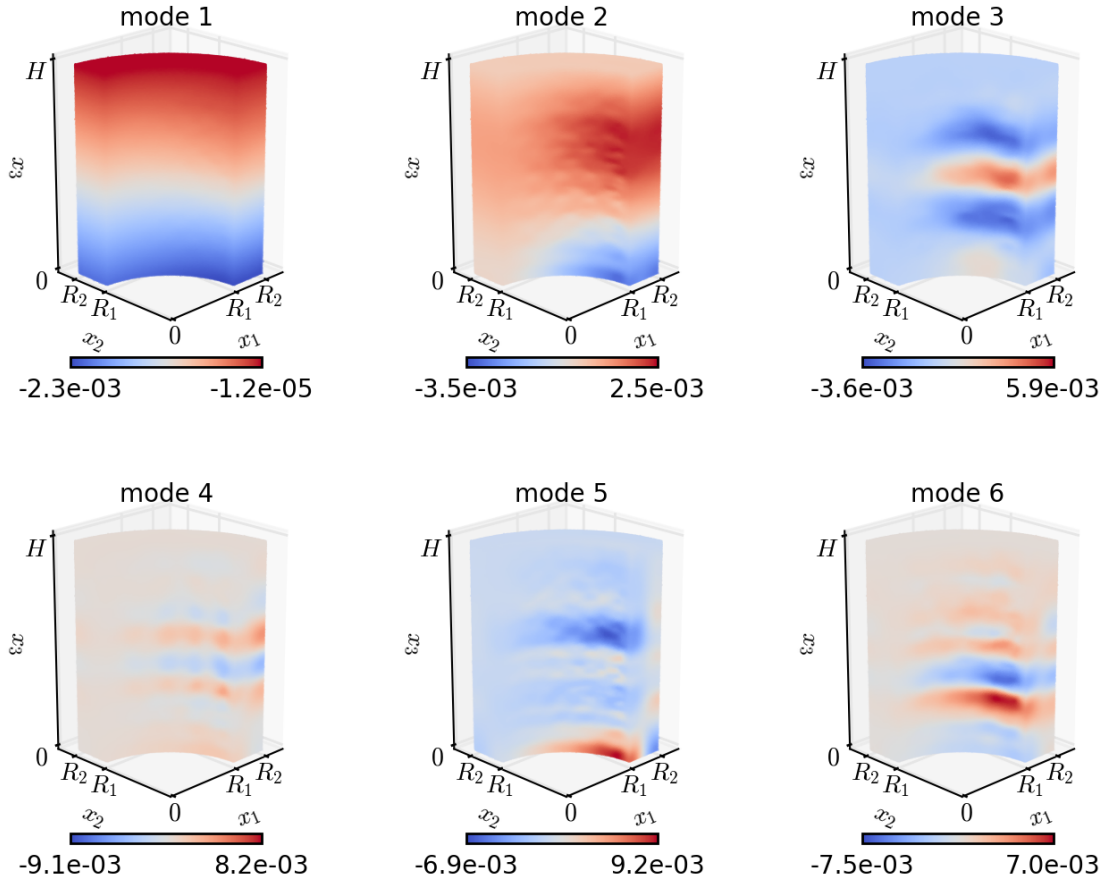


Figure 5.30: First six POD modes of the pressure field on the reference domain computed with 50 HF pressure snapshots.

5.2.2 Results

In this section, the ROMs have been trained with 20 to 170 HF snapshots, and all error metrics are computed with 30 HF snapshots, reserved for validation, from the tail of the DoE Halton sequence. Therefore, validation designs evenly cover the design space and avoid overlapping with the training snapshots. If the convex hull of the training designs is considered, part of the validation designs will fall outside of the hull, and part of them will stay inside. This implies that, depending on the training set, the ROM predictions might be extrapolating the fields of interest.

The first metric that we are going to consider is the projection error, which measures the POD's capability to represent unseen snapshots with the POD basis functions. Recalling Equation 2.18, another projection error interpretation is, intuitively, the best possible approximation that the ROM can provide under the assumption that there is no interpolation error due to the regression models. Figure 5.31 shows the projection error trends for different numbers of training snapshots and different POD energy thresholds, for all fields T, U and p.

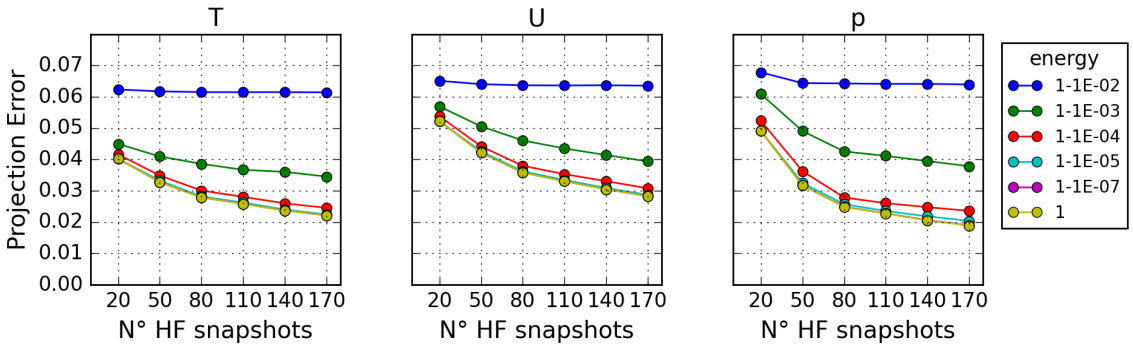


Figure 5.31: Mean projection error on validation for T, U and p single-fidelity HF ROMs, given different POD energy thresholds and different numbers of HF training snapshots.

Figure 5.32 presents the reconstruction and projection error of the final ROM approximations for both the multi-fidelity and single-fidelity HF cases. In the multi-fidelity ROM, 230 LF snapshots have been added, and for both the HF and multi-fidelity ROMs, the considered HF snapshots are always nested between each other: the 20 HF snapshots ROMs share 20 of the 50 HF snapshots with the 50 HF snapshots ROMs, and so on until 170 HF snapshots.

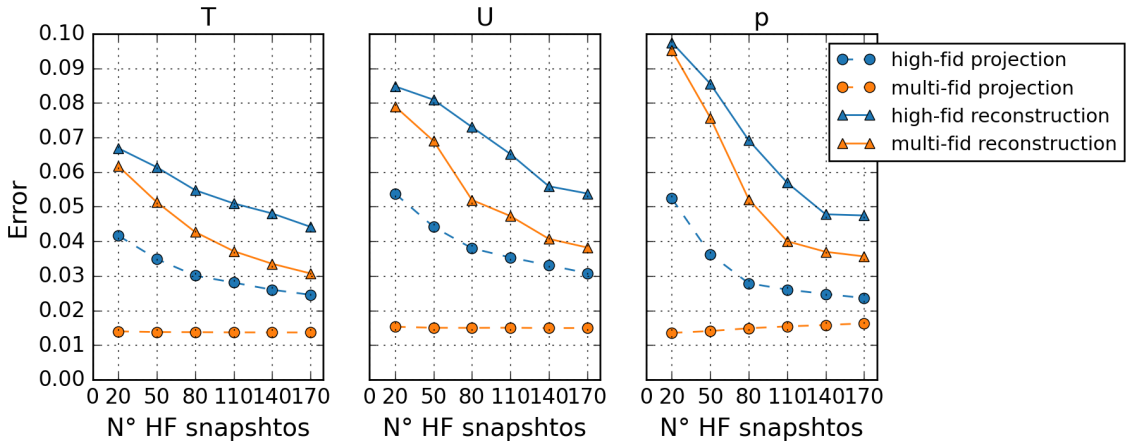


Figure 5.32: Mean reconstruction and projection errors on validation, for the T, U and p HF and multi-fidelity ROMs, for different numbers of HF training snapshots. 230 LF snapshots have been utilized for the multi-fidelity ROMs.

In Figure 5.32 the multi-fidelity approach shows a remarkable reduction in the projection error, which positively affects the overall reconstruction error. This is true for all three fields of interest. We can observe that the distance between the HF and multi-fidelity ROMs projection errors decreases as we add more HF snapshots, since the multi-fidelity projection error is almost constant for all fields. This means that the inexpensive LF addition of information can have a significant impact, especially in the HF data scarcity contexts, on the POD capabilities to represent unseen snapshots. This is consistent with the aim of multi-fidelity modeling, which is to reduce the overall computational cost. The slight monotonic increase in the p projection error might look counterintuitive, but it is due to the high similarity of LF and HF pressure solutions, which is less evident for the T, U fields. Indeed, fixing the POD energy threshold and adding redundant information in terms of HF snapshots introduces a slight degradation of the multi-fidelity POD basis projection capabilities. Increasing the POD energy threshold could solve this problem, but we did not observe significant differences in the resulting reconstruction errors.

Moreover, in Figure 5.32, the differences between the reconstruction and projection errors of the single-fidelity ROMs are almost constant, given any number of HF training snapshots. This difference represents the interpolation error quota introduced by the regression models. On the other hand, the multi-fidelity ROM shows a decreasing difference between the reconstruction and projection errors. Together with a constant projection error, this underlines an improvement in the interpola-

tion capabilities, introduced by the multi-fidelity regression model. Hence, a better regressor, together with a more effective encoding, allows the multi-fidelity ROM to increase the accuracy over the single-fidelity HF ROM.

Figure 5.33 compares the HF field values from a CFD solution with those predicted by the HF and multi-fidelity ROMs. In particular, a validation design has been used, corresponding to $Re = 318.0$, $A_1 = -0.0012$, $A_2 = 0.0143$, $\omega_z = 3.9$, and $\omega_\theta = 4.5$ parameters. Higher deviations from the diagonal highlight a poor agreement with the HF solution; therefore, the higher dispersion of the HF ROM suggests that the multi-fidelity ROM provides better predictions.

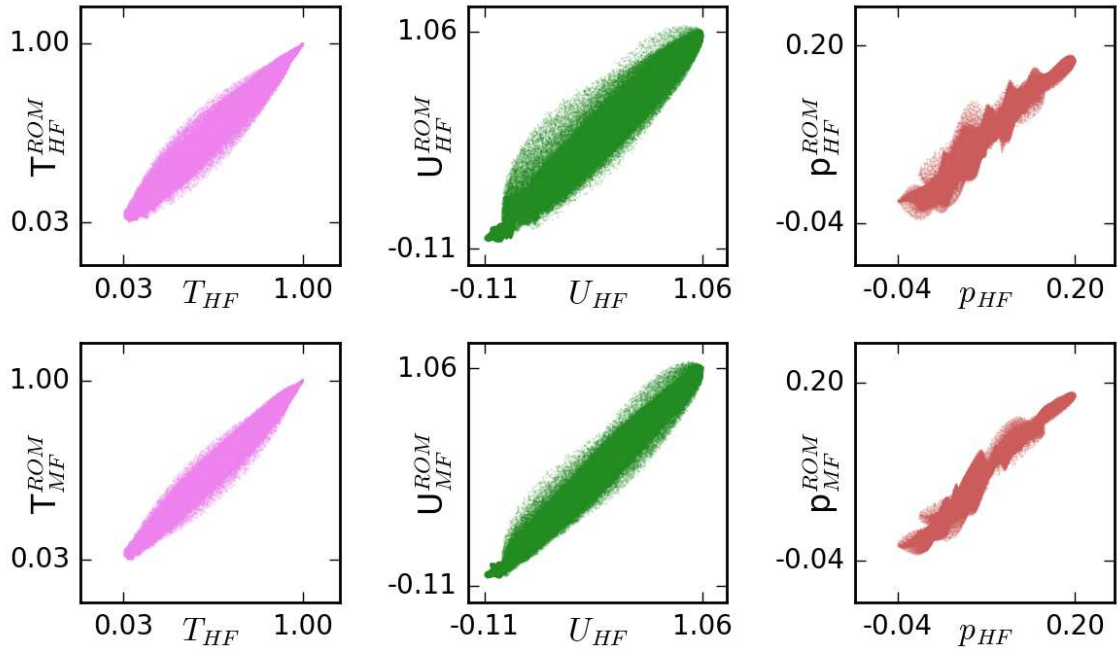
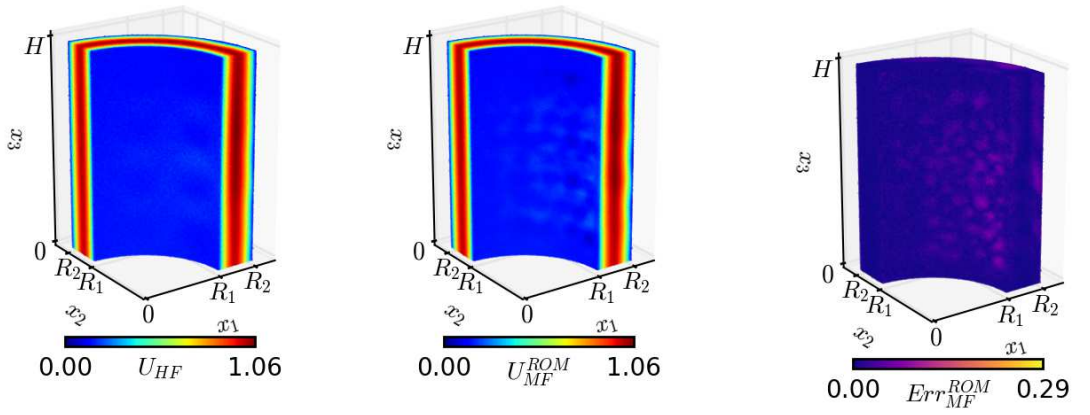


Figure 5.33: Scatter plots of T, U and p comparing the HF fields values with the HF ROM results (first row) and the multi-fidelity ROM results (second row) for an out-of-sample design. The ROMs have been trained with 140 HF snapshots, plus 230 LF snapshots for the multi-fidelity one.

Figure 5.34 shows an example of the multi-fidelity ROM prediction for the same validation design considered in Figure 5.33, with the absolute error field indicating that the fluid domain locations with the highest errors are localized near the walls. Here, the velocity field is represented on the reference domain.



(a) HF solution. (b) Multi-fidelity ROM. (c) Absolute error.

Figure 5.34: U magnitude HF solution (left), multi-fidelity ROM prediction trained with 140 HF snapshots (center) and corresponding absolute error (right). Input parameter $Re = 318.0$, $A_1 = -0.0012$, $A_2 = 0.0143$, $\omega_z = 3.9$ and $\omega_\theta = 4.5$.

Table 5.5 reports the computational time relative to the ROM training, excluding the offline cost of the snapshot generations. All computations were done with an Intel Xeon E5-2630 v3 CPU. The multi-fidelity ROM training is more demanding; however, the increase in computational cost is almost negligible when compared to the offline database generation. Table 5.6, on the other hand, reports the cost for the RBF-FD meshless simulations for both the LF and HF solvers. The snapshot mapping relies on the k-nearest neighbor, which is a very fast interpolation algorithm; thus, its computational time is negligible compared to the CFD solutions.

	HF ROM	MF ROM
Decomposition time $[s]$	4.6	88.8
Regression time per model $[s/\text{model}]$	1.5	27.8
Number of POD modes	16	140

Table 5.5: Single- and multi-fidelity velocity ROM's computational training wall time. 20 HF snapshots have been used for HF ROM, 20 HF plus 230 LF snapshots for the multi-fidelity ROM.

RBF-FD meshless simulation	LF	HF
Number of nodes	$150k$	$750k$
Wall-time	13 minutes	2 hours
Time-ratio	1:9	

Table 5.6: Details on a RBF-FD meshless simulation wall time cost.

The results shown in Figure 5.32, in particular the reconstruction error, can be presented in terms of computational time too. Figure 5.35 illustrates how the mean reconstruction error on the validation set changes with the offline computational cost. Since the multi-fidelity ROM always uses the same 230 LF snapshots, and having the cost of an HF simulation being 9 times the one of a single LF simulation, as in Table 5.6, a multi-fidelity ROM costs the equivalent of 25 HF simulations more than the single-fidelity equivalent. Nonetheless, the U, T break-even point in terms of computational cost is around 50 HF snapshots for the HF ROM and 20 HF snapshots plus 230 LF snapshots for the multi-fidelity ROM. Concerning p, the break-even point is slightly higher, with 80 HF snapshots for the single-fidelity ROM, and 50 HF snapshots plus 230 LF snapshots for the multi-fidelity ROM. These considerations are strictly dependent on the problem considered, the parameterization, and the relative cost of the LF solver.

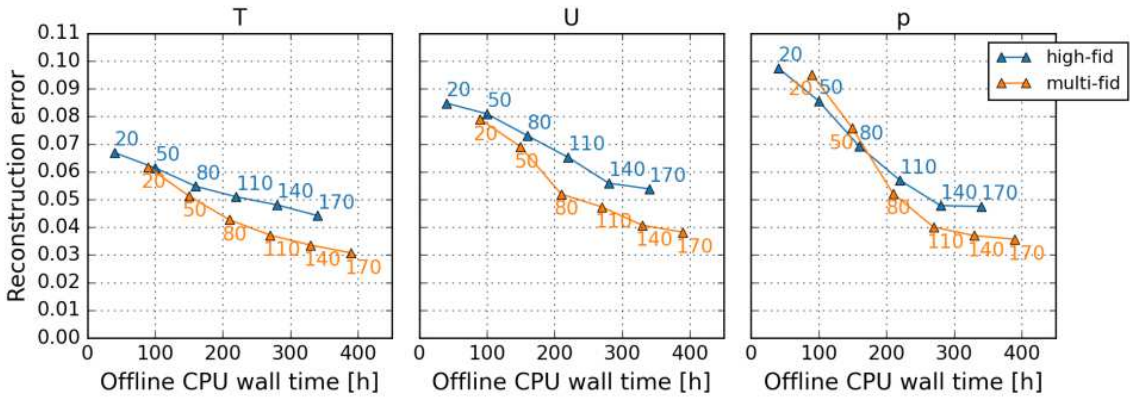


Figure 5.35: Mean reconstruction error vs offline computational time for the HF and multi-fidelity ROMs. The labels indicate the number of HF training snapshost. 230 LF snapshots were used for all multi-fidelity ROMs.

Discussion

In this experiment, the non-intrusive multi-fidelity ROM, combined with the RBF-FD meshless solver has been used to approximate a parametric 3D thermo-fluid dynamic problem, with both physical and geometrical parameters. We saw that the multi-fidelity ROM not only performs better in HF data scarcity conditions, with lower reconstruction error when compared to an equivalent single-fidelity HF ROM, but it is also effective from a computational time standpoint. The improved capabilities of the multi-fidelity ROM have to be found in the low projection error obtained by mixing LF solutions at the POD encoding stage. Moreover, the high correlation between LF and HF POD coefficients enables the multi-fidelity model, namely the NARGP, to improve the out-of-sample accuracy of the POD coefficients approximations. However, the necessity to tailor the ROMs to the specific test case when larger deformations occur proves the ROM approaches to be worth only for analyses that require many simulations. We observed that the meshless methodology alleviates some of the challenges related to the mesh generation for complex and irregular geometries, which can become time-consuming from a user perspective. This makes the coupling of meshless solvers and POD-based ROMs particularly powerful for automated design or optimization, with intricate and variable geometries.

5.3 Uncertainty Quantification for a Vertical Axis Wind Turbine

Darrieus Vertical Axis Wind Turbines (VAWTs) differ from conventional horizontal-axis turbines in that they maintain efficiency even in locations characterized by unpredictable wind patterns. Their space-saving geometry and omnidirectional wind capture make them particularly suitable for use in urban smart grids, offshore facilities, telecommunications stations, and industrial complexes.

Wind turbines experience various uncertainties that can affect both their design process and control. In particular, rotational velocity is a crucial parameter, often controlled to ensure a stable and optimal power output, preventing the turbine from over-speeding, while ensuring system compatibility [70, 71]. VAWT control is achieved in different ways, e.g., with blade pitch control, mechanical breaks, electric dump loads, or variable rotor resistance [72, 73, 74]. In this experiment we study the uncertainty introduced by the rotational velocity on the aerodynamic forces acting on the turbine blades of a Darrieus VAWT during an entire revolution. Specifically, we aim to assess the capabilities of an aerodynamic forces signal multi-fidelity ROM, parameterized on the rotational velocity, when performing the Uncertainty Quantification (UQ). To do so, the multi-fidelity ROM has been compared to the results of a Polynomial Chaos Expansion (PCE) and to an equivalent single-fidelity ROM. Indeed, UQ plays an important role whenever there is limited knowledge about the problem of interest [75]. In general, uncertainties have an inherently stochastic nature and can originate from multiple sources, such as measurements, physical modeling, geometrical tolerances, or computational approximations.

This study is based on our work for the *UNCECOMP2025* conference, whose proceedings can be found in [76].

5.3.1 Problem Description

We consider a 2D representation of a three-bladed Darrieus VAWT's flow, as in the schematic of Figure 5.38, in a rectangular domain with a length of 36 VAWT diameters and a width of 18 diameters. The radius between a blade and the center of rotation is equal to $R = 1.4$ m, and the blades of the VAWT are three NACA0015 airfoils with 420 mm chord, as in [77]. The wind speed is kept constant at $U = 10$ m/s, and the air properties are standard.

The problem is inherently unsteady, and with constant wind speed and rotational speed, it reaches a periodic flow, with the period identified by one complete revolution. We are interested on the forces acting on the VAWT blades during one turbine rotation. Due to the symmetry of the turbine, the forces acting on all three blades are the same, shifted by 120° between each other; therefore, we consider the forces acting on only one of the airfoils. These forces are time signals, and, for convenience, are reported as a function of the azimuthal angle position of the considered blade. In particular, we project the forces on the tangential and radial directions of the blade trajectory, identifying, respectively, the thrust force component F_T and the radial force component F_R. From an engineering standpoint, F_T is the sole contributor to the power generation, making it the most important force signal in this study. The turbine power P_T generated during one revolution is presented in Equation 5.21, highlighting its thrust force dependence.

$$P_T = \omega R N_{blades} \frac{1}{2\pi} \int_0^{2\pi} F_T(\theta) d\theta \,, \tag{5.21}$$

where N_{blades} is the number of blades, θ is the azimuthal angle indicating the blade position and $F_T(\theta)$ is the force signal during one revolution. F_R, on the other hand, remains fundamental for the design of the VAWT, especially for the structural considerations. Figure 5.36 outlines how the aerodynamic forces act on the considered airfoil.

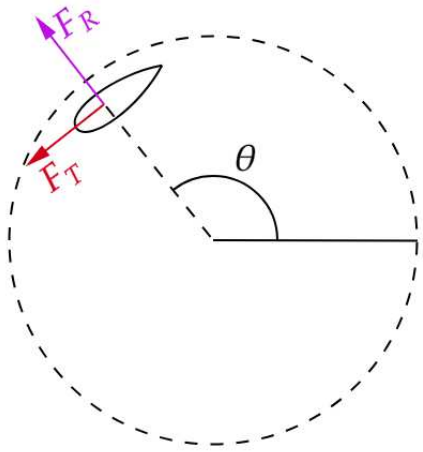


Figure 5.36: Thrust and radial forces F_T, F_R components acting on a VAWT blade, given the azimuthal angle position θ.

Parameterization

The only variable of the problem is the rotational velocity ω. The range of variation is chosen to have the tip-speed ratio (TSR) $\lambda \in [0.9, 1.32]$. The TSR definition is

$$\lambda = \frac{\omega R}{U}, \tag{5.22}$$

and, since U is constant, the TSR and the rotational velocity ω can be used interchangeably. Table 5.7 reports all the considered HF designs in terms of TSR. Part of the designs will be used to train the ROMs, both a single- and multi-fidelity one, and another part is used to perform PCEs, which will be used as the ground truths in the result section. The details on the PCE will be discussed in Section 5.3.2. LF designs are sampled evenly in the same TSR range, up to 23 LF snapshots.

Tip-speed ratio	Use-case
0.9	ROM
0.95	ROM
1.0	ROM
1.1	ROM
1.15	ROM
1.2	ROM
1.225	ROM
1.23953	PCE 4^{th} deg
1.24365	PCE 2^{nd} deg
1.24503	PCE 4^{th} deg
1.25	PCE $2^{nd}/4^{th}$ deg
1.25497	PCE 4^{th} deg
1.25635	PCE 2^{nd} deg
1.26047	PCE 4^{th} deg
1.32	ROM

Table 5.7: Input parameter locations used for the ROMs and the PCE.

Changes in the rotational velocity directly affect both the force signals and the turbine power, as Equation 5.21 shows. Figure 5.37 illustrates some examples of the force signals resulting from the CFD simulation at different values of the parameter λ. Both thrust and radial forces present great differences when changing the rotational

velocity. Most of the interest will be on the F_T component, due to its relationship with the VAWT power output. The one-dimensional signals will be used as snapshots for the ROM training.

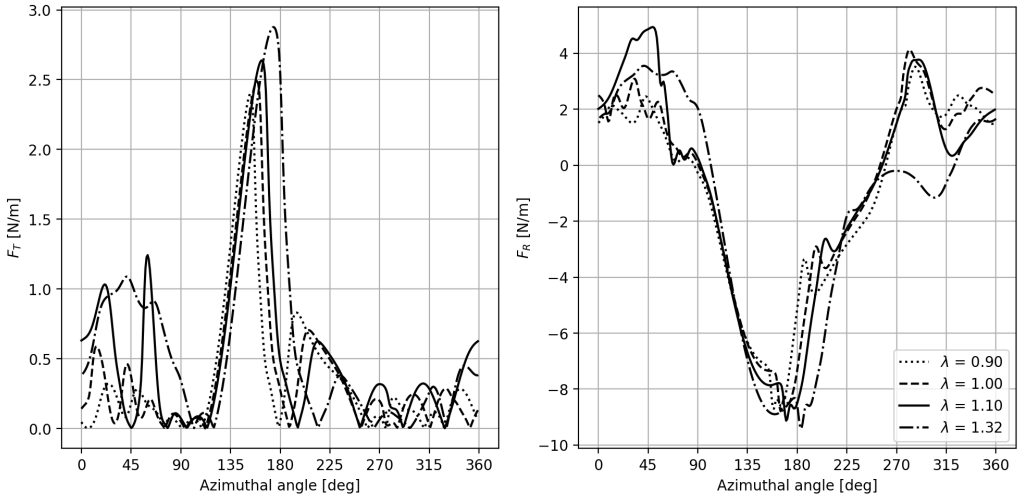


Figure 5.37: Thrust (left) and radial (right) force component signals during an entire revolution. Different curves correspond to different values of λ.

Simulation setup

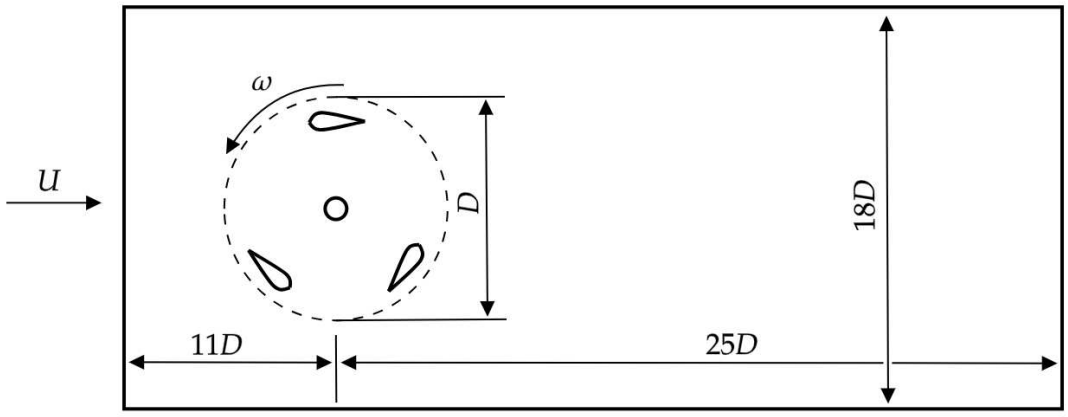


Figure 5.38: Schematics of the 2D CFD domain, with $D = 2.8$ m.

The computation has been carried out with ANSYS Fluent [78], solving for the transient RANS equations, alongside the $k\omega$-SST turbulence model. The blades' rotation was modeled with a cylindrical sliding mesh centered on the VAWT rotational axis. Figure 5.38 shows a schematic representation of the domain and its main dimensions. Best procedures and recommendations for the setup of a 2D VAWT CFD simulation can be found in the literature, e.g., in [79].

Since this experiment will require LF and HF simulations for the multi-fidelity ROM, a LF and a HF solver have been considered. The spatial discretization defines the fidelity level of the solver, having the higher fidelity simulations computed on $411k$ cell meshes, while the LF ones have coarser grids with $107k$ cells. The time discretization does not affect the fidelity of the simulation, having kept the same criterion for both solvers. The time step is set to ensure that at each temporal iteration there is a rotation of 1°, regardless of the fidelity level. Thus, the time step is a function of the rotational speed ω and, at the same time, ensures that all the snapshots are consistent for the POD encoding of the ROM for every parameter value. Figure 5.39 shows a detail of the LF and HF meshes around the airfoil, highlighting the differences between the two grid refinements. The computational cost ratio between HF and LF CFD simulations is $5:1$.

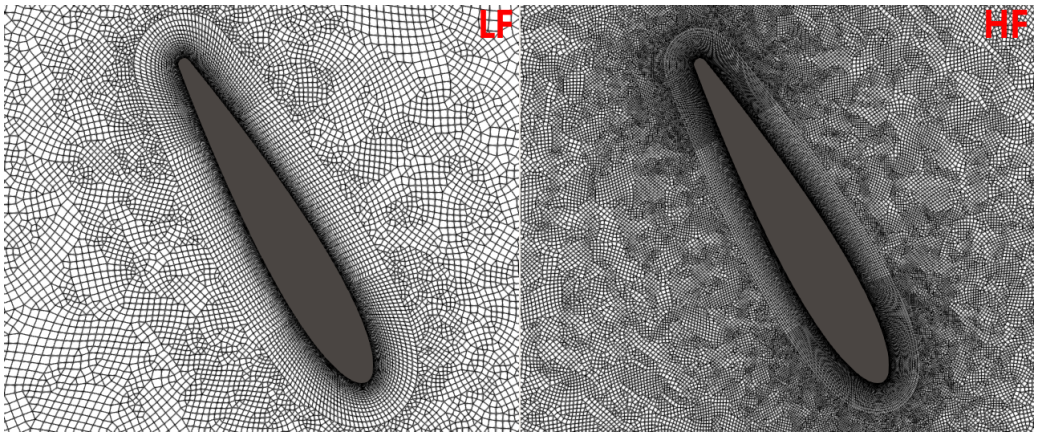


Figure 5.39: Close up of the computational mesh around one blade for a LF solver (left) and for an HF solver (right).

The HF results are reasonably consistent with the reference experimental measurements in [77]. Figure 5.40 shows the power coefficient C_P values of the HF CFD simulations compared with the experimental results. C_P represents the performance of the wind turbine, and it is defined as the ratio between the turbine power P_T,

defined in Equation 5.21, and the wind power P_W. The wind power is given by the wind crossing the turbine swept area $A_T = 2RH$, with H being the height swept by the VAWT, as in

$$P_W = \frac{1}{2}\rho U^3 A_T = \rho U^3 RH \,, \tag{5.23}$$

with ρ being the air density. The power coefficient C_P becomes

$$C_P = \frac{P_T}{P_W} = \frac{\omega R N_{blades} \frac{1}{2\pi} \int_0^{2\pi} F_T(\theta) d\theta}{\rho U^3 RH} \tag{5.24}$$

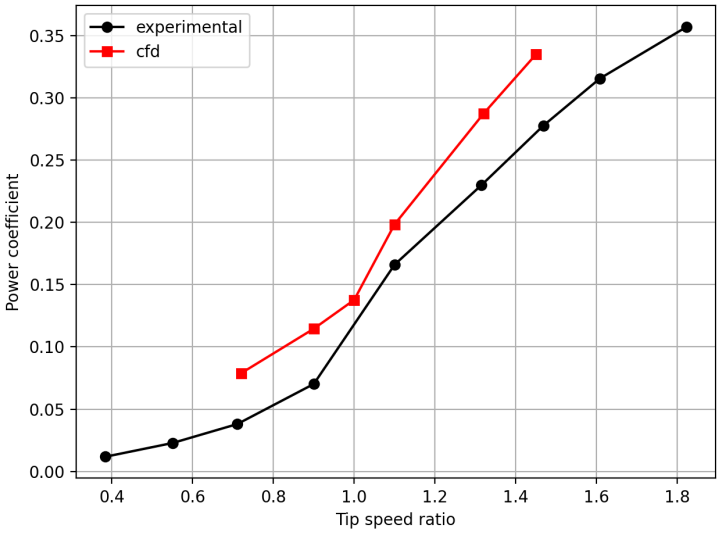


Figure 5.40: Power coefficient measurements from [77] vs current work HF CFD results.

The slight C_P overestimation in the CFD results can be attributed to the 2D simplification of the problem. The dissipative effects relative to inherently 3D phenomena can not be modeled in 2D simulations, such as the tip vortices or the fluid-structure interactions with the horizontal supports.

ROM setup

In this application, we compare the performance of a single- and a multi-fidelity ROM in a UQ context. The single-fidelity ROM has been trained with only HF

snapshots, while the multi-fidelity ROM has been trained with both LF and HF snapshots. Since the number of snapshots is low, there will be few POD modes. Consequently, a high POD energy $\varepsilon = 1$ was considered for both ROMs, and there was no truncation of the POD basis.

The POD coefficients in the single-fidelity ROM have been approximated with single-fidelity GPR models, with Matérn 3/2 covariance functions. The noise variance is constrained in $[0, 10^{-6}]$, having all POD coefficients normalized between 0 and 1. All other GP hyperparameters are kept unconstrained. The multi-fidelity regression model used in the multi-fidelity ROM formulation is, again, the NARGP model. Multi-fidelity approximations of the POD coefficients are utilized for every POD mode, and the NARGP HF kernel, as in Equation 3.50, has been obtained as the composition of three Matérn 3/2 kernels. Similarly to the single-fidelity ROM GPs, the noise variance has been constrained in $[0, 10^{-6}]$. Named l_ρ, l_δ the lengthscales corresponding to the $k_\rho(\mathrm{x}, \mathrm{x}'), k_\delta(\mathrm{x}, \mathrm{x}')$ kernels, these have been both constrained in the range $[0, 2]$, given that the input parameters have been normalized between 0 and 1, similarly to the POD coefficients. This has been done to avoid too large covariance functions that can hinder the optimization process, since it is based on a gradient minimization algorithm. The optimizer of choice for both ROMs is the L-BFGS with 10 random restarts to improve the robustness of the optimization.

5.3.2 Uncertainty Quantification

UQ studies the effect of the uncertainty in the input parameters, often modeled as stochastic variables, on some quantities of interest. In this experiment, we use both the PCE and a ROM-enabled Monte Carlo approach to perform the UQ. PCE will represent the ground truth that we aim to replicate with the ROM Monte Carlo. The idea of using ROMs as the building block for a Monte Carlo simulation comes from the high computational cost of Monte Carlo approaches and the quasi-real-time computation capabilities of non-intrusive ROMs, which enable the otherwise too expensive UQ technique.

In this problem, the only input variable is the rotational speed ω, which we suppose to be uncertain. Indeed, in VAWTs, or any low-speed rotor in general, the rotational speed can be measured with different sensors and techniques, such as Hall-effect sensors, tachometers, or image-processing methodologies [80, 81, 82]. For the purposes of this experiment, we assume an accuracy of ± 0.5 rpm for the VAWT hypothetical rotational speed measurement. Under the considered operating

conditions, the turbine rotates between 60 and 90 rpm, and a 0.5-1% error is close to what has been observed in literature. Consequently, the UQ is performed under the hypothesis that ω follows a Gaussian distribution with a standard deviation $\sigma = 0.25$ rpm, having 95% of the distribution lying within 1 rpm of the mean.

Polynomial Chaos Expansion

Orthogonal polynomials are used as a basis to approximate the functional form between the stochastic response output Y and each of its random inputs ξ. The chaos expansion Y takes the form

$$Y(\xi) \approx \sum_{j=0}^{\infty} c_j \varphi(\xi)\,, \tag{5.25}$$

with, only in this section, φ_j being the orthogonal polynomial of j-th order and c_j its corresponding chaos coefficient. From a practical standpoint, the infinite expansion needs to be truncated at a finite expansion order p, leading to

$$Y(\xi) \approx \sum_{j=0}^{P} c_j \varphi(\xi)\,, \tag{5.26}$$

where P is the number of basis. The PCE includes a complete basis of polynomials up to a fixed total-order specification, and, in that case, the total number of terms N_t is given by

$$N_t = 1 + P = 1 + \sum_{i=1}^{p} \frac{1}{i!} \prod_{j=0}^{i} (n+i) = \frac{(n+p)!}{n!p!}\,, \tag{5.27}$$

where n is the number of random variables. A stochastic collocation technique [83] can be used to determine all the c_j chaos coefficients, with $j = 0, ..., P$. Once these coefficients are known, a direct estimation of the system statistical moments can be done. The expected value $\mathbb{E}[Y(\xi)]$ is analytically derived as

$$\mathbb{E}[Y(\xi)] \approx \sum_{j=0}^{P} c_j \langle \varphi(\xi) \rangle = c_0 \tag{5.28}$$

and the response variance σ_Y^2 as

$$\sigma_Y^2 = \mathbb{E}[Y(\xi)^2] - \mathbb{E}[Y(\xi)]^2 \tag{5.29}$$

$$\approx \sum_{i=0}^{P}\sum_{j=0}^{P} c_i c_j \langle \varphi_i(\xi)\varphi_j(\xi)\rangle - c_0^2 \tag{5.30}$$

$$= \sum_{j=0}^{P} c_j^2 \langle \varphi_j^2(\xi)\rangle \,. \tag{5.31}$$

Depending on the probability distribution of the input random variable ξ different polynomials φ_j have to be considered. In this work, we consider only one input variable which is distributed according to a Gaussian distribution; therefore, Hermite polynomials have to be utilized.

Monte-Carlo simulations

Classical Monte Carlo methods are used in UQ for uncertainty propagation [84]. If the model's input parameter uncertainties are represented by a random variable ξ, a Monte Carlo method aims to approximate the model's stochastic response output Y. Indeed, Monte Carlo methods require sampling many times the random variables and computing the corresponding deterministic responses, to approximate the expected value $\mathbb{E}[Y(\xi)]$. Given M samples drawn from the random variable ξ, and the deterministic responses $y_1, ..., y_M$, $\mathbb{E}[Y(\xi)]$ is estimated with the average of the outputs, as in

$$\mathbb{E}[Y(\xi)] \approx \overline{y} = \frac{1}{M}\sum_{i=1}^{M} y_i \,. \tag{5.32}$$

Similarly, the variance σ_Y^2 can be estimated with the sample variance of the deterministic responses, as in

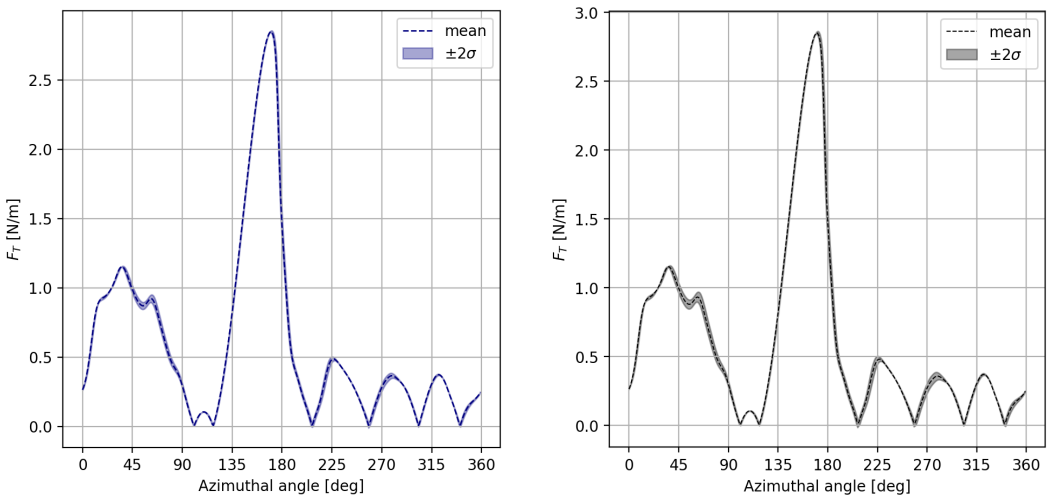
$$\sigma_Y^2 = \mathbb{E}[Y(\xi)^2] - \mathbb{E}[Y(\xi)]^2 \approx \frac{1}{M}\sum_{i=1}^{M}(y_i - \overline{y})^2 \,. \tag{5.33}$$

Due to the strong law of large numbers, $\overline{y}$ should converge to the actual expected value $\mathbb{E}[Y(\xi)]$ with enough samples M. However, the convergence rate is rather slow, being it $\mathcal{O}(\sqrt{M})$. Due to its poor efficiency, Monte Carlo methods are not suited for those problems which deterministic response is expensive. Thus, surrogate models, i.e., ROMs, can be inferred with close to no computational effort, and enable slow converging methods, such as Monte Carlo ones.

5.3.3 Results

We considered a design configuration with $\lambda = 1.25$, or alternatively $\omega = 85.3$ rpm, to perform the UQ study. As previously stated in Section 5.3.2, we assume that the random input variable is Gaussian, with a standard deviation of $\sigma = 0.25$ rpm, having $\omega \sim \mathcal{N}(85.3, 0.25^2)$. The UQ is performed for the entire force signal, with a particular focus on the thrust force component F_T.

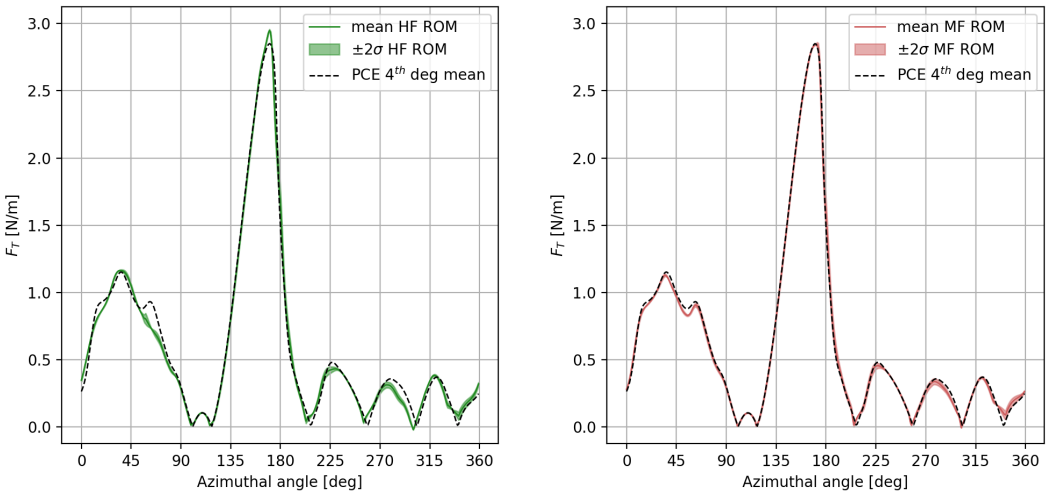
The ground truth for the UQ problem will be given by a 4th-degree PCE. Figure 5.41 shows the differences between a 2nd and a 4th degree PCE for F_T when $\omega \sim \mathcal{N}(85.3, 0.25^2)$. In general, the results are similar, but the confidence interval is generally wider with the 4th degree expansion.



(a) 2nd degree PCE. (b) 4th degree PCE.

Figure 5.41: First and second order moment estimates of F_T with PCE.

A single-fidelity HF ROM and a multi-fidelity ROM have been used to perform a Monte Carlo simulation and estimate the statistical moments for F_T. A total of 8 HF snapshots were used for both ROMs, plus 23 LF snapshots for the sole multi-fidelity model. 3000 samples have been drawn from $\omega \sim \mathcal{N}(85.3, 0.25^2)$ to perform the Monte Carlo simulations, and the deterministic responses have been computed with the ROMs.



(a) Single-fidelity HF ROM. (b) Multi-fidelity ROM.

Figure 5.42: First and second order moment Monte Carlo estimates of F_T with ROMs, compared to a 4^{th} degree PCE.

Figure 5.42 presents the results for the Monte Carlo ROM approach with both the single- and multi-fidelity ROMs. For the sake of readability, the confidence intervals of the PCE ground truth have been omitted; however, Figure 5.43 shows the differences in the standard deviation estimates between the Monte Carlo ROMs and the PCE.

To measure the error between the expected value estimates, we use a consistent error metric with the ROM reconstruction error e_{rec}, as in Equation 2.17. Specifically, we denote this error metric with e_μ, and it is obtained as

$$e_\mu = \frac{\|\mathbb{E}[\mathbf{u}_{pred}] - \mathbb{E}[\mathbf{u}_{true}]\|_2}{\|\mathbb{E}[\mathbf{u}_{true}]\|_2}\,, \tag{5.34}$$

where $\mathbf{u}$ represents the generic signal, i.e., the thrust force F_T. Having e_μ consistent with e_{rec}, enables a meaningful comparison between the error metrics. A mean e_{rec} has been computed in the given the 7 available ω design configurations used to perform the PCEs to have a local estimate of the ROMs reconstruction errors where the UQ study is performed.

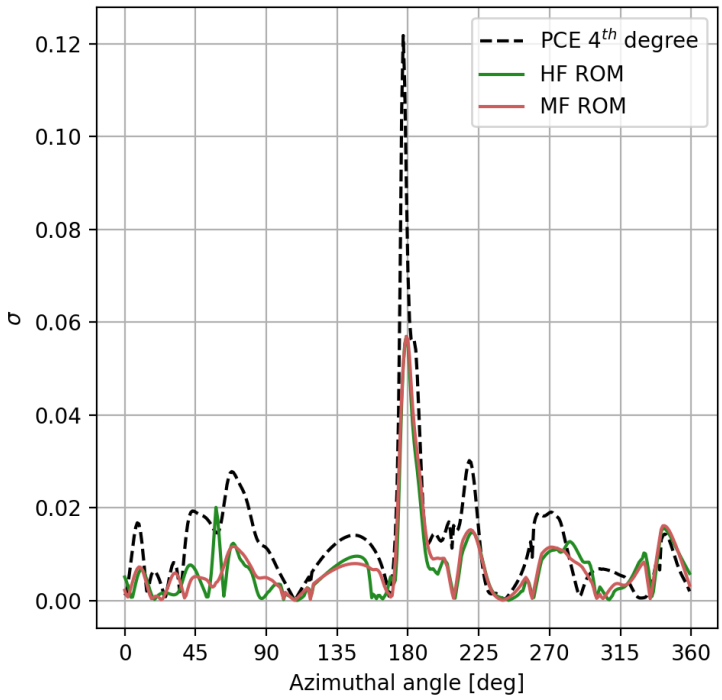


Figure 5.43: Comparison between the standard deviation σ estimates of the HF ROM, the multi-fidelity ROM and the ground truth PCE.

All the error results can be found in Table 5.8. For reference, the direct approximation of the same 7 design configurations with the LF CFD solver has an error of 19.7%, several times bigger than both ROMs.

	single-fidelity HF ROM	multi-fidelity ROM
e_μ	6.8%	2.9%
e_{rec}	6.7%	2.8%
UQ wall time [s]	0.0255	21.6

Table 5.8: Monte Carlo ROMs errors in the expected values estimates, ROMs local mean reconstruction errors, and ROM Monte Carlo online computational cost.

According to Equation 5.24, it is possible to integrate the F_T realizations obtained with the single- and multi-fidelity ROMs, each corresponding to a different ω, and compute the turbine power coefficient C_P. Similarly, the PCE mean estimate can

be used to compute the corresponding C_P, supposing that the rotational velocity corresponds to the mean one of $\omega \sim \mathcal{N}(85.3, 0.25^2)$. Figure 5.44 shows the mean C_P estimates obtained from the PCE and the Monte Carlo ROM F_T predictions. If the mean values are to be compared, the HF ROM has a relative error of 1.4% with the PCE result, and the multi-fidelity ROM has a 0.51% one.

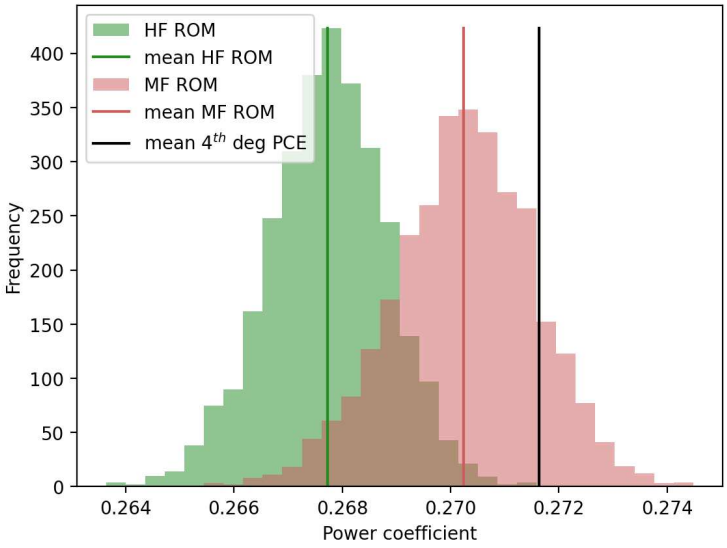
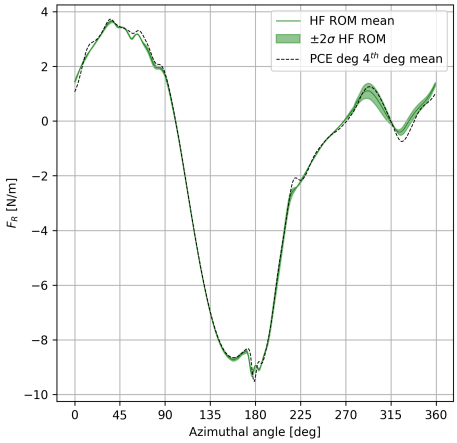
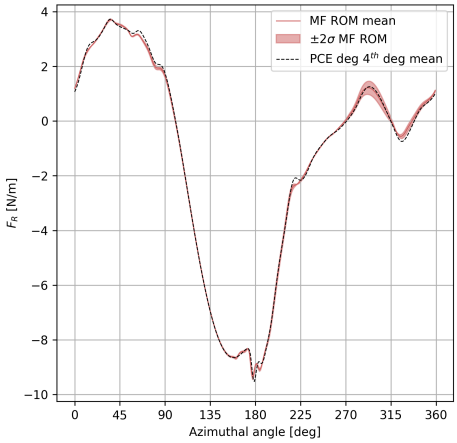


Figure 5.44: Comparison between the power coefficient values computed from the HF ROM realizations, the multi-fidelity ROM ones and the PCE mean estimate.

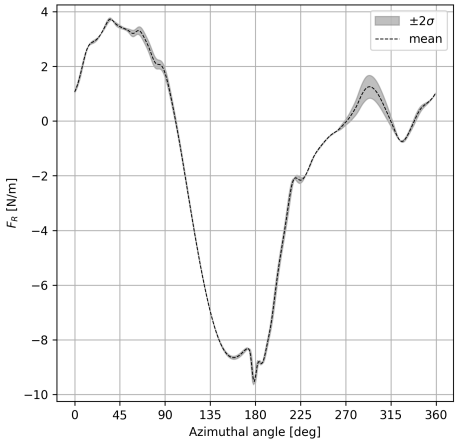
Similar analyses can be done for the radial component F_R, even though the uncertainty on the rotational velocity of F_R is less relevant from an engineering point of view. Figure 5.45 shows equivalent results of those obtained for the thrust component in Figures 5.41 - 5.43. The ROM errors on the expected value estimates are in line with those seen for F_T, being $e_\mu = 3.9\%$ for the HF ROM and $e_\mu = 2.4\%$ for the multi-fidelity ROM, respectively. A reasonable explanation for the slightly lower errors for F_R is that the force signals have less variability given the parameterization, as it can be seen in Figure 5.37.



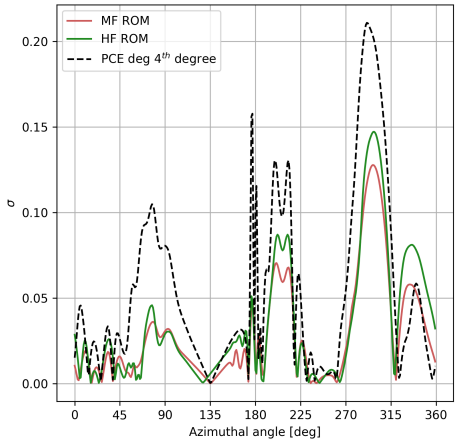
(a) HF ROM vs PCE 4th degree.



(b) multi-fidelity ROM vs PCE 4th degree.



(c) 4th degree PCE.



(d) Standard deviations.

Figure 5.45: UQ study results for the radial force component F_R.

Discussion

The Monte Carlo ROM approach showed reasonable performance in terms of expected value estimation of the force signals. The addition of inexpensive and otherwise useless LF information, which we recall it is several times less accurate than the HF one, improved significantly the multi-fidelity ROM accuracy and its UQ capabilities. Indeed, the mean estimation went from 6.8% error to 2.9% error. The same can not be said for the estimation of the second-order statistical moment, the standard deviation, which differs greatly from the PCE estimate. Even though the difference is quantitatively high, it is possible to notice in Figures 5.43 and 5.45(d) that the σ fields appear to be correlated, identifying correctly the high-uncertainty locations in the computational domain. Focusing on the C_P, as in Figure 5.44, it is possible to translate the UQ results obtained for the entire signal into a single quantity of interest. The multi-fidelity ROM, with no surprise given that the C_P depends on the integral of F_T, outperforms the single-fidelity ROM, lowering the relative error for the mean estimate almost 3 times. Keeping in mind that the ROM was trained with 8 HF snapshots in a very wide domain, i.e., $\lambda \in [0.9, 1.32]$, while the PCE requires less but more concentrated designs, i.e., 5 snapshots in $\lambda \in [1.23953, 1.26047]$, the Monte Carlo ROM approach showed promising results with 12 times less designs per unit length of the input domain space. This is true for the first statistical moment estimate, especially with the multi-fidelity formulation. However, the approach was not accurate enough for estimating the second-order statistical moment. Finally, we observed that the local ROM reconstruction errors were aligned with the error for the UQ estimates, as Table 5.8 illustrates. This suggests that the out-of-sample capabilities of both ROMs can be used as a possible empirical error estimator for e_μ.

5.4 Gyroid

This section presents an additional application of the proposed methodology, which is of a more exploratory nature compared to the main studies discussed earlier in Sections 5.1, 5.2 and 5.3. A thermo-fluid-dynamic study of the flow inside a gyroid structure is considered, alongside an original custom mapping strategy for the specific geometrical parameterization. This problem shares some similarities with both the pipe with wavy surfaces test case of Section 5.2, and with the pedagogical TPMS example in Section 2.4. Thus, providing the same level of detail as in the previous sections would be redundant. The analysis will focus primarily on the results, so as to consolidate our findings and give further perspective. When compared with the previous test cases, we will see that the multi-fidelity ROM advantages are less incisive, and this study offer useful insights into its potential and limitations.

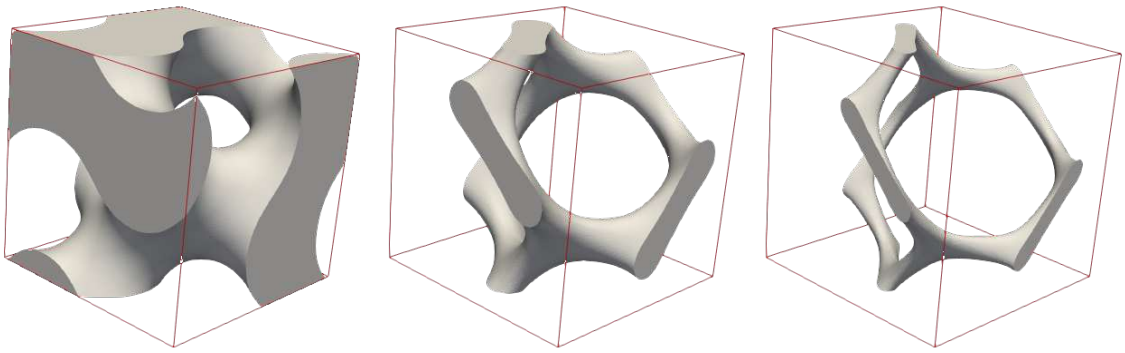


Figure 5.46: Fluid channels bulk representations for different wall thicknesses. From left to right, the wall thickness parameter increases, leaving thinner fluid channels.

Gyroids are TPMSs, similarly to the Schwarz-D surfaces presented in Section 2.4. These surfaces are promising candidates for heat exchanger applications due to their smooth curvature and high surface-to-volume ratio [85, 86]. In this application, we aim to approximate the velocity, pressure, and temperature fields inside a unit-cell gyroid given two design parameters: the wall thickness and the pressure drop applied to the gyroid. Since the thickness variation identifies geometries with different wall thicknesses, a mapping strategy will be presented. An example of the gyroid geometries obtained with different thicknesses is presented in Figure 5.46.

5.4.1 Problem description

The specific surface we consider is derived from the original gyroid formulation to have a meaningful thickness parameterization. The canonical gyroid surfaces are obtained as a level sets of

$$\begin{aligned} f(x,y,z) = &\sin\left(\frac{2\pi\omega_x}{L}x+\varphi_x\right)\cos\left(\frac{2\pi\omega_y}{L}y+\varphi_y\right)+ \\ &+\sin\left(\frac{2\pi\omega_y}{L}y+\varphi_y\right)\cos\left(\frac{2\pi\omega_z}{L}z+\varphi_z\right)+ \\ &+\sin\left(\frac{2\pi\omega_z}{L}z+\varphi_z\right)\cos\left(\frac{2\pi\omega_x}{L}x+\varphi_x\right)\,, \end{aligned} \tag{5.35}$$

where x, y, z are the Cartesian coordinates, $\omega_{x/y/z}$ are frequencies, $\varphi_{x/y/z}$ are phase shifts, and $L = 1$ cm is the periodic cell dimension. In this application, we consider level sets $\Phi(x, y, z) = th$, where th is the nondimensional wall thickness parameter, and Φ is defined as

$$\Phi(x,y,z) = \frac{f(x,y,z)}{\|\nabla f(x,y,z)\|_2}\,. \tag{5.36}$$

This formulation of the gyroid divided by the norm of its gradient is related to the thickness, meaning that higher values of Φ have larger wall thicknesses and, consequently, thinner fluid channels. Therefore, the surfaces we are going to consider are not strictly gyroids, but, for sake of simplicity, we refer to them as gyroids anyway.

Concerning the thermo-fluid-dynamic study, we refer to [85]. In summary, the simulations were computed with `buoyantPimpleFoam` from the open-source software `OpenFoam`, solving for the Navier-Stokes equations and the energy equation. The flow is supposed to be unsteady, laminar, and incompressible. However, for this application, we consider only design configurations with steady-state solutions. In general, all boundary conditions are periodic, and a pressure drop Δp is applied along the x-direction to force the flow.

The parameterization has been done according to the nondimensional thickness th and the pressure drop Δp. The DoE is a quasi-uniform grid, as in Figure 5.47. Here, we distinguish the steady and unsteady regions of the design space, the LF and HF design configurations eligible for training, and the reserved validation snapshots. The unsteadiness is present when the fluid channels become too large, due to the low

wall thickness th, and Δp is too high. Validation designs were manually selected from an already existing database from [85]. A total of 97 valid HF simulations and 263 LF simulations is considered. The main difference between the LF and HF solvers, similarly to the problems in Sections 5.1 and 5.3, is in the mesh discretization. A mesh for the HF solver has $\approx$ 500k cells, depending on the thickness parameter, while a mesh for the LF one has $\approx$ 65k cells.

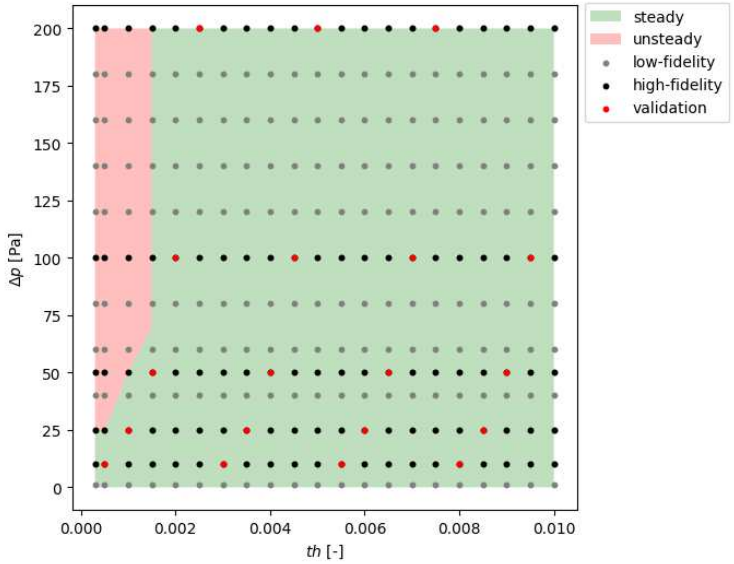


Figure 5.47: Design of Experiment for the gyroid test case. In green the steady state region of the design space, in red the unsteady part.

5.4.2 Mapping

As Figure 5.46 suggests, the geometry can change significantly with the th input parameter. As a consequence, it is impossible to work with a single computational mesh. Every simulation will have its own CFD tesselation, requiring a mapping of the solution to have coherent snapshots. To do so, we developed an original mapping strategy that transforms any point inside a gyroid in a smooth and coherent manner. This transformation allows us to move all internal points of a target gyroid, i.e., where the CFD solution is defined, inside the volume of a second gyroid, called the reference gyroid. By doing this, it is possible to interpolate the morphed CFD results on the reference mesh, similarly to Section 5.1.2 for the DrivAer problem. Consequently, all snapshots are defined on the same discretization and have the

same cardinality.

The first step is to define a reference gyroid. We choose a low-thickness gyroid as a reference for simplicity, but this does not limit the applicability of the method. Once this is done, we consider a point P inside the target gyroid volume, which will be a point belonging to the CFD tessellation. Successively, we compute the Φ values for the reference and target gyroids boundaries, obtaining Φ_r, Φ_t, respectively. Then, starting from the point P, we integrate a curve following the gradient $\nabla\Phi$ in the direction of the target gyroid surface. The $\nabla\Phi$ is always orthogonal to a gyroid function Φ level set, and, at a certain point, the integral trajectory will reach a point T belonging to the gyroid level set $\Phi = \Phi_t$, or, equivalently, the target gyroid surface. Figure 5.48 illustrates this process well.

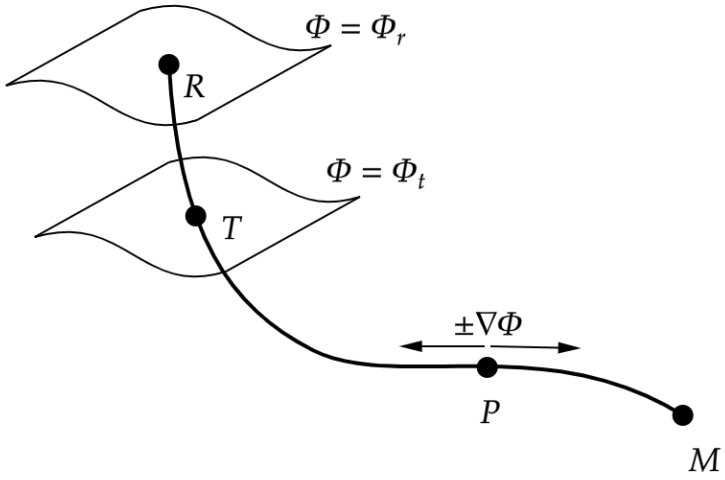


Figure 5.48: Integral trajectory of the gradient $\nabla\Phi$ from the target point P. T, R, M are the intersections with the target gyroid, the reference gyroid and the medial axis, respectively.

Since we suppose that the reference surface is associated with a low wall thickness, the reference gyroid surface will contain the target one. If we prolong the integral trajectory from T, it will eventually reach the reference gyroid, where $\Phi = \Phi_r$, at a point R. Given a function Φ, the reference and target gyroids, and a point P, the points T, R are uniquely defined. The integral trajectory can also be prolonged along the opposite gradient $\nabla\Phi$ direction from P, until reaching point M, as in Figure 5.48.

We define M as the point where the variation of Φ along the trajectory changes in sign, but, from a practical standpoint, it corresponds with the intersection between the integral trajectory and the medial axis. The integration of the trajectory is straightforward, and it is done explicitly as in

$$X_{k+1} = X_k \pm \delta \frac{\nabla\Phi(X_k)}{\|\nabla\Phi(X_k)\|}\,, \tag{5.37}$$

where X_k is the current point on the trajectory, X_{k+1} is the next point, and δ is the integration step. This is inexpensive to compute since both $\nabla\Phi(X_k)$ and $\|\nabla\Phi(X_k)\|$ can be evaluated analytically. Given the points P, T, R, M, and calling P_m the point where we want to move the original point P, we can enforce this proportionality relationship

$$\frac{MT}{MR} = \frac{MP}{MP_m}\,, \tag{5.38}$$

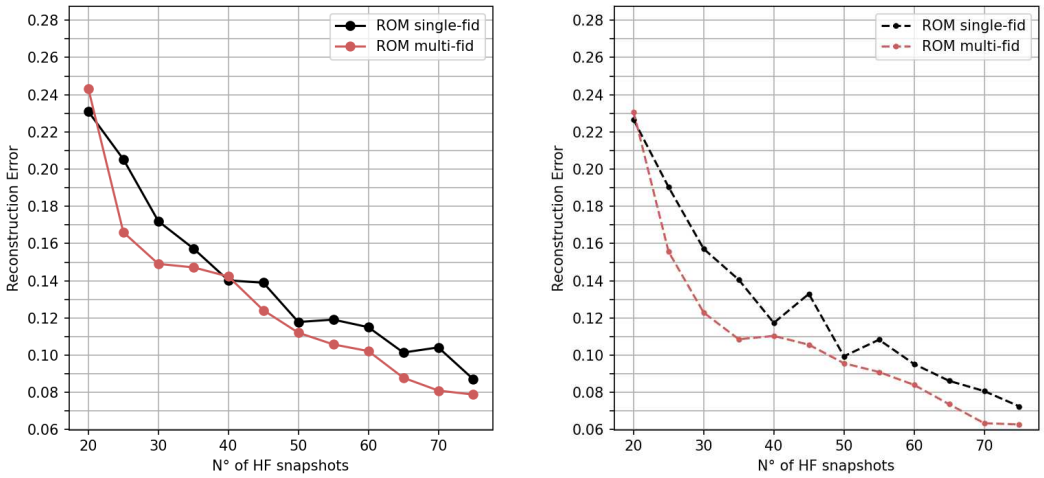
where MT, MR, MP, MP_m are all lengths computed on the integral trajectory. Therefore, from Equation 5.38 we know the position of P_m relative to the point M as

$$MP_m = MP\frac{MR}{MT}\,, \tag{5.39}$$

which can be used together with the integral trajectory to find the Cartesian coordinates of P_m. This needs to be performed for every node of the target computational mesh. In general, the morphing will not completely fill the reference volume with just the points from one CFD tessellation. To avoid this problem, we exploit the gyroid periodicity and apply the morphing to the repeated CFD tessellations as well. This ensure that in the reference gyroid there are no gaps prior the interpolation. To interpolate the fields from the morphed target CFD nodes to the reference nodes, a k-nearest interpolator is used. This technique is very inexpensive, fast, and tailored to point cloud problems; however, it is not the most accurate interpolation scheme. Prior to mapping the pressure component, we removed the linear contribution due to the applied pressure drop Δp, and we offset the pressure field so all fields have zero mean integral.

5.4.3 Results

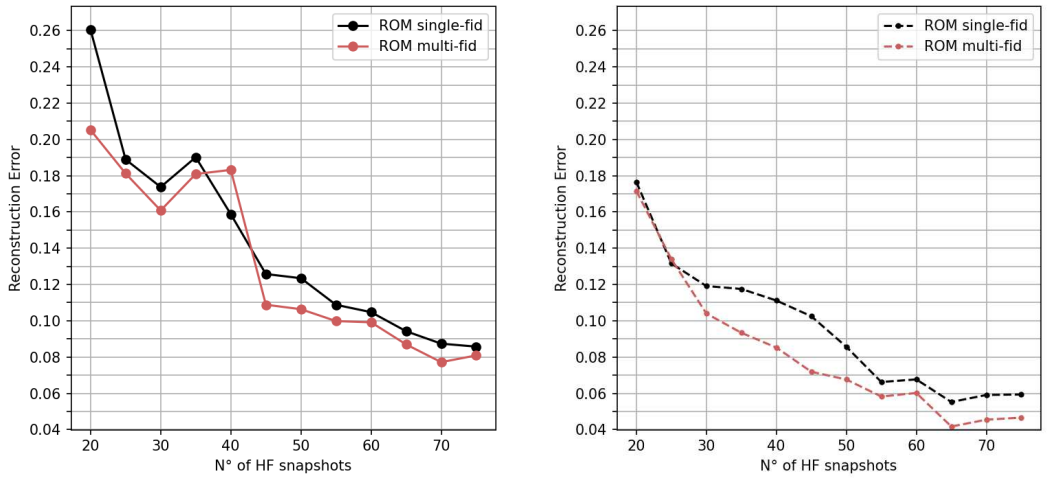
Similarly to what has been done for the test cases in Sections 5.1, 5.2 and 5.3, we compare a single-fidelity ROM and a multi-fidelity ROM predictions. Here we modeled the velocity, pressure, and temperature fields. For the training snapshots, we select only viable and steady designs, see Figure 5.47, and we consider nested databases with different cardinalities. Similar to what has been done for the DrivAer problem in Section 5.1, we defined a reasonable sequence of designs to have even coverage and filling of the input domain space as the training set size increases. Figures 5.49 - 5.51 show the results for the mean and median reconstruction error for the three fields of interest U, p and T. Finally, Figure 5.52 illustrates an example of multi-fidelity ROM predictions for the velocity, pressure, and temperature fields for the median velocity reconstruction error in the validation set.



(a) Mean reconstruction error.

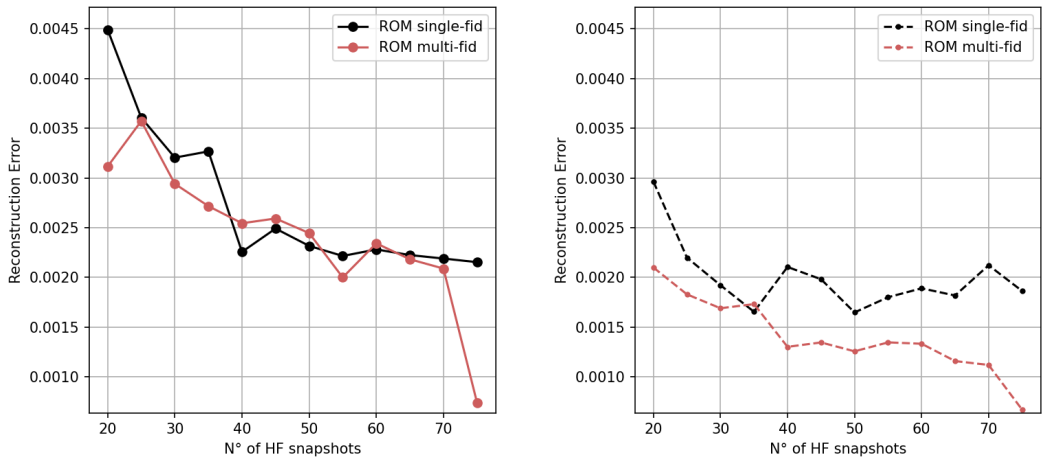
(b) Median reconstruction error.

Figure 5.49: Velocity single- and multi-fidelity reconstruction error statistics for different training set sizes.



(a) Mean reconstruction error. (b) Median reconstruction error.

Figure 5.50: Pressure single- and multi-fidelity reconstruction error statistics for different training set sizes.



(a) Mean reconstruction error. (b) Median reconstruction error.

Figure 5.51: Temperature single- and multi-fidelity reconstruction error statistics for different training set sizes.

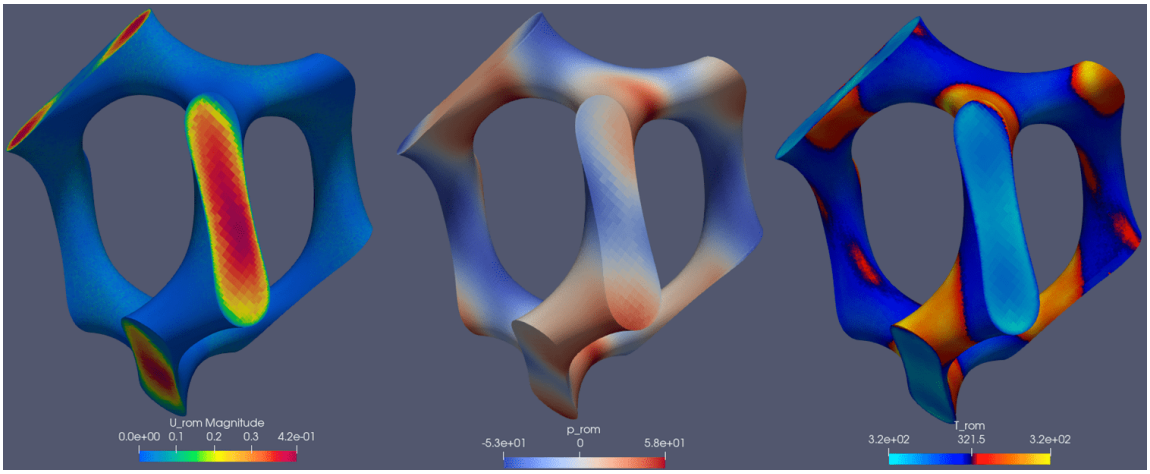


Figure 5.52: From left to right, an example of multi-fidelity ROM prediction of U, p and T. The pressure has the linear component from due to the imposed Δp removed. Median velocity reconstruction error design in the validation set, with $th = 0.005$ and $\Delta p = 200$ Pa.

Discussion

If compared to previous test cases, here the multi-fidelity ROM does not guarantee significant improvements over an equivalent single-fidelity ROM. This is especially true for the mean reconstruction error, while the median errors show a more favorable situation for the multi-fidelity ROM. Either way, the multi-fidelity ROM effectiveness is limited in this application. The higher mean reconstruction error, when compared to the median errors, is due to slighter tail-heavy validation error distributions for the multi-fidelity ROM. The multi-fidelity ROM shows an accuracy improvement when considering the median reconstruction error, especially with 25 or more HF training snapshots, depending on the considered field. The most interesting results are for the U, p fields, since T is already well approximated by the single-fidelity ROM. From a practical standpoint, the achieved accuracy gains by the multi-fidelity ROM for the temperature field do not have an important effect, if compared to the other fields.

In this application, it has been shown once again the importance of the parameterization when there are geometrical deformations. The proposed novel mapping approach allowed us to use POD-based ROMs. Otherwise, we would not be able to obtain coherent snapshots, unless using other geometrical registration techniques. This is not always straightforward; in this test case, there are no geometrical features

to exploit for the registration, and the dataset was not developed with ROMs in mind.

Conclusions

In this thesis, we have proposed a novel POD-based non-intrusive multi-fidelity ROM tailored to real-world problems. To complete the methodological part, we presented some industrial applications of our multi-fidelity ROM.

ROMs enable otherwise intractable many-query high-fidelity studies with limited resources. Nonetheless, the offline cost of ROM training can be too demanding, especially in industrial scenarios. Multi-fidelity modeling confronts this limitation for response surfaces, leveraging variable fidelity information sources. However, ROMs predict vector quantities instead of scalar ones. Moreover, ROMs exploit encoding strategies for dimensionality reduction, raising an ulterior barrier to the utilization of multi-fidelity techniques.

With the proposed multi-fidelity ROM, we want to address these problems. At the same time, we aim to understand how to utilize low-fidelity information to improve the accuracy of the ROM for unseen high-fidelity design configurations.

The proposed multi-fidelity ROM identifies a unique set of POD modes to encode the low-fidelity and high-fidelity snapshots, and it is achieved by mixing together snapshots from both fidelity levels. This has a positive effect on the accuracy, since the mixed fidelity POD basis functions better represent out-of-sample snapshots. In addition, the POD modes are representative of both low- and high-fidelity problems and provide a consistent latent representation of the variable fidelity projections. The consistent latent representation is then exploited by a nonlinear multi-fidelity regression model, the NARGP, improving further the ROM capabilities. Again, this positively affects the ROM accuracy, reducing the ROM reconstruction error.

The NARGP's nonlinearity plays a crucial role in the proposed multi-fidelity ROM. In a canonical multi-fidelity regression problem, the end user might have an intuition on the correlation between the low-fidelity and high-fidelity scalar quantity of interest, e.g., due to the differences in the numerical solvers, and can choose the best multi-fidelity regressor for the task. However, the same can not be said for multi-fidelity ROMs. Here, many multi-fidelity problems are built on the variable

fidelity latent snapshot representations, namely the POD coefficients, and their variable fidelity correlation is not known a priori. Consequently, NARGP's nonlinearity handles, with minimal bias, a wide variety of multi-fidelity correlations, while keeping a low computational cost thanks to the GP-based structure.

In this work, we observed that the multi-fidelity encoding requirement of coherent low- and high-fidelity snapshots is even more substantial in real-world problems. Indeed, industrial applications are often characterized by geometrical parameters, leading to non-coherent numerical solutions even for the same fidelity level. Therefore, mapping is a crucial aspect that needs to be opportunely taken into account for any ROM, possibly prior to the definition of the parametrized problem. In general, meaningful geometrical registrations or transformations are possible only in a limited set of cases. This information manipulation, together with the necessary interpolations on the reference discretization, has an effect on the snapshot's quality, but it is usually minimal or not as important for low-fidelity snapshots.

In this dissertation, we utilized the multi-fidelity ROM in four industrial applications, each with its own peculiarities. The first experiment saw an external aerodynamic automotive problem, namely the DrivAer, with a free-form deformation based geometrical parameterization. Here, we exploited the deformation fields resulting from the free-form deformation to map the non-coherent tessellations and trained the multi-fidelity ROM. Both the mapping approach and the ROM were effective. The multi-fidelity ROM had higher accuracy than the equivalent single-fidelity ROM, especially with small numbers of high-fidelity snapshots. This proved that the addition of inaccurate low-fidelity snapshots can have a positive effect on the ROM accuracy in out-of-sample design configurations.

A similar result was obtained in a completely different problem, when studying the internal flow inside an annular pipe with parametrized wavy surfaces. In this second test case, geometrical and operational parameters were mixed, and the flow was approximated with a meshless solver. We observed both a reduction of the reconstruction error thanks to the multi-fidelity ROM and that the combination of meshless solvers and multi-fidelity ROMs can lead to efficient surrogate solvers with minimal user intervention.

In a third experiment, we did an uncertainty quantification study on the forces acting on a vertical axis wind turbine under uncertain rotational velocity. The multi-fidelity ROM was used to approximate the force signal acting on the blades during one revolution, which is possible thanks to the non-intrusive formulation. Again, the addition of low-fidelity snapshots improved the accuracy of the ROM. This translated to better estimates of the expected values for the uncertainty quantification study

if compared to a single-fidelity ROM, and enabled a more accurate Monte Carlo approach thanks to the low online computational cost.

Finally, we presented the results for a second internal flow study, namely the gyroid. Here, we parameterized both the geometry and the operational conditions of a gyroid cell for a heat exchanger application. Although this problem is similar to the annular pipe with wavy surfaces, the multi-fidelity ROM showed modest improvements over the single-fidelity ROM. This highlights the dependence on the specific case and the choice of the low-fidelity data sources. Nonetheless, this test case offer an interesting insight on the custom mapping approach used to take into account the variable wall thickness of the gyroid structures.

All the presented industrial applications proved that our multi-fidelity approach to reduced order modeling can improve the surrogate model's accuracy, moving towards increasingly automated workflows. All tests were purposely done with the same high-fidelity snapshots to isolate the contribution of low-fidelity information addition when comparing to single-fidelity ROMs. However, this proves only that the multi-fidelity techniques are effective, not that there are efficiency gains. What was observed was that the multi-fidelity ROM is more data efficient depending on the problem and, most importantly, the cost of low-fidelity information. I.e., in the DrivAer problem, there is an advantage in using the multi-fidelity ROM, but it is not as decisive as in the annular pipe with wavy surfaces problem. The data efficiency dependency on the low-fidelity solver appears to be an important aspect to consider when using the proposed multi-fidelity ROM. A second limitation that emerged during this thesis work, which can also be transferred to single-fidelity POD-based ROMs, is the harmonization of non-coherent snapshots in industrial applications. Custom mapping strategies are necessary when the parameterization induces large geometrical deformations, hindering the adoption of ROMs in some industrial contexts. However, this problem can be mitigated by careful planning and a wise choice of the shape parameterization.

Future perspectives for this work should address the aforementioned limitations and improve the multi-fidelity modeling of the ROM. The proposed multi-fidelity ROM would benefit from an a priori methodology for the determination of a sensible low-fidelity solver to improve the multi-fidelity ROM's data efficiency. Moreover, improved correlation metrics can be beneficial to the multi-fidelity modeling, since Pearson's R correlation coefficient captures linear trends only between fidelities. Finally, industrial applications call for more flexible mapping techniques when dealing with shape deformations.

Appendix A

A.1 Simple multi-fidelity model: an application to hydrogen burners

In this appendix, we present an industrial application of our original multi-fidelity regression model, SimpleMF, presented in Section 3.2.2. The aim is to predict the flame length inside the combustion chamber in a premixed hydrogen burner, given different operational conditions. Figure A.1 shows a section of the hydrogen burner.

This study is based on our work for the *AIAA SciTech Forum 2025* conference, whose proceedings can be found in [76], where the author contributed mainly to the multi-fidelity modeling part.

A.1.1 Problem Description

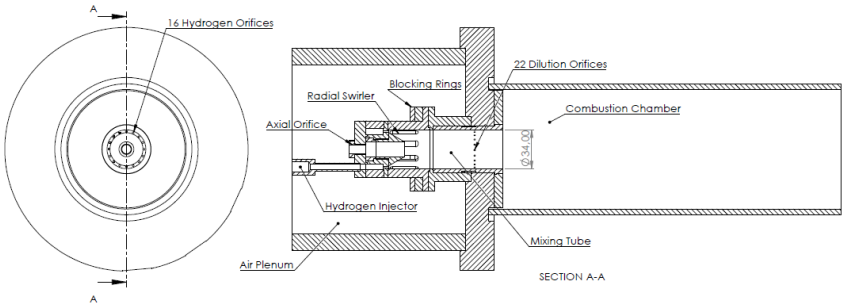


Figure A.1: Section of the premixed hydrogen burner with annotations, from [41].

The HF data for the multi-fidelity problem has been obtained with experimental measurements, while the LF data comes from 2D steady axisymmetric RANS simulations, following the results in [41]. All flame positions are adimensionalized with respect to the mixing tube diameter of 34 mm. Different design configurations are obtained by varying three input parameters: air mass flow, temperature, and equivalence ratio. Their definition can be found in [41]. The model has been trained with 3 to 20 HF data points and 53 LF data points. Ulterior training sets are obtained as subsets of 38 HF experimental results, nested within the LF designs. The other 15 HF data points have been reserved for testing. The LF and HF data are moderately linearly correlated, with a Pearson's R correlation coefficient of 0.77. Figure A.2 presents y_{LF} and y_{HF} values, the LF and HF quantities of interest, respectively, for the same design configurations. In Figure A.2, y^*_{HF} represents the underlying linear model of SimpleMF, as in Equation 3.65.

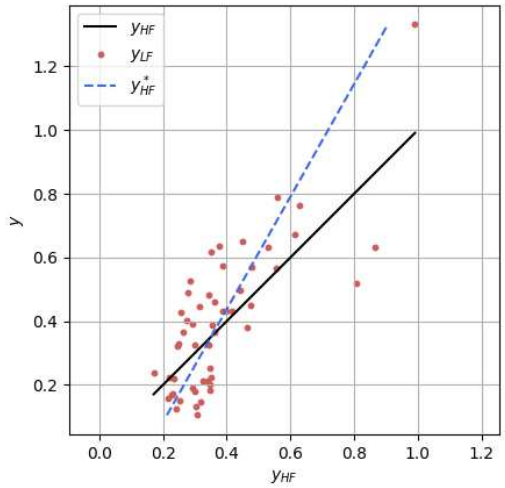


Figure A.2: Correlation plot for the hydrogen burner problem. y^*_{HF} represents the underlying linear model used by the SimpleMF model.

A.1.2 Results

We modeled y_{HF} with the SimpleMF model and compared the approximations with other three models: a single-fidelity Kriging, a multi-fidelity Cokriging model, and the NARGP. To have meaningful statistics, we extracted different training subsets from the available training data. To create the subsets, we used a similar sub-

sampling strategy from the DrivAer test case in Section 5.1.1. Given the HF data points, we select a random design and then add the next designs recursively so that for any design, the last one is at maximum distance from the previous ones. We repeated this process for each of the 38 available HF training data points, obtaining 38 different training subsets with up to 20 HF designs each. To measure the errors in the out-of-sample designs, the root mean squared error (RMSE) on the test set was used. Figure A.3 shows the mean RMSE given the 38 different subsets.

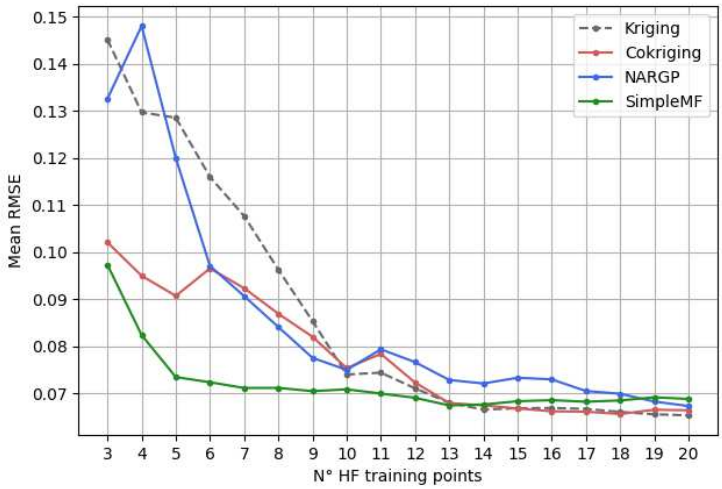


Figure A.3: Mean RMSE on the test set over 38 different training sets for different regression models.

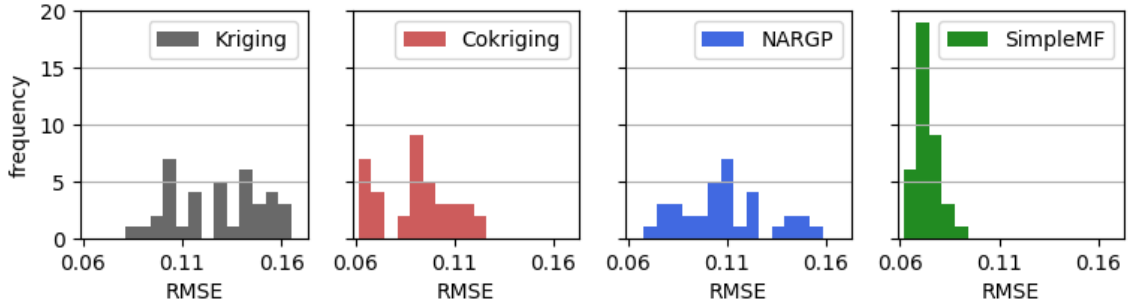


Figure A.4: RMSE distributions for different regression models, over 38 different training sets with 5 HF points.

The most evident differences between the models can be seen for 5 HF training points. Figure A.4 shows the RMSE distribution for all four models.

Figure A.3 shows that the SimpleMF model is more accurate when compared to the Kriging, Cokriging and NARGP models. This is especially true when dealing with HF data scarcity conditions. After 5 to 7 HF training points the SimpleMF performance plateaus, while the other models need 10 to 13 HF data points to close the gap in accuracy. After 13 HF training points, all the models are almost equivalent. The poor results of the other models are to be expected due to the high number of hyperparameters to optimize. The NARGP has the highest number of hyperparameters, followed by the Cokriging and the Kriging model. On the other hand, both the NARGP and the Cokriging are multi-fidelity model and can exploit LF information. This is reflected by the error trends: the NARGP behaves as the single-fidelity model, until it has enough HF information and performs similarly to the simpler Cokriging model. Finally, Figure A.4 shows that with few HF training points the SimpleMF model has both lower RMSE and a less dispersed error distribution, suggesting that the SimpleMF is more robust too. It goes without saying that the improvements of the SimpleMF model are possible only if there is close to linear correlation between fidelities. Moreover, using Kriging, Cokriging and NARGP models with so little training data in a three-dimensional input parameter space is not advisable. These models still need a certain amount of HF information to ensure a meaningful training process.

In this experiment, we found that the SimpleMF model was an effective choice for multi-fidelity regression in an industrial application. If the quasi-linear correlation requirements are satisfied, the SimpleMF outperforms other single- and multi-fidelity models in severe HF data scarcity conditions and reduces the effect of the input parameter space dimensionality.

A.2 DeepSDF mapping approach

In this appendix, we present a registration approach based on DeepSDF, a deep geometric encoding model [19]. The DeepSDF model is agnostic to the problem parameterization, and we aim to utilize it for registering two geometries. The main idea is to exploit a learned embedding space where geometries are encoded into compact latent vectors, and to interpolate within this space to construct meaningful trajectories in the Cartesian spaces from the points of one geometry to another.

A.2.1 DeepSDF model

The DeepSDF model is a NN auto-decoder [19] that maps a point in space and a geometrical latent code to the Signed Distance Function (SDF) of that point for a given geometry. Figure A.5 presents the NN architecture. The geometry is represented by a latent code $\mathbf{z} \in \mathbb{R}^l$, which is learned by the auto-decoder NN during training. From a practical standpoint, the training phase encodes the geometries. Once the DeepSDF model has been trained, it can be used for inference to predict the SDF field for unseen geometries, corresponding to different latent codes, and implicitly represent the geometries themselves.

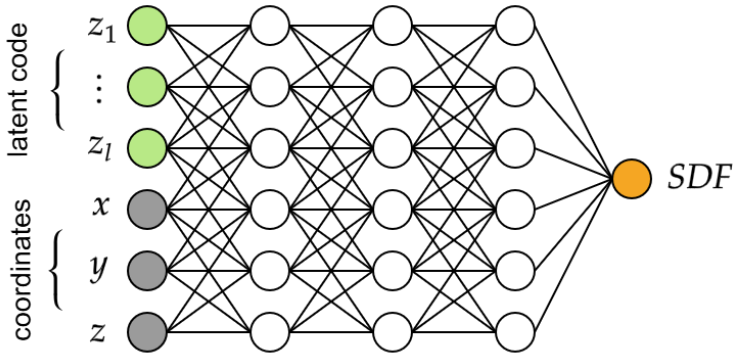


Figure A.5: DeepSDF model architecture schematics.

The DeepSDF training process requires a set of SDF fields X_i for different geometries, having

$$X_i = \{(\mathbf{x}_j, s_j) : s_j = SDF(\mathbf{x}_j) \, \forall \mathbf{x}_j \in A\} \quad \text{with} \quad i = 1, ..., N \,, \tag{A.1}$$

where $\mathbf{x}_j \in A \subset \mathbb{R}^3$ is a point in space from the sample set A of cardinality k, and $s_j \in \mathbb{R}$ is the SDF of the point $\mathbf{x}_j$ from the corresponding geometry. Unlike POD, which can be regarded as a linear encoder, the method does not impose strict constraints on the training data's cardinality or topology. Therefore, each SDF field X_i associated with a generic geometry can be sampled independently from any other geometry in the dataset.

The NN assumes that each geometry X_i can be represented with a latent code $\mathbf{z}_i \in \mathbb{R}^l$, and its prior probability distribution is a zero-mean multivariate-Gaussian with a spherical covariance $p(\mathbf{z}_i) \sim \mathcal{N}(0, \sigma^2 I)$. At training time, if we call $\mathcal{L}$ a loss function, i.e., the clamped L_1 norm, and $\hat{s}_j(\mathbf{z}_i, \mathbf{x}_j)$ the NN approximation of the SDF in a point $\mathbf{x}_j$ for the tentative $\mathbf{z}_i$ latent code, the joint log posterior over all training geometries is maximized w.r.t. the individual latent codes and the NN parameters, or equivalently

$$\min \sum_{i=1}^{N} \left(\sum_{j=1}^{k} \mathcal{L}\left(\hat{s}_j(\mathbf{z}_i, \mathbf{x}_j), s_j\right) + \frac{1}{\sigma^2} \|z_i\|_2^2 \right). \tag{A.2}$$

During training, both the NN weights and the latent codes are optimized together, and $\frac{1}{\sigma^2}\|z_i\|_2^2$ acts as a regularization term. Further details can be found in [19].

When the model is trained, given any latent code vector $\mathbf{z}$, it is possible to infer the SDF field at any point in space.

A.2.2 Mapping

We propose a geometrical registration technique for snapshots' mapping based on the DeepSDF model. We developed two registration approaches, namely the closest-point projection approach and the Lagrangian approach. The aim of both methodologies is to obtain a smooth registration that aligns a target geometry onto a reference one.

Given a dataset of geometries, their SDF representation, and a trained DeepSDF NN to encode them, we consider two shapes from the dataset: the target geometry and the reference geometry. For the sake of simplicity, we suppose that both geometries are part of the training dataset, ensuring control over the model accuracy in the SDF field prediction. With appropriate out-of-sample validation of the model, the target and reference geometries can also be unseen geometries. Each of the two

geometries is associated to a latent code, having $\mathbf{z}_{tar}, \mathbf{z}_{ref} \in \mathbb{R}^l$ for the target and reference shape, respectively.

We can define a trajectory Γ in the latent space from $\mathbf{z}_{tar}$ to $\mathbf{z}_{ref}$, i.e., the geodesic in $\mathbb{R}^l$, as in

$$\Gamma = \{\mathbf{z}_t = \mathbf{z}_{tar} + t(\mathbf{z}_{ref} - \mathbf{z}_{tar}) : t \in [0, 1]\} \ . \tag{A.3}$$

This trajectory smoothly interpolates between two latent codes, and, for each intermediate latent vector $\mathbf{z}_t \in \Gamma$, the corresponding SDF field can be predicted with the DeepSDF decoder. The 0 level set of each SDF field prediction corresponds to a different geometry, which can be considered intermediate between the target and the reference ones, and depends on the trained DeepSDF model. The core idea is to exploit these intermediate 0 level sets to perform the registration. Small variations in $\mathbf{z}_t \in \Gamma$ produce similar geometries, and if the topology of the training shapes remains unchanged, we empirically observed that the intermediate geometries are topologically similar. To take advantage of these intermediate geometries, we developed two approaches, detailed in Sections A.2.2 and A.2.2. Experimental results indicate that both exhibit comparable behavior, characterized by similar strengths and limitations.

Closest-point projection approach

With the closest-point projection the aim is to define a trajectory of points, starting from a target geometry discretization, ending in the reference geometry. For every intermediate latent code $\mathbf{z}_t \in \Gamma$, we predict the SDF field and extract the 0 level set $\mathcal{C}_t$, as in

$$\mathcal{C}_t = \{\mathbf{x} \in \mathbb{R}^3 : \hat{s}(\mathbf{z}_t, \mathbf{x}) = 0\} \, . \tag{A.4}$$

Let us consider a point $P_0 \in \mathcal{C}_\prime$ from the target discretization, where $\mathcal{C}_\prime$ is the target geometry since $\mathbf{z}_0 = \mathbf{z}_{tar}$. From a practical standpoint, the $\mathcal{C}_t$ level sets are finite in number and are obtained by sampling uniformly $t \in [0, 1]$, having $t = \{t_0, ..., t_N\}$, with $t_0 = 0$ and $t_N = 1$. Following $\mathbf{z}_t \in \Gamma$, the next latent code in the progression is $\mathbf{z}_{t_1}$ and the corresponding 0 level set is $\mathcal{C}_{t_1}$. We can project P_0 onto $\mathcal{C}_{t_1}$, identifying $P_{t_1} \in \mathcal{C}_{t_1}$. This process can be repeated $\forall \mathbf{z}_t \in \Gamma$, obtaining the Cartesian space's trajectory γ for the target point P_0

$$\gamma = \{P_t : t \in \{t_0, ..., t_N\}\} \ , \tag{A.5}$$

with the last point $P_{t_N}(= P_1)$ belonging to the reference geometry. Figure A.6 shows a schematic of the registration process. If this process is repeated for all points in the target discretization, the registration is complete.

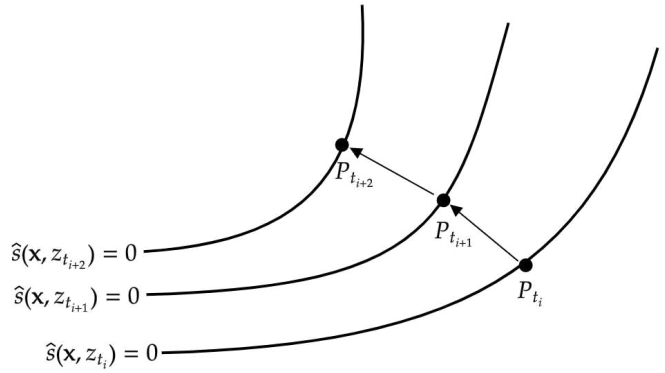


Figure A.6: Closest-point projection registration approach 2D schematics for a generic point P_{t_i} belonging to the corresponding 0 level set.

Successively, the the field of interest for the ROM can be interpolated from the registered target geometry to the reference discretization, allowing us to have coherent snapshots for the POD.

Lagrangian approach

In the Lagrangian approach, the trajectory γ is defined differently, even though it shares some similarities with the closest-point projection approach. Under the same assumptions, we consider a point of the target geometry $P_0 \in \mathcal{C}_0$. In P_0, we evaluate the gradient of the DeepSDF model $\nabla \hat{s}$ w.r.t. x, y, z given the current latent code $\mathbf{z}_0 = \mathbf{z}_{tar}$. Since this gradient is not known analytically, we compute an approximation with finite differences, having for the generic point P_t

$$\nabla \hat{s}(\mathbf{z}_t)|_{P_t} \approx \left[\frac{\hat{s}(\mathbf{z}_t, P_t + \delta e_i) - \hat{s}(\mathbf{z}_t, P_t)}{\delta} \right]_i , \tag{A.6}$$

with δ being the perturbation step for the finite differences, and e_i being the unit vector along the i-th direction, i.e., x, y, z. Then we apply a Newton-Raphson correction to find the next point on the following 0 level set. If we consider the generic point P_{t_i} during the registration, the next point $P_{t_{i+1}}$ is found iterating

$$P_{t_{i+1}} = P_{t_i} - \hat{s}(\mathbf{z}_{t_i}, P_i) \frac{\nabla \hat{s}(\mathbf{z}_{t_i}, P_i)}{\|\nabla \hat{s}(\mathbf{z}_{t_i}, P_i)\|^2}\,, \tag{A.7}$$

with a fixed $\mathbf{z} = \mathbf{z}_{t_i}$, until we reach the next SDF 0 level set $\mathcal{C}_{t_{i+1}}$, that corresponds to $\mathbf{z} = \mathbf{z}_{t_{i+1}}$, and finding the actual $P_{t_{i+1}}$ point. Figure A.7 illustrates schematically how the method works for a generic point P_{t_i}. In general, the Newton-Raphson method will not find $P_{t_{i+1}}$ in one iteration, so Equation A.7 will identify intermediate points, represented as Q_i in Figure A.7. This process needs to be repeated for every discrete value of $\mathbf{z}_t \in \{\mathbf{z}_{t_0}, ..., \mathbf{z}_{t_N}\}$ to obtain the full trajectory γ from a starting target point P_0.

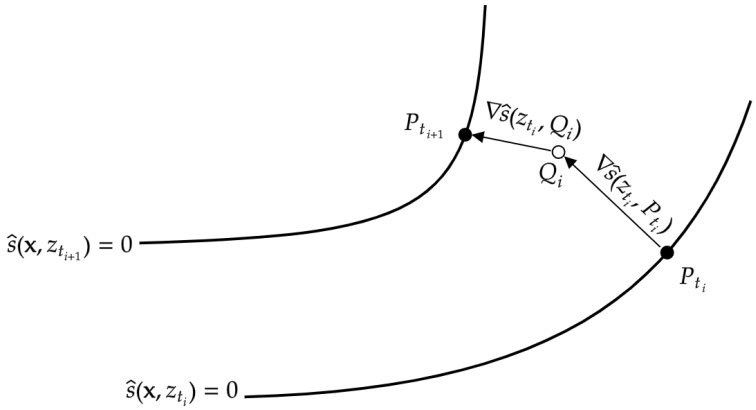


Figure A.7: Lagrangian registration approach 2D schematics for transforming a point P_{t_i} into $P_{t_{i+1}}$. Here, Q_i is used to represent an intermediate point during Newton-Raphson iterations.

Keeping in mind that the gradient $\nabla \hat{s}(\mathbf{z}_t, \mathbf{x})$ is always orthogonal to the 0 level set for $\hat{s}(\mathbf{z}_t, \mathbf{x})$, if the discretization in the latent space is fine enough, this method is very close to the closest-point projection approach. The main difference is that the gradient and the Newton-Raphson iterations are more computationally demanding than the closest-point projection.

A.2.3 Dataset

The proposed mapping methodology has been tested on a 2D pedagogical example. A dataset of closed planar curves and their surrounding SDF fields has been collected for this purpose. We parametrize the curves as circles with periodic radial perturbations,

as in

$$\begin{cases} x = r(\theta; A, \omega, r_0) \cdot \cos\theta \\ y = r(\theta; A, \omega, r_0) \cdot \sin\theta \end{cases}, \tag{A.8}$$

where

$$r(\theta; A, f, r_0) = r_0 + A\cos(\omega\theta), \tag{A.9}$$

with $\theta \in [0, 2\pi]$ being the phase, $r_0 \in [0.5, 0.6]$ is a radius, $A \in [-0.15, 0.15]$ is the radial perturbation's intensity, and $\omega \in [2\pi, 6\pi]$ is the frequency of the radial perturbation.

We generated 14 2D curves for this study, of which one is always considered as the reference geometry, and the remaining ones can be selected as target geometries. Figure A.8 shows all 14 2D curves.

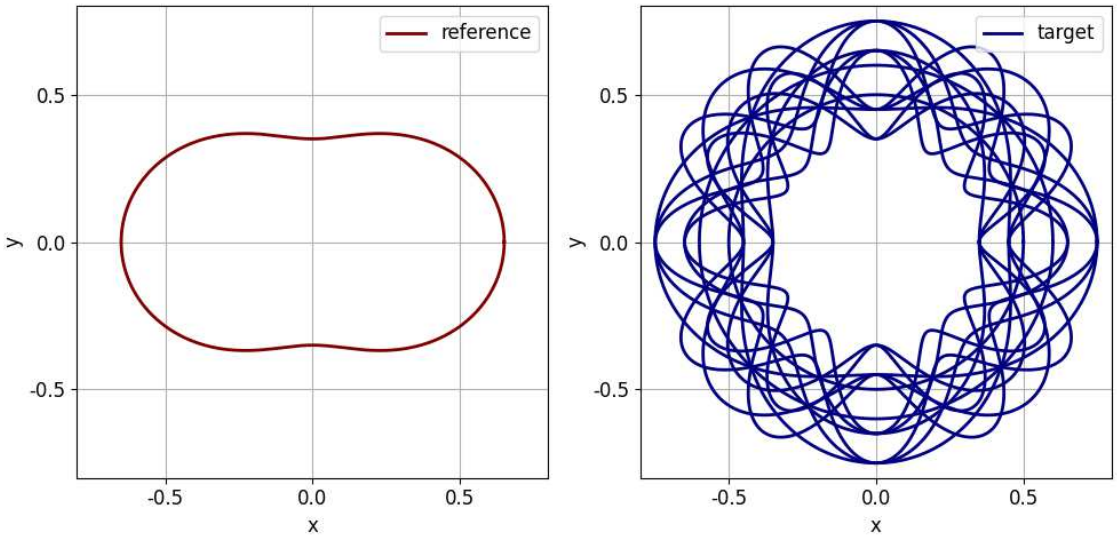


Figure A.8: All curves in the dataset. Reference geometry (left) and target geometries (right).

With these curves, we compute the SDF field in the unit circle, obtaining fields analogous to Figure A.9. All geometries have been utilized to train the DeepSDF NN model.

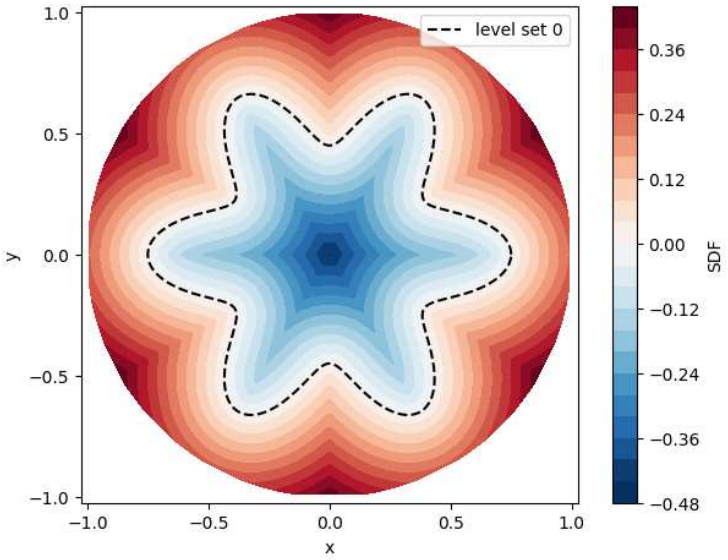


Figure A.9: SDF field example for one of the training geometries.

A.2.4 Results

In this section the aim is to provide a qualitative presentation of this method's capabilities. The study is preliminary and wants to highlight the limitations of the current mapping strategy and what can be improved. All the following results have been obtained with the closest-point projection approach, since the performance of the Lagrangian approach is very close, as discussed in Section A.2.2. The geodesic in the latent space has been discretized with 1000 latent vectors, equally spaced in the Euclidean norm of $\mathbb{R}^l$.

Figures A.10 and A.11 illustrate how the registration happens. Here, with different color it is possible to appreciate how the points trajectories evolve when projecting them on successive 0 level sets. These figures also give an intuition of how smoothly the DeepSDF model predictions of the SDF change, since all points stays on different 0 level sets. In particular, the individual points on the intermediate steps of the registrations always belong to a SDF 0 level set for the current latent vector $\mathbf{z}_t$. This enables us to see how the target geometry evolves when transforming in the reference geometry following the latent space geodesic. Moreover, the distribution of the points at each intermediate step fills evenly the current 0 level set, preventing vacancies in the coverage.

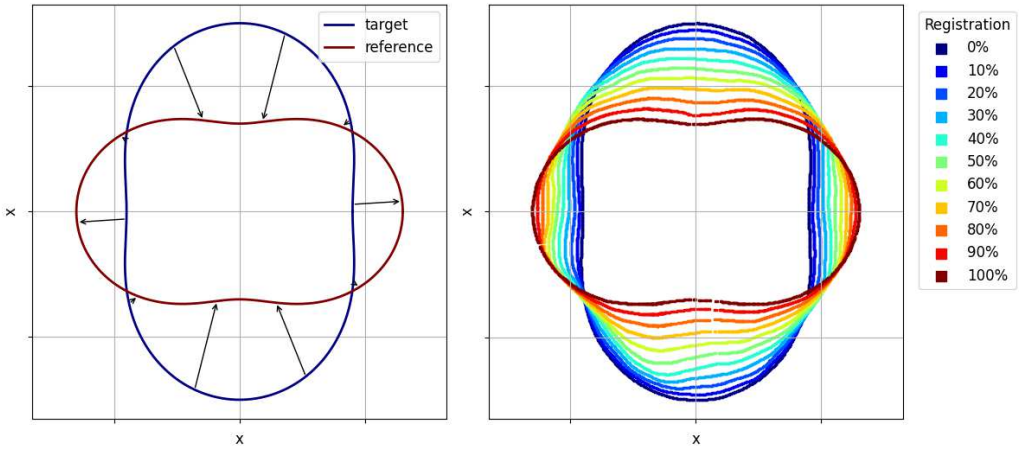


Figure A.10: Reference and target geometries (left). Progression of the target points at different stages of the registration (right).

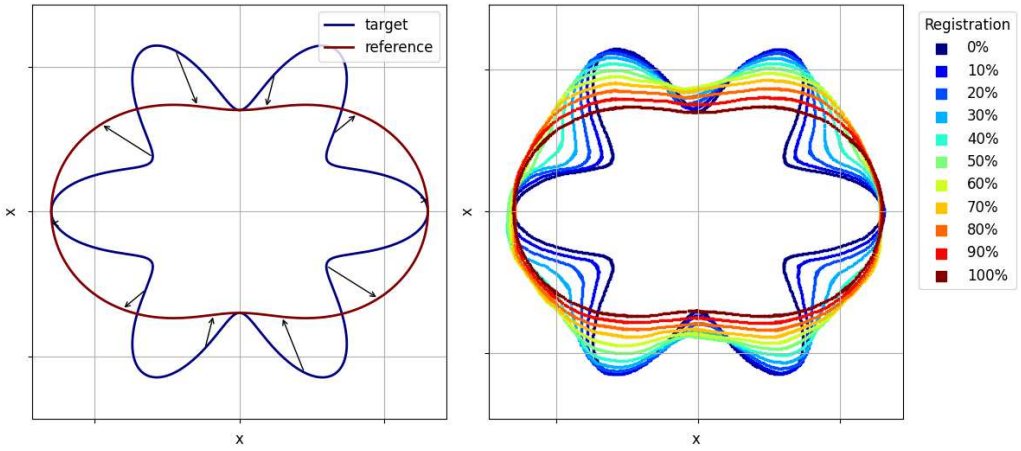


Figure A.11: Reference and target geometries (left). Progression of the target points mapping at different stages of the registration (right).

Figures A.12 and A.13 show the entire trajectories from the single target points to the reference geometry.

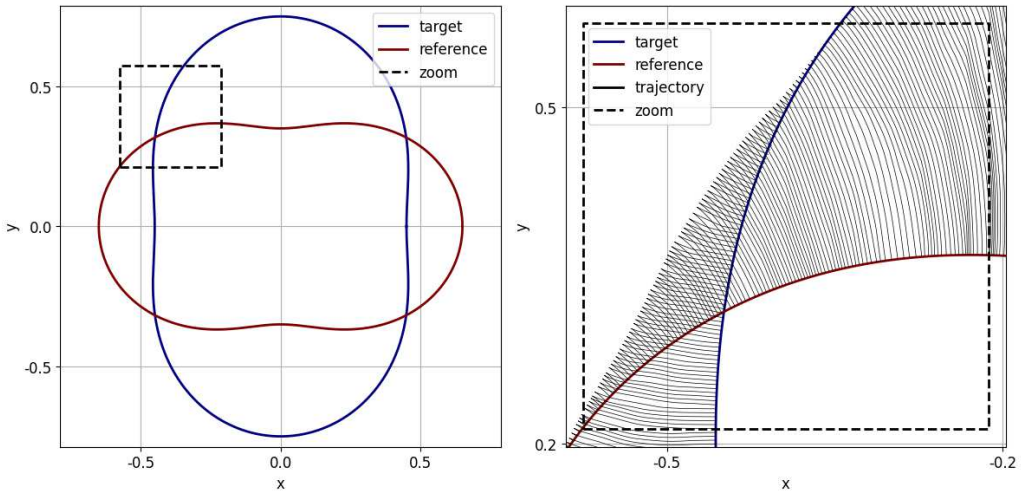


Figure A.12: Reference and target geometries (left). Local zoom and representation of the single points trajectories during the registration (right).

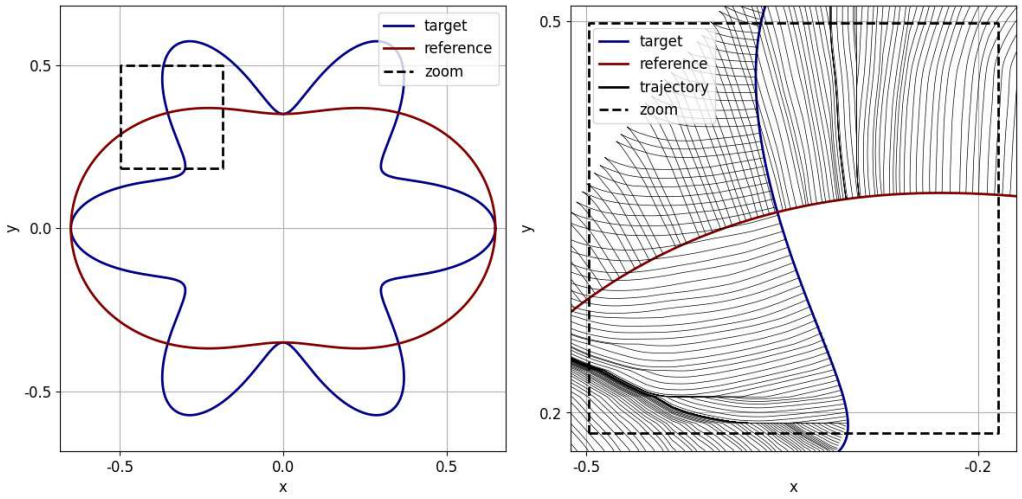


Figure A.13: Reference and target geometries (left). Local zoom and representation of the single points trajectories during the registration (right).

The registration happens smoothly, especially in Figure A.12. However, in Figure A.13, we can observe some darker zones where the trajectories collapse on the same path. This is not desirable since different target points are mapped on the same reference location. In this example, this effect is minimal and localized, and depending

on the application can be acceptable.

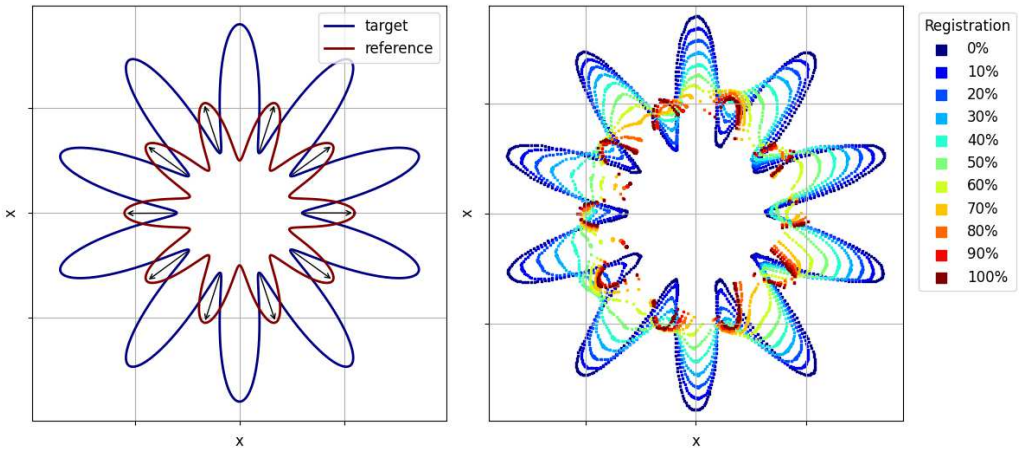


Figure A.14: Reference and target geometries (left). Progression of the target points mapping at different stages of the registration (right). Problematic test case.

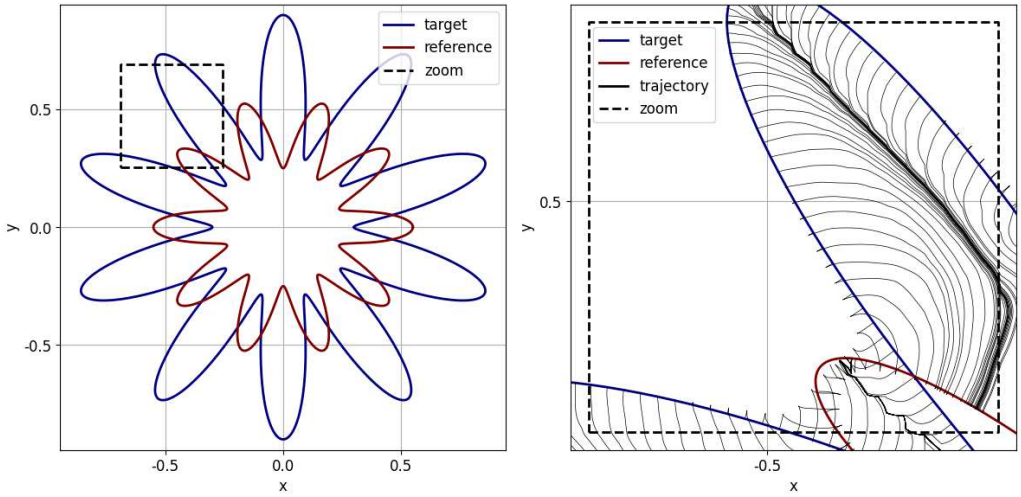


Figure A.15: Reference and target geometries (left). Local zoom and representation of the single points trajectories during the registration (right). Problematic test case.

Nonetheless, we observed that this behavior is amplified by strong differences in convexity between the geometries, or when there are high curvature features. To explore this behavior, we created an affine database with more complexity in the

geometries and tested the approach. We registered a multi-lobe curve onto a different multi-lobe geometry, with higher local curvatures values. Figures A.14 and A.15 show the results for this purposely difficult test case. Here, the registration fails completely and this can be observed in Figure A.14 on the right, where the 100% registration step has only a partial coverage of the reference curve. Figure A.15 clearly illustrates how the trajectories collapse into single dense streams of points, leaving part of the reference geometry untouched by the registration.

The possibility to register different geometries and map all the fields of interest on a common reference discretization is crucial for ROMs. The proposed method appears to be capable to handle simple geometries, remaining agnostic from the parameterization. However, this model currently shows some limitations, especially with more complex geometries, making it less attractive for industrial applications.

Bibliography

[1] K. Carlberg, C. Bou-Mosleh, and C. Farhat. "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations". In: *International Journal for Numerical Methods in Engineering* 86.2 (2011), pp. 155–181. DOI: https://doi.org/10.1002/nme.3050.

[2] A. Scardigli, R. Arpa, A. Chiarini, and H. Telib. "Enabling of Large Scale Aerodynamic Shape Optimization Through POD-Based Reduced-Order Modeling and Free Form Deformation". In: *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*. Ed. by E. Minisci, M. Vasile, J. Periaux, N. R. Gauger, K. C. Giannakoglou, and D. Quagliarella. Cham: Springer International Publishing, 2019, pp. 49–63. ISBN: 978-3-319-89988-6. DOI: 10.1007/978-3-319-89988-6_4.

[3] R. Leroux and L. Cordier. "Dynamic mode decomposition for non-uniformly sampled data". In: *Experiments in Fluids* 57.5 (2016), p. 94. ISSN: 1432-1114. DOI: 10.1007/s00348-016-2165-1.

[4] T. Nonomura, K. Nankai, Y. Iwasaki, A. Komuro, and K. Asai. "Quantitative evaluation of predictability of linear reduced-order model based on particle-image-velocimetry data of separated flow field around airfoil". In: *Experiments in Fluids* 62.5 (2021), p. 112. ISSN: 1432-1114. DOI: 10.1007/s00348-021-03205-8.

[5] M. Castellani, Y. Lemmens, and J. E. Cooper. "Parametric reduced-order model approach for simulation and optimization of aeroelastic systems with structural nonlinearities". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 230.8 (2016), pp. 1359–1370. DOI: 10.1177/0954410015608888.

[6] W. Bova, E. Lappano, P. G. Catera, and D. Mundo. "Development of a parametric model order reduction method for laminated composite structures". In:

Composite Structures 243 (2020), p. 112219. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2020.112219.

[7] D. Xiao, P. Yang, F. Fang, J. Xiang, C. Pain, and I. Navon. "Non-intrusive reduced order modelling of fluid–structure interactions". In: *Computer Methods in Applied Mechanics and Engineering* 303 (2016), pp. 35–54. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2015.12.029.

[8] C. W. Rowley, T. Colonius, and R. M. Murray. "Model reduction for compressible flows using POD and Galerkin projection". In: *Physica D: Nonlinear Phenomena* 189.1 (2004), pp. 115–129. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2003.03.001.

[9] D. A. Bistrian and I. M. Navon. "An improved algorithm for the shallow water equations model reduction: Dynamic Mode Decomposition vs POD". In: *International Journal for Numerical Methods in Fluids* 78.9 (2015), pp. 552–580. DOI: https://doi.org/10.1002/fld.4029.

[10] A. Solera-Rico, C. Sanmiguel Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. T. M. Dawson, and R. Vinuesa. "β-Variational autoencoders and transformers for reduced-order modelling of fluid flows". In: *Nature Communications* 15.1 (Feb. 2024), p. 1361. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45578-4.

[11] K. Carlberg, M. Barone, and H. Antil. "Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction". In: *Journal of Computational Physics* 330 (2017), pp. 693–734. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.10.033.

[12] L. Sirovich. "Turbulence and the dynamics of coherent structures. I. Coherent structures". In: *Quart. Appl. Math.* 45 (1987), pp. 561–571. DOI: https://doi.org/10.1090/qam/910462.

[13] P. Benner, S. Gugercin, and K. Willcox. "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems". In: *SIAM Review* 57.4 (2015), pp. 483–531. DOI: 10.1137/130932715.

[14] J. Weller, E. Lombardi, M. Bergmann, and A. Iollo. "Numerical methods for low-order modeling of fluid flows based on POD". In: *International Journal for Numerical Methods in Fluids* 63.2 (2010), pp. 249–268. DOI: https://doi.org/10.1002/fld.2025.

[15] A. Iollo, S. Lanteri, and J.-A. Désidéri. "Stability Properties of POD–Galerkin Approximations for the Compressible Navier–Stokes Equations". In: *Theoretical and Computational Fluid Dynamics* 13.6 (Mar. 2000), pp. 377–396. ISSN: 1432-2250. DOI: 10.1007/s001620050119.

[16] A. Scardigli. "Aerodynamic Shape Optimization through Reduced-Order Modelling in Industrial Problems". url: https://hdl.handle.net/2318/1995442. PhD thesis. Politecnico di Torino - Università di Torino, 2018.

[17] T. Taddei. "A Registration Method for Model Order Reduction: Data Compression and Geometry Reduction". In: *SIAM Journal on Scientific Computing* 42.2 (2020), A997–A1027. DOI: 10.1137/19M1271270.

[18] S. Cucchiara, A. Iollo, T. Taddei, and H. Telib. "Model order reduction by convex displacement interpolation". In: *Journal of Computational Physics* 514 (2024), p. 113230. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2024.113230.

[19] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174. DOI: 10.1109/CVPR.2019.00025.

[20] R. Zamolo, L. Bacer, D. Miotti, E. Nobile, and M. Munerato. "RBF-FD meshless simulation of 3D fully developed flow and heat transfer in triply periodic minimal surfaces". In: *International Journal of Heat and Mass Transfer* 242 (2025), p. 126798. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2025.126798.

[21] R. Zamolo. "Radial basis function-finite difference meshless methods for CFD problems". url: https://hdl.handle.net/11368/2991045. PhD thesis. Università degli Studi di Trieste, 2018.

[22] F. Dicech, R. Zamolo, and L. Parussini. "Parametric reduced order model built from RBF-FD meshless simulations of flow and temperature fields in a 3D pipe with wavy surfaces". In: *Computers & Fluids* 301 (2025), p. 106803. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2025.106803.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[24] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN: 0-262-18253-X. DOI: https://doi.org/10.7551/mitpress/3206.001.0001.

[25] R. Hardy. "Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968–1988". In: *Computers & Mathematics with Applications* 19.8 (1990), pp. 163–208. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(90)90272-L.

[26] M. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511543241.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[29] M. Kennedy and A. O'Hagan. "Predicting the output from a complex computer code when fast approximations are available". In: *Biometrika* 87.1 (2000), pp. 1–13. DOI: https://doi.org/10.1093/biomet/87.1.1.

[30] L. L. Gratiet and J. Garnier. "Recursive co-kriging model for design of computer experiments with multiple levels of fidelity". In: *Int J Uncertain Quantif* 4.5 (2014), pp. 365–386. DOI: 10.1615/int.j.uncertaintyquantification.2014006914.

[31] Z.-H. Han and S. Görtz. "Hierarchical kriging model for variable-fidelity surrogate modeling". In: *AIAA journal* 50.9 (2012), pp. 1885–1896. DOI: https://doi.org/10.2514/1.J051354.

[32] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. "Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2198 (2017), p. 20160751. DOI: 10.1098/rspa.2016.0751.

[33] X. Meng and G. E. Karniadakis. "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems". In: *Journal of Computational Physics* 401 (2020), p. 109020. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2019.109020.

[34] M. Guo, A. Manzoni, M. Amendt, P. Conti, and J. S. Hesthaven. "Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114378. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114378.

[35] M. Motamed. *A multi-fidelity neural network surrogate sampling method for uncertainty quantification*. 2020. DOI: https://doi.org/10.48550/arXiv.1909.01859.

[36] J. Hesthaven and S. Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363 (2018), pp. 55–78. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.02.037.

[37] S. M. Ross. *Stochastic processes*. John Wiley & Sons, 1995.

[38] D. G. Krige. "A statistical approach to some basic mine valuation problems on the Witwatersrand". In: *Journal of the Southern African Institute of Mining and Metallurgy* 52.6 (1951), pp. 119–139.

[39] G. B. Wright. "Radial basis function: numerical and analytical developments". PhD thesis. University of Colorado, 2003.

[40] M. Giselle Fernández-Godino. "Review of multi-fidelity models". In: *Advances in Computational Science and Engineering* 1.4 (2023), pp. 351–400. ISSN: 2837-1739. DOI: 10.3934/acse.2023015.

[41] L. Folcarelli, A. Spagnolo, F. Dicech, A. Ferrero, F. Masseni, and D. Pastrone. "Multi-Fidelity Modeling of a Lean Premixed Swirl-Stabilized Hydrogen Burner With Axial Air Injection". In: *AIAA SCITECH 2025 Forum*. 2025. DOI: 10.2514/6.2025-0941.

[42] T. Benamara, P. Breitkopf, I. Lepot, C. Sainvitu, and P. Villon. "Multi-fidelity POD surrogate-assisted optimization: Concept and aero-design study". In: *Structural and Multidisciplinary Optimization* 56.6 (2017), pp. 1387–1412. DOI: https://doi.org/10.1007/s00158-017-1730-4.

[43] C. Perron, D. Sarojini, D. Rajaram, J. Corman, and D. Mavris. "Manifold alignment-based multi-fidelity reduced-order modeling applied to structural analysis". In: *Structural and Multidisciplinary Optimization* 65.8 (2022), p. 236. DOI: 10.1007/s00158-022-03274-1.

[44] K. Decker, N. Iyengar, D. Rajaram, C. Perron, and D. Mavris. "Manifold alignment-based nonintrusive and nonlinear multifidelity reduced-order modeling". In: *AIAA Journal* 61.1 (2023), pp. 454–474. DOI: https://doi.org/10.2514/1.J061720.

[45] A. Thenon, V. Gervais, and M. L. Ravalec. "Multi-fidelity meta-modeling for reservoir engineering-application to history matching". In: *Computational geosciences* 20.6 (2016), pp. 1231–1250. DOI: https://doi.org/10.1007/s10596-016-9587-y.

[46] A. Bertram, C. Othmer, and R. Zimmermann. "Towards real-time vehicle aerodynamic design via multi-fidelity data-driven reduced order modeling". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018, p. 0916. DOI: https://doi.org/10.2514/6.2018-0916.

[47] X. Wang, J. Kou, and W. Zhang. "Multi-Fidelity surrogate reduced-order modeling of steady flow estimation". In: *International Journal for Numerical Methods in Fluids* 92.12 (2020), pp. 1826–1844. DOI: https://doi.org/10.1002/fld.4850.

[48] H. Yang, S. H. Hong, G. Wang, and Y. Wang. "Multi-fidelity reduced-order model for GPU-enabled microfluidic concentration gradient design". In: *Engineering with Computers* 39.4 (2023), pp. 2869–2887. DOI: https://doi.org/10.1007/s00366-022-01672-z.

[49] M. Kast, M. Guo, and J. S. Hesthaven. "A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems". In: *Computer Methods in Applied Mechanics and Engineering* 364 (2020), p. 112947. DOI: https://doi.org/10.1016/j.cma.2020.112947.

[50] J. Yu and J. S. Hesthaven. "Flowfield reconstruction method using artificial neural network". In: *Aiaa Journal* 57.2 (2019), pp. 482–498. DOI: https://doi.org/10.2514/1.J057108.

[51] H. Kang, Z. Tian, G. Chen, L. Li, and T. Chu. "Investigation on the nonintrusive multi-fidelity reduced-order modeling for PWR rod bundles". In: *Nuclear Engineering and Technology* 54.5 (2022), pp. 1825–1834. DOI: https://doi.org/10.1016/j.net.2021.10.036.

[52] L. Partin, G. Geraci, A. A. Rushdi, M. S. Eldred, and D. E. Schiavazzi. "Multifidelity data fusion in convolutional encoder/decoder networks". In: *Journal of Computational Physics* 472 (2023), p. 111666. DOI: https://doi.org/10.1016/j.jcp.2022.111666.

[53] D. H. Song and D. M. Tartakovsky. "Transfer learning on multifidelity data". In: *Journal of Machine Learning for Modeling and Computing* 3.1 (2022). DOI: 10.1615/JMachLearnModelComput.2021038925.

[54] J. Kou, C. Ning, and W. Zhang. "Transfer learning for flow reconstruction based on multifidelity data". In: *AIAA Journal* 60.10 (2022), pp. 5821–5842. DOI: https://doi.org/10.2514/1.J061647.

[55] S. Fresca and A. Manzoni. "POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 388 (2022), p. 114181. DOI: https://doi.org/10.1016/j.cma.2021.114181.

[56] S. E. Ahmed and P. Stinis. "A multifidelity deep operator network approach to closure for multiscale systems". In: *Computer Methods in Applied Mechanics and Engineering* 414 (2023), p. 116161. DOI: https://doi.org/10.1016/j.cma.2023.116161.

[57] F. Dicech, K. Gkaragkounis, L. Parussini, A. Spagnolo, and H. Telib. "Integration of multi-fidelity methods in parametrized non-intrusive reduced order models for industrial applications". In: *Journal of Computational Science* 85 (2025), p. 102511. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2024.102511.

[58] P. Theissen, J. Wojciak, K. Heuler, R. Demuth, et al. "Experimental Investigation of Unsteady Vehicle Aerodynamics under Time-Dependent Flow Conditions - Part 1". In: *SAE Technical Paper*. 2011-01-0177. 2011. DOI: https://doi.org/10.4271/2011-01-0177.

[59] A. Heft, T. Indinger, and N. Adams. "Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations". In: *SAE Technical Paper*. 2012-01-0168. 2012. DOI: https://doi.org/10.4271/2012-01-0168.

[60] F. Salmoiraghi, A. Scardigli, H. Telib, and G. Rozza. "Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation". In: *International Journal of Computational Fluid Dynamics* 32.4-5 (2018), pp. 233–247. DOI: 10.1080/10618562.2018.1514115.

[61] Optimad Srl. *mimic*.

[62] E. M. Papoutsis-Kiachagias, V. G. Asouti, K. C. Giannakoglou, K. Gkagkas, S. Shimokawa, and E. Itakura. "Multi-point aerodynamic shape optimization of cars based on continuous adjoint". In: *Structural and Multidisciplinary Optimization* 59.2 (2019), pp. 675–694. ISSN: 1615-1488. DOI: 10.1007/s00158-018-2091-3.

[63] R. Soares, K. Garry, and J. Holt. "Comparison of the Far-Field Aerodynamic Wake Development for Three DrivAer Model Configurations using a Cost-Effective RANS Simulation". In: *SAE Technical Paper*. 2017-01-1514. 2017. DOI: https://doi.org/10.4271/2017-01-1514.

[64] OpenCFD Ltd. *OpenFOAM*.

[65] G. G. Wang. "Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points". In: *Journal of Mechanical Design* 125.2 (2003), pp. 210–220. ISSN: 1050-0472. DOI: 10.1115/1.1561044.

[66] C. Prince, M. Gu, and S. D. Peterson. "A Numerical Study of the Impact of Wavy Walls on Steady Fluid Flow Through a Curved Tube". In: *Journal of Fluids Engineering* 135.7 (May 2013), p. 071207. ISSN: 0098-2202. DOI: 10.1115/1.4023662.

[67] Y. J. Kim, M. Kim, S. Kim, J. K. Min, and M. Y. Ha. "Numerical study of fluid flow and convective heat transfer characteristics in a sinusoidal wavy circular tube". In: *Journal of Mechanical Science and Technology* 30.3 (2016), pp. 1185–1196. ISSN: 1976-3824. DOI: 10.1007/s12206-016-0222-6.

[68] L. Wang, S. Wang, F. Wen, X. Zhou, and Z. Wang. "Heat transfer and flow characteristics of U-shaped cooling channels with novel wavy ribs under stationary and rotating conditions". In: *International Journal of Heat and Mass Transfer* 126 (2018), pp. 312–333. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2018.05.123.

[69] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. New York, NY: Springer, 2009, p. 745. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7.

[70] L. Mihet-Popa, F. Blaabjerg, and I. Boldea. "Wind turbine Generator modeling and Simulation where rotational speed is the controlled variable". In: *IEEE Transactions on Industry Applications* 40.1 (2004), pp. 3–10. DOI: 10.1109/TIA.2003.821810.

[71] M. Sadman Sakib, D. Todd Griffith, S. Hossain, S. Bayat, and J. T. Allison. "Intracycle RPM control for vertical axis wind turbines". In: *Wind Energy* 27.3 (2024), pp. 202–224. DOI: https://doi.org/10.1002/we.2885.

[72] D. Burnham, S. Santoso, and E. Muljadi. "Variable rotor-resistance control of wind turbine generators". In: *2009 IEEE Power & Energy Society General Meeting*. 2009, pp. 1–6. DOI: 10.1109/PES.2009.5275637.

[73] T. Yamada, T. Kiwata, T. Kita, M. Hirai, N. Komatsu, and T. Kono. "Overspeed Control of a Variable-Pitch Vertical-Axis Wind Turbine by Means of Tail Vanes". In: *Journal of Environment and Engineering* 7.1 (2012), pp. 39–52. DOI: 10.1299/jee.7.39.

[74] J. M. Pinar Pérez, F. P. García Márquez, A. Tobias, and M. Papaelias. "Wind turbine reliability analysis". In: *Renewable and Sustainable Energy Reviews* 23 (2013), pp. 463–472. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.03.018.

[75] P. Cinnella, P. Congedo, L. Parussini, and V. Pediroda. "Quantification of thermodynamic uncertainties in real gas flows". In: *International Journal of Engineering Systems Modelling and Simulation* 2.1-2 (2010), pp. 12–24. DOI: 10.1504/IJESMS.2010.031867.

[76] F. Dicech and L. Parussini. "A multi-fidelity reduced-order model to quantify aerodynamic forces on a vertical-axis wind turbine with uncertain rotational speed". In: *UNCECOMP 2025 Proceedings*. https://hdl.handle.net/11368/3116658. 2025.

[77] K. McLaren. "A numerical and experimental study of unsteady loading of high solidity vertical axis wind turbines". PhD thesis. McMaster University, 2011.

[78] ANSYS, Inc. *Ansys®Free Student Fluent, Release 25.1*.

[79] F. Balduzzi, A. Bianchini, R. Maleci, G. Ferrara, and L. Ferrari. "Critical issues in the CFD simulation of Darrieus wind turbines". In: *Renewable Energy* 85 (2016), pp. 419–435. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2015.06.048.

[80] D. Rapos, C. Mechefske, and M. Timusk. "Dynamic sensor calibration: A comparative study of a Hall effect sensor and an incremental encoder for measuring shaft rotational position". In: *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2016, pp. 1–5. DOI: 10.1109/ICPHM.2016.7542858.

[81] Y. Wang, L. Wang, and Y. Yan. "Rotational speed measurement through digital imaging and image processing". In: *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2017, pp. 1–6. DOI: 10.1109/I2MTC.2017.7969697.

[82] Y.-H. Liao, L. Wang, and Y. Yan. "Instantaneous Rotational Speed Measurement of Wind Turbine Blades using a Marker-Tracking Method". In: *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2022, pp. 1–5. DOI: 10.1109/I2MTC48687.2022.9806658.

[83] D. Xiu. In: *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010. ISBN: 9780691142128. DOI: 10.2307/j.ctv7h0skv.

[84] J. Zhang. "Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey". In: *WIREs Computational Statistics* 13.5 (2021), e1539. DOI: https://doi.org/10.1002/wics.1539.

[85] A. Chiodi, A. Alaia, E. Lombardi, M. Cisternino, K. Gkaragkounis, and S. Shahpar. "Multidisciplinary Optimisation of Additive Manufactured Heat Exchangers for Aeronautical Applications". In: vol. Volume 13: Heat Transfer: General Interest / Additive Manufacturing Impacts on Heat Transfer; Wind Energy. Turbo Expo. June 2024, V013T13A026. DOI: 10.1115/GT2024-126786.

[86] T. Dixit, E. Al-Hajri, M. C. Paul, P. Nithiarasu, and S. Kumar. "High performance, microarchitected, compact heat exchanger enabled by 3D printing". In: *Applied Thermal Engineering* 210 (2022), p. 118339. ISSN: 1359-4311. DOI: https://doi.org/10.1016/j.applthermaleng.2022.118339.