SISSA

Scuola
Internazionale
Superiore di
Studi Avanzati

Mathematics Area - PhD course in
Mathematical Analysis, Modelling, and Applications

Nonlinear Parameter Space and Model Order Reductions enhanced by scientific machine learning

Candidate:
Francesco Romor

Advisor:
Prof. Gianluigi Rozza
Co-advisor:
Dr. Giovanni Stabile

Academic Year 2022-2023

Ai miei genitori

Abstract

Two of the main critical issues of model order reduction are the curse of dimensionality and the slow Kolmogorov n-width decay. With this thesis we want to provide novel methodologies that can ameliorate them: for the former we develop nonlinear parametric space reduction methods and for the latter nonlinear model order reduction methods. The most successful strategies applied for both approaches are local linear approximants and nonlinear solution manifold learning methods from scientific machine learning. Despite many successful applications envision purely data-driven surrogate modelling, we focus our efforts towards the advancement of more interpretable frameworks that combine the numerical understanding of the models underneath with machine learning paradigms.

The first part is devoted to nonlinear parameters space reduction. We introduce hierarchical active subspaces for local parameter space reduction: new metrics to perform the clustering and classify the regions of the parameters space based on the local Grassmannian manifold's intrinsic dimension are validated through academic benchmarks. Shape optimization studies for automotive applications are the target of new multi-fidelity regression methods: nonlinear autoregressive Gaussian processes are combined with the response surfaces designed on the active subspace. Finally, the generation with free form deformation of new computational geometries that satisfy multilinear geometrical constraints, such as the volume, is sped up with constrained generative models. New tailored metrics to validate the posterior distributions are defined.

The second part is focused on nonlinear model order reduction. Friedrichs' systems are presented as a new class of structure-preserving reduced order models (ROMs) with the possibility to obtain optimally stable error estimates. At the same time, Friedrichs' systems are employed as testing ground for new repartitioning algorithms for domain decomposable ROMs based on the discontinuous Galerkin method. Vanishing viscosity solutions are inferred with graph neural networks as the limit of the predictions of high viscosity domain decomposable ROMs. The last two chapters are dedicated to the embedding inside residual-based reduced numerical schemes of nonlinear approximants of the solution manifold. Such nonlinear parametrization maps are represented by neural networks obtained with teacher-student training or a combination of neural networks and singular value decomposition's modes. Finally, local nonlinear manifold approximations and adaptive hyper-reduction methods are used to reduce the transient dynamics of numerical models affected by a slow Kolmogorov n-width decay.

Every result is supported by numerical experiments performed on parametrized computational fluid dynamics benchmarks.

Table of contents

List of figures

List of tables

Acronyms

AAE Adversarial Autoencoder.

ADR Advection–Diffusion–Reaction test case.

AE Autoencoder.

AS Active Subspaces.

BEGAN Boundary Equilibrium Generative Adversarial Networks.

C Collocated.

CAE Convolutional Autoencoder.

CFD Computational Fluid Dynamics.

cFFD Constrained Free Form Deformation.

cGM Constrained Generative Models.

CLE Compressible Linear Elasticity test case.

CNN Convolutional Neural Networks.

CNS Compressible Navier-Stokes equations test case.

DD-ROM Domain Decomposable ROMs.

DDES Delayed Deatched Eddy-Simulations.

DDES-AS response surfaces of Delayed Deatched Eddy-Simulations with Active Subspaces.

DDES-RANS Multi-fidelity RANS-DDES regression.

DEIM Discrete Empirical Interpolation Method.

DGM Discontinuous Galerkin Method.

ECSW Energy-Conserving Sampling and Weighting method.

FB Field Basis.

FEM Finite Element Method.

FFD Free Form Deformation.

FOM Full-Order Models.

FS Friedrichs' Systems.

FVM Finite Volumes Method.

GAN Generative Adversarial Networks.

GM Generative Models.

GNAT Gauss-Newton Tensor Approximation.

GNN Graph Neural Networks.

GP	Gaussian Process.
GPR	Gaussian Process Regression.
HAS	Hierarchical top-down clustering with Active Subspaces.
HB	naval Hull's bulb test case.
HF	High-Fidelity.
INS	Incompressible Navier-Stokes equations test case.
KnW	Kolmogorov n-width decay.
LAS	Local Active Subspaces.
LF	Low-Fidelity.
LSPG	Least-Squares Petrov-Galerkin.
LSTM	Long Short-Term Memory networks.
MF	Multi-Fidelity.
ML	Machine Learning.
MOR	Model Order Reduction.
MS	Maxwell equations in Stationary regime test case.
NARGP	Nonlinear Autoregressive Gaussian Process Regression.
NARGP-AS	Nonlinear Autoregressive Gaussian Process Regression with Active Subspaces.
NCL	Nonlinear Conservation Law test case.
NLL	Nonlinear Level Set Learning.
NM	Nonlinear Manifold.
NN	Neural Networks.
PCA	Principal Component Analysis.
PDE	Partial Differential Equation.
POD	Proper Orthogonal Decomposition.
PODI	Proper Orthogonal Decomposition with Interpolation.
RANS	Reynolds Averaged Navier-Stokes equations.
RB	Residual Basis.
RBF	Radial Basis Functions.
ROC	Reduced over-collocation.
ROM	Reduced Order Models.
rSVD	Randomized Singular Value Decomposition.
SB	Stanford Bunny test case.
SOPT	S-Optimal sampling for hyper-reduction.
SVD	Singular Value Decomposition.
SWE	Shallow Water Equations test case.
TS	Teacher-Student Training.
UP	UPdate of hyper-reduction's nodes.
VAE	Variational Autoencoder.
VV	Vanishing Viscosity.

Chapter 1

Introduction

While the technological advancement brings more and more computational power to perform high-fidelity numerical simulations, at the same time mathematically principled ways to reduce the computational costs are ubiquitous in real-world applications. This is especially the case for digital twins that require real-time responses and the simultaneous adaptation of the embedded numerical models and integration of the observations coming from sensors. The models we consider are represented by parametric, often nonlinear and time-dependent, partial differential equations (PDEs).

Model order reduction, also coupled with dimension reduction of the space of the parameters, has produced a substantial literature of linear certified projection-based approaches. However, they have a limited range of validity since the solution manifold of a large class of PDEs is not reducible with linear approximants. This is not a minor concern, a great number of real-world applications are affected: from the simplest linear advection problem with discontinuous initial solutions to multiphase flows, transient dynamics, advection-dominated problems, decomposable systems, like fluid-structure interaction problems, and high-resolution turbulent flows.

As the reduced basis method defines a more efficient and sufficiently accurate low-dimensional approximation space of the solution manifold of parametric PDEs with respect to the classical discretization spaces (finite elements, finite volumes, discontinuous finite elements), many techniques from scientific machine learning introduce nonlinear approximations spaces more suitable to tackle a slow Kolmogorov n-width decay. The price paid for the higher expressiveness of the low-dimensional nonlinear approximants is the loss of interpretability of the results and the less manageability of the new nonlinear ansatz spaces when designing new numerical schemes upon them. A widespread conduct is not to discern between reduced order modelling and machine learning with the application of non-intrusive data-driven approaches that forget about the underlying mathematical and physical frameworks that define the models beneath. While this is an effective and relatively straightforward strategy to implement, more has to be done to bring out synergies between numerical modelling and scientific machine learning.

Recognizing that nonlinear approximants of the solution manifold are necessary, we believe that an effort towards more interpretable and tailored approaches for numerical applications is valuable.

This is our aim. We develop efficient implementations of the original nonlinear manifold method introduced by Carlberg et al. [161] with hyper-reduction. Hyper-reduction is the most crucial part, without it the initial formulation cannot be employed in practice due to the high computational costs. We remark that this approach differs from standard non-intrusive data-driven ROMs since the predicted numerical solutions are not obtained from black-box evaluations of neural networks but through the residual minimization of first principles through numerical schemes. We propose two variants. The first one, in chapter 6, is more suited to solution manifolds with a slow Kolmogorov n-width decay since it employs truly nonlinear approximants represented by generic neural networks architectures such as convolutional autoencoders and graph neural networks. The second one, in chapter 7, is more suited to solution manifolds with a moderately slow Kolmogorov n-width decay, for which sufficiently high dimensional linear approximants can still be employed. The latter formulation facilitates the use of adaptive hyper-reduction node selection algorithms and local nonlinear solution manifolds approximation. These new approaches are tested reducing the transient dynamics of incompressible and compressible flows in `OpenFoam` [264], extending the open-source software library `ITHACA-FV` [238, 237] tailored for ROMs based on the finite volumes method (FVM).

Another novel framework that we introduce in chapter 5 involves the employment of graph neural networks (GNNs) to infer the vanishing viscosity solutions of parametric hyperbolic PDEs in a multi-fidelity and multi-resolution fashion. In this case, the training of heavy autoencoders of GNNs is bypassed: classical projection-based ROMs are used to predict a succession of high viscosity solutions whose limit are the vanishing viscosity solutions avoiding the problem of a slow Kolmogorov n-width decay. The testing ground is represented by Friedrichs' systems, a new class of structure preserving ROMs that include many classes of PDEs, including symmetric hyperbolic PDEs, like the linearized Euler equations in entropy variables. The discretization method of choice is the Discontinuous Galerkin method (DGM) in `deal.II` [12]. Domain decomposable (DD-ROMs) approaches for local solution manifold approximation are also investigated, with the introduction of strategies to repartition the computational domain: new indicators that generate a better partition of the domain in terms of solution manifold approximation are studied.

Often, when ROMs are designed for real-world applications, a great number of parameters influence the numerical models to reduce. This not only exacerbates the computational costs of studies that require multi-queries of the FOMs, like shape optimization or uncertainty quantification but also increases the difficulties in the realization of ROMs due to the curse of dimensionality. The Active Subspaces method introduced by Constantine [60] is effectively employed to tackle this problem. We decline its employment in two novel approaches. For local regressions and the classification of the parameter space and of the computational domain based on the local AS dimension in chapter 2. For Gaussian process regression with nonlinear multi-fidelity information fusion in chapter 3. The open-source software library `ATHENA` [218] was developed in collaboration with Marco Tezzele to perform parameter space reduction with Active Subspaces and some nonlinear variants like kernel-based active subspaces [221] and Nonlinear Level Set Learning (NLL) [283].

Sometimes even just obtaining the computational mesh for some specific real-world applications is costly. This is the case for patient-specific biomedical applications and many other shape optimization tasks. In order to speed up a version of Free Form Deformation (FFD) that imposes multilinear constraints, like volume preservation, we implement multilinear geometrically constrained generative models (cGMs) in chapter 4. We show that these ML architectures can effectively reduce the space of FFD displacements with a low-dimensional latent space, they exactly impose multilinear constraints and they can be coupled with non-intrusive MOR. Crucial is the validation of the posterior distributions with *ad hoc* metrics.

1.1 Thesis outline

This thesis is organized into two parts. The first part covers the presentation of novel numerical methods that combine linear and certified classical procedures, such as the active subspaces method, with nonlinear dimension reduction techniques from machine learning. The second part involves nonlinear model order reduction: to solve the problem of a slow Kolmogorov n-width of the solution manifold of some characteristic parametric partial differential equations, new strategies that employ nonlinear solution manifold approximants, hyper-reduction and domain decomposable reduced order models are developed. The outline of the thesis is summarized in Figure 1.1.

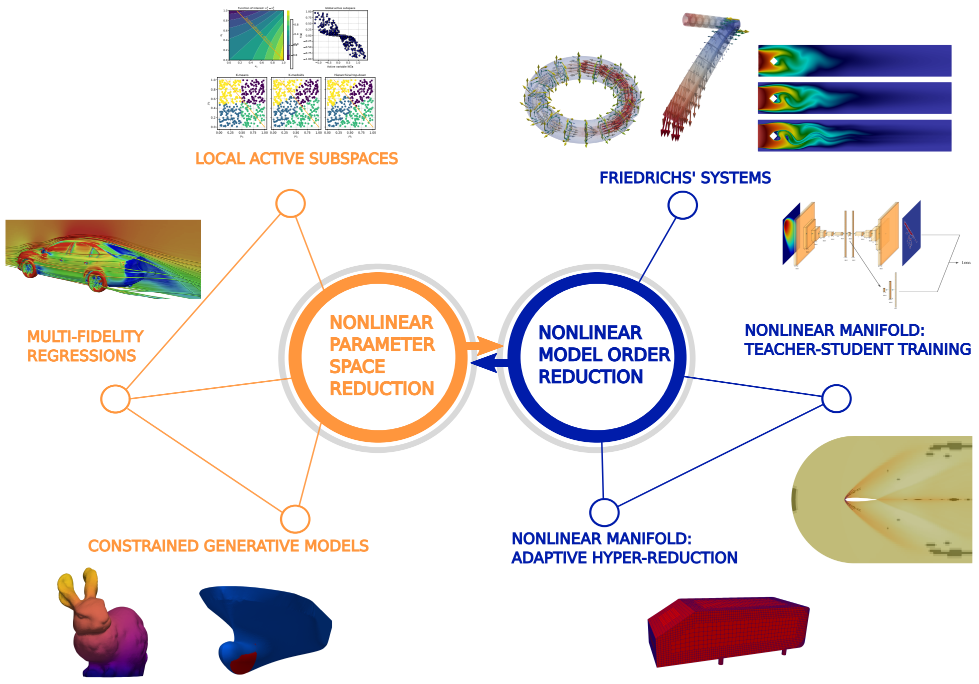

LOCAL ACTIVE SUBSPACES
FRIEDRICHS' SYSTEMS
MULTI-FIDELITY REGRESSIONS
NONLINEAR PARAMETER SPACE REDUCTION
NONLINEAR MODEL ORDER REDUCTION
NONLINEAR MANIFOLD: TEACHER-STUDENT TRAINING
CONSTRAINED GENERATIVE MODELS
NONLINEAR MANIFOLD: ADAPTIVE HYPER-REDUCTION

Fig. 1.1 Thesis structure: the first part focuses on nonlinear parameter space reduction, the second part on nonlinear reduced order modelling.

- Part I. **Nonlinear parameter space reduction**

– chapter 2 **Local Active Subspaces**. This chapter refers to the preprint *A local approach to parameter space reduction for regression and classification tasks* [220] in collaboration with Marco Tezzele (Oden Institute, University of Texas at Austin).

– chapter 3 **Multi-fidelity nonlinear regression with Active Subspaces**. This chapter refers to the preprint *Multi-fidelity data fusion through parameter space reduction with applications to automotive engineering* [225] in collaboration with Marco Tezzele (Oden Institute, University of Texas at Austin), and Markus Mrosek and Carsten Othmer from Volkswagen. This work was partially supported by the European Commission H2020 ARIA (Accurate ROMs for Industrial Applications) project.

– chapter 4 **Constrained Generative Models**. This chapter refers to the preprint *Generative Models for the Deformation of Industrial Shapes with Linear Geometric Constraints: model order reduction and reduction in parameter space* [189] in collaboration with Guglielmo Padula (University of Trieste).

- Part II. **Nonlinear model order reduction**

– chapter 5 **Friedrichs' systems**. This chapter refers to the preprint *Friedrichs' systems discretized with the Discontinuous Galerkin method: domain decomposable model order reduction and Graph Neural Networks approximating vanishing viscosity solutions* [224] in collaboration with Davide Torlo (SISSA).

– chapter 6 **Hyper-reduced nonlinear manifold method: teacher-student training**. This chapter refers to the publication *Non-linear manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation method* [223].

– chapter 7 **Hyper-reduced nonlinear manifold method: adaptive strategies**. This chapter refers to the preprint *Explicable hyper-reduced order models on nonlinearly approximated solution manifolds of compressible and incompressible Navier-Stokes equations* [222].

Part I

Nonlinear parameter space reduction

Chapter 2

Local Active Subspaces

Parameter space reduction has been proved to be a crucial tool to speed up the execution of many numerical tasks such as optimization, inverse problems, sensitivity analysis, and surrogate models' design, especially when in the presence of high-dimensional parametrized systems. We introduce a new method called local active subspaces (LAS), which explores the synergies of active subspaces with supervised clustering techniques in order to carry out a more efficient dimension reduction in the parameter space. The clustering is performed without losing the input-output correlations defining a new clustering metric induced by the global active subspace or the local Grassmannian manifold. Two possible clustering algorithms are presented: K-medoids and a hierarchical top-down approach, which is able to impose a variety of subdivision criteria specifically tailored for parameter space reduction tasks. This method is particularly useful for the community working on surrogate modelling. Frequently, the parameter space presents subdomains where the objective function of interest varies less on average along different directions. So, it could be approximated more accurately if restricted to those subdomains and studied separately. The new method is tested over several numerical experiments of increasing complexity, vectorial outputs are also treated, and it is explained how to classify the different regions with respect to the local active subspace dimension. Employing this classification technique as a preprocessing step in the parameter space, or output space in the case of vectorial outputs, brings remarkable results for the purpose of surrogate modelling. The benchmarks considered are an elliptic partial differential equation with random inputs and an incompressible flow past a NACA airfoil geometrically parametrized.

Contents

2.1 Literature review

Parameter space reduction [60] is a rapidly growing field of interest which plays a key role in fighting the curse of dimensionality. The need of reducing the number of design inputs is particularly important in engineering for advanced CFD simulations to model complex phenomena, especially in the broader context of model order reduction [54, 53, 24, 227] and industrial numerical pipelines [36, 35, 226, 247].

Active subspaces [60] is one of the most used techniques for linear reduction in input spaces. It has been proven useful in many numerical tasks such as regression, using a multi-fidelity data fusion approach with a surrogate model built on top of the AS as low-fidelity model [225], shape optimization [170, 249, 30] and a coupling with the genetic algorithm to enhance its performance [73, 72], inverse problems [280], and uncertainty quantification [63]. It has also been used to enhance classical model order reduction techniques such as POD-Galerkin [248], and POD with interpolation [71, 252]. Other attempts toward nonlinear parameter space reduction have been proposed recently: kernel-based active subspaces [221], nonlinear level-set learning [283], and active manifold [33] are the most promising. In [50], instead, they project the input parameters onto a low-dimensional subspace spanned by the eigenvectors of the Hessian corresponding to its dominating eigenvalues.

In this work, we propose a new local approach for parameter space dimensionality reduction for both regression and classification tasks, called Local Active Subspaces (LAS). We do not simply apply a clustering technique to preprocess the input data, we propose a supervised metric induced by the presence of a global active subspace. The directions individuated by local active subspaces are locally linear, and they better capture the latent manifold of the target function.

From a wider point of view, there is an analogy between local parameter space reduction and local model order reduction. With the latter, we mean both a spatial domain decomposition approach for model order reduction of parametric PDEs in a spatial domain $\Omega \subset \mathbb{R}^d$ and a local reduction approach in the parameter space. As representatives methods for the first paradigm we report the reduced basis element method [167], which combines the reduced basis method in each subdomain with a mortar-type method at the interfaces, and more in general domain decomposition methods applied to model order reduction. For the second approach, we cite the interpolation method in the Grassmannian manifold of the reduced subspaces [7]; in particular in [69] the K-medoids clustering algorithm with Grassmann metric is applied to the discrete Grassmann manifold of the training snapshots as a step to perform local model order reduction. With this work, we want to fill the gap in the literature regarding localization methods in the context of parameter space reduction.

In this regard, other methods have been developed in the last years exploiting the localization idea. We mention localized slice inverse regression (LSIR) [272] which uses local information of the slices for supervised regression and semi-supervised classification. LSIR improves local discriminant information [122] and local Fischer discriminant analysis [240] with more efficient computations for classification problems. The main difference between slice inverse regression (SIR) [162] and AS is in the construction of the projection matrix. While SIR needs the elliptic assumption, AS exploits the gradients of the function of interest with respect to the input parameters. Recently a new work on the subject was disclosed [277]. Here we emphasize the differences and the original contributions of our work: 1) we implemented hierarchical top-down clustering applying K-medoids with a new metric that includes the gradient information through the active subspace. In the previous work, they employed hierarchical bottom-up clustering with unweighted average linkage and a distance obtained as a weighted sum of the Euclidean distance of the inputs and the cosine of the angle between the corresponding gradients; 2) we included vector-valued objective functions and answered questions about the employment of the new method to decrease the ridge approximation error with respect to a global approach; 3) we also focused on classification algorithms, and we devised a method to classify the inputs based on the local active subspace dimension with different techniques, including the use of the Grassmannian metric; 4) our benchmarks include vector-valued objective functions from computational fluid dynamics. We also show that clustering the outputs with our classification algorithms as pre-processing step leads to more efficient surrogate models.

This work is organized as follows: in section 2.2 we briefly review the active subspaces method, in section 2.3 we introduce the clustering algorithms used and the supervised distance metric based on the presence of a global active subspace, focusing on the construction of response surfaces and providing theoretical considerations. In section 2.4 we present the algorithms to exploit LAS for classification. We provide extensive numerical results in section 2.5 from simple illustrative two-dimensional dataset to high-dimensional scalar and vector-valued functions. Finally, in section 2.6 we draw some conclusions and future perspectives.

2.2 Active subspaces for parameter space reduction

Active subspaces (AS) [60] are usually employed as dimension reduction method to unveil a lower dimensional structure of a function of interest f, or provide a global sensitivity measure not necessarily aligned with the coordinate axes [241]. Through spectral considerations about the second moment matrix of f, the AS identify a set of linear combinations of the input parameters along which f varies the most on average.

We will make some general assumptions about the inputs and function of interest [60, 281, 241]. Let us introduce the inputs as an absolutely continuous random vector $\mathbf{X}$ with values in $\mathbb{R}^n$ and probability distribution $\boldsymbol{\mu}$. We represent with $\mathcal{X} \subset \mathbb{R}^n$ the support of $\boldsymbol{\mu}$ and as such our parameter space. We want to compute the active subspace of a real-valued function $f : (\mathcal{X}, \mathcal{B}(\mathbb{R}^n), \boldsymbol{\mu}) \to \mathbb{R}$, where $\mathcal{B}(\mathbb{R}^n)$ is the Borel σ-algebra of $\mathbb{R}^n$.

An extension to vector-valued functions has been presented in [281] and extended for kernel-based AS in [221]. Even if in this section we focus only on scalar functions, the following considerations can be carried over to the multivariate case without too much effort.

Let Σ be the second moment matrix of ∇f defined as

$$\Sigma := E_{\boldsymbol{\mu}}[\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T] = \int (\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^T \, d\boldsymbol{\mu}, \tag{2.1}$$

where $E_{\boldsymbol{\mu}}$ denotes the expected value with respect to $\boldsymbol{\mu}$, and $\nabla_{\mathbf{x}} f = \nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial \mathbf{x}_1}, \dots, \frac{\partial f}{\partial \mathbf{x}_n}\right]^T$ is the column vector of partial derivatives of f. Its real eigenvalue decomposition reads $\Sigma = \mathbf{W}\Lambda\mathbf{W}^T$. We can retain the most energetic eigenpairs by looking at the spectral decay of the matrix Σ. The number r of eigenpairs we select is the active subspace dimension, and the span of the corresponding eigenvectors defines the active subspace. The partition is the following

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_1 & \\ & \boldsymbol{\Lambda}_2 \end{bmatrix}, \qquad \mathbf{W} = [\mathbf{W}_1 \quad \mathbf{W}_2], \tag{2.2}$$

where $\boldsymbol{\Lambda}_1 = \mathrm{diag}(\lambda_1, \dots, \lambda_r)$, and $\mathbf{W}_1$ contains the first r eigenvectors arranged by columns. With this matrix we can project the input parameters onto the active subspace, and its orthogonal complement, that is the inactive subspace, as follows:

$$\mathbf{Y} = P_r(\mathbf{X}) = \mathbf{W}_1\mathbf{W}_1^T\mathbf{X} \in \mathbb{R}^n, \qquad \mathbf{Z} = (I - P_r)(\mathbf{X}) = \mathbf{W}_2\mathbf{W}_2^T\mathbf{X} \in \mathbb{R}^n, \tag{2.3}$$

with $P_r : \mathbb{R}^n \to \mathbb{R}^n$ the linear projection operator $P_r := \mathbf{W}_1\mathbf{W}_1^T$. The selection of the active subspace dimension r can be set a priori, or by looking at the presence of a spectral gap [60], or by imposing a cumulative energy threshold for the eigenvalues.

We will consider the problem of ridge approximation [204] in our applications. The AS are, in fact, the minimizers of an upper bound of the ridge approximation error.

Definition 1 (Ridge approximation)**.** *Given $r \in \mathbb{N}$, $r \ll n$ and a tolerance $\varepsilon \geq 0$, find the profile function $h : (\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n), \boldsymbol{\mu}) \to \mathbb{R}$ and the r-rank projection $P_r : \mathbb{R}^n \to \mathbb{R}^n$ such that the following upper bound on the ridge approximation error is satisfied*

$$E_{\boldsymbol{\mu}}[\|f - h \circ P_r\|_2^2] \leq \varepsilon^2, \tag{2.4}$$

where $\|\cdot\|_2$ *is the* L^2*-norm of* $\mathbb{R}$.

For a fixed projection P_r the optimal profile $\tilde{h}$ is given by the conditioned random variable $E_{\boldsymbol{\mu}}[f|P_r]$. Under the additional assumptions on the probability distribution $\boldsymbol{\mu}$, reported in section 2.2.1, the AS can indeed be defined as a minimizer of an upper bound of the ridge approximation error [60, 281, 192, 221]. The proof is a direct consequence of the Poincaré inequality and standard properties of eigenspaces, and for this specific version of the theorem, it can be found in [192].

Theorem 1 (Definition of AS through ridge approximation)**.** *The solution* P_r *of the ridge approximation problem in definition 1, with optimal profile* $\tilde{h} = E_{\boldsymbol{\mu}}[f|P_r]$*, is the orthogonal projector to the eigenspace of the first r-eigenvalues of* Σ *ordered by magnitude*

$$\Sigma v_i = \lambda_i v_i \qquad \forall i \in \{1,\dots,n\}, \qquad \tilde{P}_r = \sum_{j=1}^{r} v_j \otimes v_j,$$

with $r \in \mathbb{N}$ *chosen such that*

$$\begin{aligned} E_{\boldsymbol{\mu}}\left[\|f-\tilde{h}\|_2^2\right] &\leq C(C_P,\tau) \underset{\substack{P^2=P, P=P^T,\\ rank(P)=r}}{\operatorname{argmin}} E_{\boldsymbol{\mu}_i}[\|(Id-P)\nabla f\|^2]^{\frac{1}{1+\tau}} \\ &\leq C(C_P,\tau)\left(\sum_{i=r+1}^{m}\lambda_i\right)^{\frac{1}{1+\tau}} \leq \varepsilon^2, \end{aligned}$$

with $C(C_P,\tau)$ *a constant depending on* $\tau > 0$ *related to the choice of* $\boldsymbol{\mu}$ *and on the Poincaré constant* C_P*, and* $\tilde{h} = E_{\boldsymbol{\mu}}[f|\sigma(P_r)]$ *is the conditional expectation of f given the* σ*-algebra generated by the random variable* $P_r \circ \mathbf{X}$.

To ease the notation, in the following considerations we will consider only the first three classes of probability distribution in the assumptions of the section 2.2.1, such that $\tau = 0$.

2.2.1 Subspace Poincaré inequality

The probabilistic Poincaré inequality for conditional probability densities or subspace Poincaré inequality [192] is valid at least for the following classes of absolutely continuous probability densities $\boldsymbol{\mu}$ with p.d.f. ρ.

Assumption 1. The p.d.f $\rho : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ satisfies one of the following:

1. $\mathcal{X}$ is bounded connected open with Lipschitz boundary, ρ is the uniform density distribution.
2. $\mathcal{X}$ is convex and bounded, $\exists\, \delta, D > 0 : 0 < \delta \leq \|\rho(\mathbf{x})\|_{L^\infty} \leq D < \infty\,, \forall \mathbf{x} \in \mathcal{X}$,
3. $\mathcal{X} = \mathbb{R}^n$, $\rho(\mathbf{x}) \sim \exp(-V(\mathbf{x}))$ where $V : \mathbb{R}^n \to (-\infty, \infty]\,, V \in \mathcal{C}^2$ is α-uniformly convex,

$$\mathbf{u}^T \mathrm{Hess}(V(\mathbf{x}))\mathbf{u} \geq \alpha\|\mathbf{u}\|_2^2, \quad \forall \mathbf{x}, \mathbf{u} \in \mathbb{R}^n, \tag{2.5}$$

where $\text{Hess}(V(\mathbf{x}))$ is the Hessian of $V(\mathbf{x})$.

4. $\mathcal{X} = \mathbb{R}^n$, $\rho(\mathbf{x}) \sim \exp(-V(\mathbf{x}))$ where V is a convex function. In this case, we require also f Lipschitz continuous.

The last class of p.d.f. provides a weaker bound (Lemma 4.3, [192]) on the ridge approximation error. For the previous classes $i \in \{1,2,3,4\}$ of p.d.f. an upper bound of the Poincaré constant $C_{P,i}$ is also provided:

$$C_{P,1} = C_{P,1}(\Omega), \qquad C_{P,2} = \frac{D\text{diam}(\mathcal{X})}{\pi\delta}, \qquad C_{P,3} = \frac{1}{\alpha}, \tag{2.6}$$

while the upper bound for $C_{P,4}$ requires the definition of other quantities and is proved in Lemma 4.4 [192].

2.2.2 Generalization of the upper bound on the approximation of the active subspace

We want to make some brief considerations about the accuracy of the active subspace as eigensubspace of the correlation matrix approximated with Monte Carlo. If we use the notation $W_1 \in \mathbb{R}^{n\times r}, W_2 \in \mathbb{R}^{n\times(n-r)}$ for the active and inactive subspaces (i.e. $P_r = W_1W_1^T$, $Id - P_r = W_2W_2^T$) and $\hat{W}_1 \in \mathbb{R}^{n\times r}, \hat{W}_2 \in \mathbb{R}^{n\times(n-r)}$ for the approximated active and inactive subspaces, we can bound the approximation error as done by Constantine in [60]: assuming f Lipschitz continuous, with high probability the following inequality is valid,

$$\text{dist}(\text{Im}(W_1), \text{Im}(\hat{W}_1)) \lesssim \frac{4L\sqrt{n}(\log(n))^{\frac{1}{2}}}{N^{\frac{1}{2}}\lambda_1(\lambda_r - \lambda_{r+1})}, \tag{2.7}$$

where L is the Lipschitz constant of f, $\{\lambda_1, \dots \lambda_n\}$ are the non-negative eigenvalues of $E_{\boldsymbol{\mu}_i}[\nabla f \otimes \nabla f]$ ordered decreasingly, and N is the number of Monte Carlo samples.

The bound in eq. (2.7) is obtained from Corollary 3.8 and Corollary 3.10 in [60]. It is founded on a matrix Bernstein inequality for a sequence of random uniformly bounded matrices (Theorem 6.1, [256]) and on the Corollary 8.1.11 from [105] that holds a bound on the sensitivity of perturbation of an invariant subspace.

From Corollary 8.1.11 of [105], a bound on the approximation error of the active subspace W_1 can be obtained making explicit $\|W_2^T E W_1\|_F$ with respect to the chosen numerical method for the discretization $\hat{C}$ of the integral $C = E_{\boldsymbol{\mu}_i}[\nabla f \otimes \nabla f]$: in [60] this has been done for the Monte Carlo method. In practice, we could use quasi Monte Carlo sampling methods with Halton or Sobol sequences [241], since

$$\begin{aligned}
\|W_2^T E W_1\|_F &\leq \sqrt{r(n-r)}\|W_2^T E W_1\|_{\max} \\
&\lesssim \sqrt{r(n-r)}D^*(\{x_i\}_i) \cdot \max_{i,j\in\{1,\dots,n\}}(V^{\text{HK}}(\nabla f_i \nabla f_j)) \\
&\lesssim 2\sqrt{r(n-r)}D^*(\{x_i\}_i) \cdot \max(|f|) \cdot \max_{i\in\{1,\dots,n\}}(V^{\text{HK}}(\nabla f_i)) \\
&\lesssim 2\sqrt{r(n-r)} \cdot \max_{i\in\{1,\dots,n\}}(V^{\text{HK}}(\nabla f_i))\frac{\log(N)^n}{N},
\end{aligned}$$

where V^{HK} is the Hardy–Krause variation and $D^*(\{x_i\}_i)$ is the star discrepancy of the quasi random sequence $\{x_i\}_i$. For the above result we have imposed $\mathcal{X} = [0,1]^n$, but it can be extended to different domains [21]. Thus, we obtain the bound

$$\begin{aligned}\operatorname{dist}(\operatorname{Im}(W_1), \operatorname{Im}(\hat{W}_1)) &\lesssim \frac{4\|W_2^T E W_1\|_F}{\lambda_r - \lambda_{r+1}} \\ &\lesssim \frac{8L\sqrt{r(n-r)} \cdot \max_{i\in\{1,\dots,n\}}(V^{\text{HK}}(\nabla f_i))}{\lambda_r - \lambda_{r+1}} \cdot \frac{\log(N)^n}{N}. \end{aligned} \tag{2.8}$$

Other numerical integration rules can be chosen so that different regularity conditions on the objective function may appear on the upper bound of the error, as the Lipschitz constant on eq. (2.7) or the Hardy–Krause variation on eq. (2.8). If the regularity of f is $\mathcal{C}^s$, we can also apply tensor product quadrature formulae or Smolyak's sparse quadrature rule [241]. For high-dimensional datasets and f less regular, the estimate in eq. (2.7) is the sharpest.

2.3 Localized parameter space reduction

Sometimes we do not have *a priori* knowledge about the target function's behavior in a particular parameter space region. This could lead to a poor selection of the parameters range, hugely affecting optimization tasks. In these cases, a preprocessing of the data using a clustering technique could be highly beneficial. With a clustering of the input parameters we can treat each subregion separately, and thus capture more accurately the target function's variability. This is always true for any function of interest, but for functions with global lower intrinsic dimensionality we can exploit such structure to enhance the clustering. To this end, we propose a new distance metric for K-medoids and hierarchical top-down clustering methods which exploits the global active subspace of the target function. By applying AS on each cluster we find the optimal rotation of the corresponding subregion of the input domain, which aligns the data along the active subspace of a given dimension.

In this section, we make some theoretical considerations regarding ridge approximation applied to partitions of the parameter space and review three clustering methods [119]: K-means, K-medoids, and hierarchical top-down clustering [144, 182]. We are going to use K-means as the baseline since the input parameter space is assumed to be a hyperrectangle — as is done in every practical case.

2.3.1 Ridge approximation with clustering and active subspaces

Regardless of the choice of clustering algorithm, given a partition of the parameter space we want to perform ridge approximation with AS in each subdomain. We will introduce some definitions and make some remarks to clarify the setting. The function of interest f represents scalar outputs, but the following statements can be extended to vector-valued outputs as well.

Definition 2 (Local ridge approximation with active subspaces)**.** *Given a partition of the domain $\mathcal{P} := \{S_i\}_{i\in\{1,\dots,d\}}$ and a map $r : \mathcal{P} \to \{1,\dots,n_r\}$, $n_r \ll n$ representing the local reduced dimension,* *the local ridge approximation with active subspaces of* $(f, \boldsymbol{\mu})$ *is the function* $R_{AS}(r, f, \boldsymbol{\mu}) : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ *that is defined locally for every* $S_i \in \mathcal{P}$ *as*

$$g|_{S_i} = E_{\boldsymbol{\mu}_i}[f|P_{r(S_i),i}], \tag{2.9}$$

where $\boldsymbol{\mu}_i := (1/\boldsymbol{\mu}(S_i)) \cdot \boldsymbol{\mu}|_{S_i} \in \mathbb{R}^n$*, and* $P_{r,i} : S_i \subset \mathbb{R}^n \to \mathbb{R}^n$ *is the orthogonal projector with rank* r *that satisfies the minimization problem:*

$$P_{r,i} = \underset{\substack{P^2=P, P=P^T,\\ rank(P)=r}}{\operatorname{argmin}} E_{\boldsymbol{\mu}_i} \|(Id - P)\nabla f\|^2. \tag{2.10}$$

With this definition, we can state the problem of local ridge approximation with active subspaces.

Problem 1 (Minimizers $(\mathcal{P}, r)$ of the ridge approximation error)**.** *Find the partition* $\mathcal{P}$ *of the domain* $\mathcal{X} \subset \mathbb{R}^n$ *and the local reduced dimension map* $r : \mathcal{P} \to \{1, \dots, n_r\}$, $n_r \ll n$*, such that the* L^2*-error between the objective function* f *and its local ridge approximation with active subspaces is minimized.*

$$E_{\boldsymbol{\mu}} \left[\|f - R_{AS}(r, f)\|^2\right] = \sum_{S_i \in \mathcal{P}} E_{\boldsymbol{\mu}} \left[\|f|_{S_i} - E_{\boldsymbol{\mu}_i}[f|P_{r(S_i),S_i}]\|^2\right]. \tag{2.11}$$

Assuming that the subspace Poincaré inequality [192] is valid also for $(f, \boldsymbol{\mu})$ restricted to the elements of the partition $\mathcal{P}$, a straightforward bound is obtained by applying the Poincaré inequality for every element of the partition

$$\begin{aligned} E_{\boldsymbol{\mu}} \left[\|f - R_{\mathrm{AS}}(r, f)\|^2\right] &= \sum_{S_i \in \mathcal{P}} E_{\boldsymbol{\mu}} \left[\|f|_{S_i} - E_{\boldsymbol{\mu}_i}[f|P_{r(S_i),S_i}]\|^2\right] \\ &\lesssim \sum_{S_i \in \mathcal{P}} E_{\boldsymbol{\mu}} \left[\|(Id - P_{r(S_i),S_i})\nabla f\|^2\right]. \end{aligned}$$

To obtain the previous upper bound, we made an assumption about the Poincaré subspace inequality that in general is not satisfied by any probability measure $\boldsymbol{\mu}$ chosen: the assumptions on the probability distributions $\{\boldsymbol{\mu}_i\}_{i=1}^d$ in section 2.2.1 have to be satisfied at each subdomain $\{S_i\}_{i=1}^d$.

For the moment we will consider the local reduced dimension map r constant and, in general, the codomain of r is a subset of $\{1, \dots, n_r\}$, $n_r \ll n$.

The previous bound suggests that a good indicator for refinement could be represented by the sum of the residual eigenvalues $\{\lambda_{S_i,j}\}_{j=r_{S_i}}^m$ of the local correlation matrices, for every $S_i \in \mathcal{P}$:

$$E_{\boldsymbol{\mu}} \left[\|f - R_{\mathrm{AS}}(r, f)\|^2\right] \lesssim \sum_{S_i \in \mathcal{P}} \sum_{j=r(S_i)+1}^{m} \lambda_{S_i,j}.$$

We also have the following immediate result that hints to indefinitely many successive refinements to lower the L^2-error ridge approximation error.

Remark 1 (Relationships between the upper bounds of consecutive refinements). Considering the sum over the number of refined clusters $cl \in \{1,\dots,d\}$ we have that

$$\begin{aligned}\int_{\mathcal{X}} \|(Id-P_r)\nabla f\|^2 \, d\boldsymbol{\mu} &= \sum_{cl=1}^{d} \int_{S_{cl}\subset\mathcal{X}} \|(Id-P_r)\nabla f\|^2 \, d\boldsymbol{\mu} \\ &\geq \sum_{cl=1}^{d} \int_{S_{cl}\subset\mathcal{X}} \|(Id-P_{r,cl})\nabla f\|^2 \, d\boldsymbol{\mu},\end{aligned} \tag{2.12}$$

since the projectors $\{P_{r,cl}\}_{cl\in\{1,\dots,d\}}$ are the minimizers of

$$P_{r,cl} = \underset{\substack{P^2=P, P=P^T,\\ \text{rank}(P)=r}}{\text{argmin}} \int_{S_{cl}\subset\mathcal{X}} \|(Id-P)\nabla f\|^2 \, d\boldsymbol{\mu}. \tag{2.13}$$

The RHS of eq. (2.12) can be used as indicator for refinement. We remark that since the refinements increase the decay of the eigenvalues in the RHS of eq. (2.12), the choice of the dimension of the active subspace may be shifted towards lower values to achieve further dimension reduction for the same accuracy, as we are going to show in the numerical experiments, in section 2.5.

Unfortunately, the minimizers of the ridge approximation error and of the upper bound are not generally the same:

$$\underset{\{P_{r(S_i),S_i}\}_{S_i\in\mathcal{P}}}{\text{argmin}} E_{\boldsymbol{\mu}}\left[\|f-R_{\text{AS}}(r,f)\|^2\right] \neq \underset{\{P_{r(S_i),S_i}\}_{S_i\in\mathcal{P}}}{\text{argmin}} \sum_{S_i\in\mathcal{P}} E_{\boldsymbol{\mu}}\left[\|(Id-P_{r(S_i),S_i})\nabla f\|^2\right].$$

There is a counterexample for the non-localized case in [281]. We start from this counterexample to show that, in general, the L^2-error of the local ridge approximation does not decrease between consequent refinements, even if the indicator from the RHS of eq. (2.12) does, as stated in the previous remark.

Corollary 1 (Counterexample for indefinite refinement as optimal clustering criterion)**.** *Let $\mathcal{P} = \{A,B,C\}$ be a partition of $\mathcal{X} = [-1,1]^2$ such that $A = [-1,\varepsilon]\times[-1,1]$, $B = [-\varepsilon,\varepsilon]\times[-1,1]$, and $C = [\varepsilon,1]\times[-1,1]$. Let $\boldsymbol{\mu}$ be the uniform probability distribution on $\mathcal{X}$. The objective function we want to approximate is*

$$f : \mathcal{X}\subset\mathbb{R}^2 \to \mathbb{R}, \quad f = \begin{cases} x_1+\varepsilon, & \mathbf{x}\in A, \\ x_1(x_1+\varepsilon)(x_1-\varepsilon)\cos(\omega x_2), & \mathbf{x}\in B, \\ x_1-\varepsilon, & \mathbf{x}\in C, \end{cases} \tag{2.14}$$

with local reduced dimension map $r(A) = r(B) = r(C) = 1$. There exist $\varepsilon > 0, \omega > 0$, such that

$$E_{\boldsymbol{\mu}}\left[\|f-R_{AS}(r,f,\boldsymbol{\mu})\|^2\right] \geq E_{\boldsymbol{\mu}}\left[\|f-E_{\boldsymbol{\mu}}\left[f|P_{1,\mathcal{X}}\right]\|^2\right],$$

where $P_{1,\mathcal{X}}$ is the optimal projector on the whole domain $\mathcal{X}$ with one-dimensional active subspace.

Proof. Let us use the notation $h_1(x_1) := x_1(x_1+\varepsilon)(x_1-\varepsilon)$, and $h_2(x_2) := \cos(\omega x_2)$, it can be shown that

$$\begin{aligned} E_\mu[\nabla f \otimes \nabla f] &= \int_B \begin{pmatrix} (h_1')^2(h_2)^2 & h_1 h_1' h_2 h_2' \\ h_1 h_1' h_2 h_2' & (h_1)^2(h_2')^2 \end{pmatrix} d\mu(\mathbf{x}) + \mu(A\cup C)\cdot \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{2}{5}\varepsilon^5\left(1+\frac{\sin(2\omega)}{2\omega}\right) & 0 \\ 0 & \frac{4}{105}\omega^2\varepsilon^7\left(1-\frac{\cos(2\omega)}{2\omega}\right) \end{pmatrix} + \mu(A\cup C)\cdot \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \end{aligned}$$

thus, since we are considering a one-dimensional active subspace, the active eigenvector belongs to the set $\{(1,0),(0,1)\}$. Similarly, we evaluate

$$\begin{aligned} E_{\mu_B}[\nabla f|_B \otimes \nabla f|_B] &= \begin{pmatrix} \frac{8}{5}\varepsilon^4\left(1+\frac{\sin(2\omega)}{2\omega}\right) & 0 \\ 0 & \frac{16}{105}\omega^2\varepsilon^6\left(1-\frac{\cos(2\omega)}{2\omega}\right) \end{pmatrix}, \\ E_{\mu_A}[\nabla f|_A \otimes \nabla f|_A] = E_{\mu_C}[\nabla f|_C \otimes \nabla f|_C] &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \end{aligned}$$

and conclude that there exist $\varepsilon > 0, \omega > 0$ such that:

$$\frac{2}{5}\varepsilon^5\left(1+\frac{\sin(2\omega)}{2\omega}\right) + 4(1-\varepsilon) \geq \frac{4}{105}\omega^2\varepsilon^7\left(1-\frac{\cos(2\omega)}{2\omega}\right), \tag{2.15}$$

$$\frac{8}{5}\varepsilon^4\left(1+\frac{\sin(2\omega)}{2\omega}\right) \leq \frac{16}{105}\omega^2\varepsilon^6\left(1-\frac{\cos(2\omega)}{2\omega}\right), \tag{2.16}$$

for example $\varepsilon \sim 10^{-2}$, $\omega \sim 10^4$ (approximately $10\varepsilon^{-2} \leq \omega^2 \leq 10\varepsilon^{-7}$). In this way, using the notations of definition 2, we have

$$P_{1,\mathcal{X}} = e_1 \otimes e_1, \quad P_{1,A} = P_{1,C} = e_1 \otimes e_1, \quad P_{1,B} = e_2 \otimes e_2,$$

and it follows that

$$E_\mu\left[\|f - R_{AS}(r,f)\|^2\right] = E_\mu\left[f^2|_B\right] = (1/\mu(\mathcal{X}))\|h_1\|^2_{L^2(\mathcal{X},\lambda)}\|h_2\|^2_{L^2(\mathcal{X},\lambda)},$$

$$\begin{aligned} E_\mu\left[\|f - E_\mu[f|P_r]\|^2\right] &= (1/\mu(\mathcal{X}))\|h_1\|^2_{L^2(\mathcal{X},\lambda)}\|h_2 - (1/\mu(\mathcal{X}))\int h_2 dx_2\|^2_{L^2(\mathcal{X},\lambda)} \\ &= (1/\mu(\mathcal{X}))\|h_1\|^2_{L^2(\mathcal{X},\lambda)}\left(\|h_2\|^2_{L^2(\mathcal{X},\lambda)} - \frac{7}{16}\left(\int h_2 dx_2\right)^2\right), \end{aligned}$$

where λ is the Lebesgue measure. □

The heuristics behind the previous proof rests on the fact that ridge approximation with active subspaces performs poorly when the objective function has a high variation. The counterexample is valid whenever the global projector $P_{1,\mathcal{X}}$ is the minimizer of a local L^2 ridge approximation error for which the minimizer of the gradient-based indicator in eq. (2.12) does not coincide. This leaves us with an indicator in eq. (2.12) that does not guarantee a non-increasing L^2-error decay for subsequent refinements, but is nonetheless useful in practice.

We conclude the section with some remarks about the response surface design through the ridge approximation with active subspaces.

Remark 2 (Approximation of the optimal profile). In practice, we do not consider the optimal profile $h(\mathbf{y}) = E_{\boldsymbol{\mu}}\left[f|\sigma(P_r)\right](\mathbf{y})$, but we employ the approximation $h(\mathbf{y}) = f(\mathbf{y}) = f(P_r\mathbf{x})$. The reason lies on the fact that to approximate the optimal profile at the values $\{\mathbf{y}_i\}_i$, additional samples from the conditional distribution $p(z|\mathbf{y}_i = P_r\mathbf{x})$ must be obtained; even if the accuracy of the ridge approximation could benefit from it, this is not always possible in practice because of the difficulty to sample from the conditional distribution or because of computational budget constraints.

If the data is split in training, validation, and test set, the local R^2 score on the validation set can be used as indicator for refinement.

Remark 3 (Estimator based on local R^2 scores). The R^2 score of a single cluster can be written with respect to the R^2 scores $\{R_l^2\}_{l\in\{1,\dots,d\}}$ relative to the clusters of the subsequent refinement. Let the sum be over the refinement clusters $l \in \{1,\dots,d\}$, we have

$$\begin{aligned} R^2 &= 1 - \frac{E[\|f - E[f|P_r]\|^2]}{\mathrm{Var}(f)} = 1 - \sum_{l=1}^{d} \frac{E[\|f|_{S_l} - E[f|P_{r,l}]\|^2]}{\mathrm{Var}(f)} \\ &= 1 - \sum_{l=1}^{d} \frac{\mathrm{Var}(f|_{S_l})}{\mathrm{Var}(f)} \cdot \frac{E[\|f|_{S_l} - E[f|P_{r,l}]\|^2]}{\mathrm{Var}(f|_{S_l})} = 1 - \sum_{l=1}^{d} \frac{\mathrm{Var}(f|_{S_l})}{\mathrm{Var}(f)} \cdot (1 - R_l^2), \end{aligned} \tag{2.17}$$

which, substituting with the empirical variance, becomes

$$R^2_{\mathrm{emp}} = 1 - \sum_{l=1}^{d} \frac{\mathrm{Var}_{\mathrm{emp}}(f|_{S_l})}{\mathrm{Var}_{\mathrm{emp}}(f)} \cdot (1 - R^2_{\mathrm{emp};l}) \cdot \frac{N_l - 1}{N - 1}, \tag{2.18}$$

where $R^2_{\mathrm{emp};l}$ is the empirical local R^2 score relative to cluster number l. The definition can be extended for component-wise vector-valued objective functions f. The numerical results shown in section 2.5 consider the mean R^2 score along the components when the output is vector-valued.

In practice, every expectation is approximated with simple Monte Carlo, and without the number of training samples increasing, the confidence on the approximation is lower and lower, the more the domain is refined. This is taken into consideration while clustering, fixing a minimum number of samples per cluster for example.

The section 2.2.2 clarifies the link between the number of Monte Carlo samples, the numerical method chosen for the discretization of the integral $E_{\boldsymbol{\mu}}\left[\nabla f \otimes \nabla f\right]$, and the approximation of the active subspace. For example for deterministic models, one could employ the more efficient Sobol sequence or a Latin hypercube sampling; if f is more regular and the parameter space dimension is not too high one could employ tensor product Gauss quadrature rule. See for example [241].

Before introducing the clustering algorithms we will employ, we specify that the partition $\mathcal{P} = \{S_i\}_{i\in\{1,\dots,d\}}$ is defined by the decision boundaries of the clustering algorithm chosen.

2.3.2 K-means clustering

We recall the K-means clustering algorithm. Let $\{x_i\}_{i=1}^N$ be a set of N samples in $\mathbb{R}^{N_F}$, where N_F denotes the number of features. The K-means algorithm divides this set into K disjoint clusters $S = \{S_j\}_{j=1}^K$, with $S_l \cap S_m = \emptyset$ for $1 \leq l,m \leq K$ and $l \neq m$. The partitioning quality is assessed by a function that aims for high intracluster similarity and low intercluster similarity. For K-means, this is done by minimizing the total within-cluster sum-of-squares criterion W_T, which reads as

$$W_T(S) := \sum_{j=1}^K W(S_j) = \sum_{j=1}^K \sum_{x_i\in S_j} \|x_i - c_j\|_{L^2}^2, \tag{2.19}$$

where c_j is the centroid describing the cluster S_j. A centroid of a cluster is defined as the mean of all the points included in that cluster. This means that the centroids are, in general, different from the samples x_i.

K-means is sensitive to outliers since they can distort the mean value of a cluster and thus affecting the assignment of the rest of the data.

2.3.3 K-medoids clustering with active subspaces-based metric

In order to overcome some limitations of the K-means algorithm, such as sensitivity to outliers, we can use K-medoids clustering technique [144, 194, 229, 176]. It uses an actual sample as cluster representative (i.e. medoid) instead of the mean of the samples within the cluster.

Following the notation introduced in the previous section, let m_j be the medoid describing the cluster S_j. The partitioning method is performed by minimizing the sum of the dissimilarities between the samples within a cluster and the corresponding medoid. To this end an absolute-error criterion E is used, which reads as

$$E(S) := \sum_{j=1}^K E(S_j) = \sum_{j=1}^K \sum_{x_i\in S_j} \|x_i - m_j\|. \tag{2.20}$$

By looking at the formula above it is clear that the use of a data point to represent each cluster's center allows the use of any distance metric for clustering. We remark that the choice of the Euclidean distance does not produce the same results as K-means because of the different references representing the clusters.

We propose a new supervised distance metric inspired by the global active subspace of the function f we want to approximate. We define a scaled L^2 norm using the eigenpairs of the second moment matrix of ∇f, which is the matrix from which we calculate the global active subspace:

$$\|x_i - x_j\|_\Lambda = \sqrt{(x_i - x_j)^T \mathbf{W}\Lambda^2\mathbf{W}^T(x_i - x_j)}, \tag{2.21}$$

where Λ stands for the diagonal matrix with entries of the eigenvalues of eq. (2.1), and $\mathbf{W}$ is the eigenvectors' matrix from the decomposition of the covariance matrix. As we are going to show in section 2.5 this new metric allows a better partitioning both for regression and classification tasks by exploiting both global and local information. For insights about the heuristics behind it, we refer to remark 5.

To actually find the medoids the partitioning around medoids (PAM) algorithm [144] is used. It uses a greedy approach after the initial selection of the medoids, also called representative objects. They are changed with a non-representative object, i.e. one of the remaining samples, if it would improve the clustering quality. This iterative process of replacing the medoids with other objects continues until the quality of the resulting clustering cannot be improved by any replacement. algorithm 1 illustrates this approach with pseudocode.

Algorithm 1: K-medoids algorithm with AS metric.

```
input  : set of samples $\{x_i\}_{i=1}^N \in \mathbb{R}^{N_F}$,
         number of clusters $K$,
         distance metric $d$ defined in 2.21
output : set of clusters $S = \{S_j\}_{j=1}^K$

1 select initial cluster medoids.
2 repeat
3     assign each sample to its closest medoid using the distance metric $d$.
4     randomly select $K$ non-representative objects.
5     swap the medoids with the newly selected objects by minimizing 2.20.
6 until clustering quality converges.
```

2.3.4 Hierarchical top-down clustering

In this section, we present hierarchical top-down clustering [144, 182], and exploit the additional information from the active subspace, as done for K-medoids. In the following sections, we refer to this technique with the acronym HAS.

In top-down hierarchical clustering, at each iteration the considered clusters, starting from the whole dataset, are split further and further based on some refinement criterion, until convergence. A nice feature of hierarchical clustering algorithms, with respect to K-means and K-medoids, is that the number of clusters can be omitted. Moreover, by stopping at the first refinement and forcing the total number of clusters to be the maximum number of clusters specified, HAS can be seen as a generalization of the previous methods: for this reason, we wanted to make the implementation consistent with K-means and K-medoids with AS induced metric as close as possible, as shown in the numerical results in section 2.5.

Pushing further the potential of clustering algorithms applied to local dimension reduction in the parameter space, HAS is a versatile clustering method that takes into account the variability of the AS dimension along the parameter space. The price paid for this is the overhead represented by the tuning of some hyperparameters introduced later in section 2.3.4.

A schematic representation of the algorithm of top-down clustering is reported in algorithm 2. The design is straightforward and it employs a tree data structure that assigns at each node a possible clustering of the whole dataset: consequent refinements are represented by children nodes down until the leaves of the tree, that represent the final clusters.

Remark 4 (Normalization of the clusters at each refinement iteration). Each cluster, at every refinement step, is normalized uniformly along dimensions onto the hyper-cube domain $[-1,1]^n$, even if the subdomain identified by the cluster is not a hyperrectangle. Another possible choice for normalization is standardization, centering the samples with their mean and dividing them by their standard deviation.

Algorithm 2: Hierarchical top-down algorithm.

```
input  : set of samples $\mathbf{S} = \{x_i\}_{i=1}^N \in \mathbb{R}^{N_F}$,
         number of clusters per tree refinement level $\mathbf{K}$,
         range of number of children $\{\mathbf{n}_{\mathbf{min}}^{\mathbf{child}}, \mathbf{n}_{\mathbf{max}}^{\mathbf{child}}\}$,
         minimum number of elements in a cluster $\mathbf{n_{el}}$,
         indicator for refinement $\mathbf{I}$,
         distance metric $\mathbf{d}$,
         minimum and maximum dimension of the active subspace $\mathbf{r_{min}}, \mathbf{r_{max}}$,
         score tolerance $\boldsymbol{\varepsilon}$.
output : refinement tree $T = S(S_1(S_{k+1}(\dots,(S_{\text{leaf},1}),\dots),\dots),\dots,S_k(\dots))$, with
         $k \in \{\mathbf{n}_{\mathbf{min}}^{\mathbf{child}}, \dots, \mathbf{n}_{\mathbf{max}}^{\mathbf{child}}\}$.
1 add the initial cluster $S$ to FIFO queue $q = \{S\}$.
2 while $q \neq \varnothing$ do
3     take $\mathbf{S}_j$, the first element from queue $q$.
4     apply the refinement function in 3 to $\mathbf{S}_j$ in order to get the outputs $\{S_i\}_i$.
5     add $\{\mathbf{S}_i\}_i$ to the queue $q$.
6     if (the score tolerance $\varepsilon$ is reached
7     or the minimum number of elements in a cluster $n_{el}$ is reached in the next iteration
8     or the maximum number of clusters $\mathbf{K}$ is reached in the next iteration) then
9         break
```

The procedure depends on many parameters that have to be tuned for the specific case or depend *a priori* on the application considered: the maximum number of clusters, the minimum and maximum number of children nodes, the tolerance for the score on the whole domain, the minimum and

Algorithm 3: Refinement function.

```
input  : cluster $\mathbf{S} = \{x_i\}_{i=1}^N \in \mathbb{R}^{N_F}$,
         number of clusters per tree refinement level $\mathbf{K}$,
         range of number of children $\{\mathbf{n}_{\mathbf{min}}^{\mathbf{child}}, \mathbf{n}_{\mathbf{max}}^{\mathbf{child}}\}$,
         minimum number of elements in a cluster $\mathbf{n}_{\mathbf{el}}$,
         indicator for refinement $\mathbf{I}$,
         distance metric $\mathbf{d}$,
         minimum and maximum dimension of the active subspace $\mathbf{r}_{\mathbf{min}}, \mathbf{r}_{\mathbf{max}}$.
output : $\{S_j\}_{j=1}^{\mathbf{n}^{\mathbf{child}}}$, the children of cluster $\mathbf{S}$, with $\mathbf{n}^{\mathbf{child}} \in \{\mathbf{n}_{\mathbf{min}}^{\mathbf{child}}, \dots, \mathbf{n}_{\mathbf{max}}^{\mathbf{child}}\}$.
1 set best score to $b = 0$.
2 foreach $\mathbf{n}^{\mathbf{child}}$ from $\mathbf{n}_{\mathbf{min}}^{\mathbf{child}}$ to $\mathbf{n}_{\mathbf{max}}^{\mathbf{child}}$ do
3     apply the chosen clustering algorithm (e.g. K-medoids) with $\mathbf{n}^{\mathbf{child}}$ clusters and metric
        $\mathbf{d}$ to obtain the clusters $\{S_j\}_j^{\mathbf{n}^{\mathbf{child}}}$.
4     evaluate the estimator of the error $\mathbf{I}$ for the refinement $\{S_j\}_j$, taking also into account
        the minimum and maximum reduced dimensions $\mathbf{r}_{\mathbf{min}}, \mathbf{r}_{\mathbf{max}}$.
5     if $\mathbf{I} > b$ and the minimum number of elements $\mathbf{n}_{\mathbf{el}}$ is not reached
6     and the maximum number of clusters $\mathbf{K}$ is not reached then
7         save the best refinement $\{S_j\}_j$ and update the best score $b$.
```

maximum dimension of the active subspace, and the minimum number of elements (el) of each cluster (usually $el > r$, where r is the local AS dimension).

More importantly, the method is versatile for the choice of clustering criterion, indicator for refinement, and regression method. In the following sections we will consider K-means and K-medoids with the active subspaces distance as clustering criterion (see section 2.3.3), but other clustering algorithms could in principle be applied at each refinement.

Remark 5 (Heuristics behind the choice of the active subspaces metric for K-medoids). Having in mind that the optimal profile $h(\mathbf{y}) = E_{\boldsymbol{\mu}_i}[f|P_{r(S_i),i}](\mathbf{y})$ from definition 2 is approximated as $h(\mathbf{y}) = f(\mathbf{y}) = f(P_r\mathbf{x})$ as reported in remark 2, we can argue that clustering with the AS metric from eq. (2.21) is effective since, for this choice of the metric, the clusters tend to form transversally with respect to the active subspace directions. This is because the metric weights the components with higher eigenvalues more. So clustering with this metric reduces heuristically also the approximation error induced by the choice of the non-optimal profile.

Other clustering criterions employed must satisfy the subspace Poincaré inequality for each cluster. Regarding the regression method we employ Gaussian process regression with RBF-ARD kernel [271]. The procedure for response surface design with Gaussian processes and ridge approximation with active subspaces can be found in [60, 221]. As for the indicator for refinement, the local R^2 score in remark 3 is employed to measure the accuracy of the ridge approximation against a validation dataset and the estimator from the RHS of eq. (2.12) is used to determine the dimension of the active subspace of each cluster.

Regarding the complexity of the algorithm, for each refinement, considering an intermediate cluster of K elements, the most expensive tasks are the active subspace evaluation $O((K/m)np^2 + (K/m)n^2p + n^3)$ (the first two costs refer to matrix multiplications, while the third to eigendecomposition), the clustering algorithm, for example K-medoids with AS distance $O(K(K-m)^2)$, and the Gaussian process regression $O((K/m)^3p^3)$, where p is the dimension of the outputs and m is the minimum number of children clusters. With M we denote the maximum number of children clusters. In the worst case the height of the refinement tree is $l = \log_m N/el$ where el is the minimum number of elements per cluster. In table 2.1 we report the computational complexity of the hierarchical top-down clustering algorithm. We report the costs divided by level of refinement.

Table 2.1 Computational complexity of hierarchical top-down clustering.

Step	Cost	Description
Root	$O(Nnp^2 + Nn^2p + n^3)$	AS
	$O(N^3p^3)$	GPR
First refinement:	$O(N(N-k)^2)$	K-medoids
k from m to M	$O((N/k)np^2 + (N/k)n^2p + n^3)$	AS
	$O((N/k)^3p^3)$	GPR
Intermediate refinements	-	-
Last refinement:	$O((N/k^{l-1})((N/k^{l-1}) - k)^2)$	K-medoids
k from m to M	$O((N/k^l)np^2 + (N/k^l)n^2p + n^3)$	AS
for each one of the m^{l-1} clusters	$O((N/k^l)^3p^3)$	GPR

2.4 Classification with local active subspace dimension

A poor design of the parameter space could add an avoidable complexity to the surrogate modelling algorithms. Often, in practical applications, each parameter range is chosen independently with respect to the others. Then, it is the responsibility of the surrogate modelling procedure to disentangle the correlations among the parameters. However, in this way, looking at the response surface from parameters to outputs, regions that present different degrees of correlation are treated indistinctly. In this matter, a good practice is to study as a preprocessing step some sensitivity measures, like the total Sobol indexes [241] among groups of parameters and split the parameter space accordingly in order to avoid the use of more expensive surrogate modelling techniques later. Sobol indices or the global active subspace sensitivity scores give summary statistics on the whole domain. So in general, one could study the parameter space more in detail, classifying nonlinearly regions with respect to the complexity of the response surface, if there are enough samples to perform such studies.

We introduce an effective approach to tackle the problem of classification of the parameter space with respect to a local active subspace information. With the latter we mean two possible alternatives.

Definition 3 (Local active subspace dimension). *Given a threshold $\varepsilon > 0$, the pairs of inputs and gradients $\{(\mathbf{X}_i, \mathbf{dY}_i)\}_i$ associated to an objective function of interest $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$, the size of the neighborhood of sample points to consider $N \geq n$, and a subsampling parameter $p \in \mathbb{N},\ p \leq N$, the local active subspace dimension r_i associated to a sample point $\mathbf{X}_i \in \mathcal{X}$ is the positive integer*

$$r_i = \underset{1 \leq r \leq p}{\operatorname{argmin}} \left\{ tr\left((Id - P_r) \left(\frac{1}{p} \sum_{i \in J} \mathbf{dY}_i \otimes \mathbf{dY}_i \right) (Id - P_r) \right) \leq \varepsilon \;\middle|\; J \in C(N,p) \right\},$$

where $C(N,p)$ is the set of combinations without repetition of the N elements of the Euclidean neighborhood of $\mathbf{X}_i$ with class p.

Definition 4 (Local active subspace). *Given the pairs of inputs and gradients $\{(\mathbf{X}_i, \mathbf{dY}_i)\}_i$ associated with an objective function of interest $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$, the size of the neighborhood of sample points to consider $N \geq n$, and a fixed dimension $p \in \mathbb{N},\ 1 \leq p \leq N$, the local active subspace W_i associated to a sample point $\mathbf{X}_i \in \mathcal{X}$ is the matrix of the first p eigenvectors of the spectral decomposition of*

$$(Id - P_r) \left(\frac{1}{p} \sum_{i \in U} \mathbf{dY}_i \otimes \mathbf{dY}_i \right) (Id - P_r), \tag{2.22}$$

where U is the neighborhood of sample points of $\mathbf{X}_i$ with respect to the Euclidean distance. In practice, we choose p close to the global active subspace dimension. The pairs $\{(\mathbf{X}_i, W_i)\}_i$ can be thought as a discrete vector bundle of rank p and $\{W_i\}_i$ can be thought as a subset of points of the Grassmannian $Gr(N,p)$.

Starting from the pairs of inputs-gradients $\{(\mathbf{X}_i, \mathbf{dY}_i)\}_i$, the procedure follows these steps:

1. Each parameter sample is enriched with the additional feature corresponding to the local active subspace dimension from definition 3 or the local active subspace from definition 4, represented by the variable $\mathbf{Z}$.

2. Each sample $\mathbf{X}_i$ is labelled with an integer l_i that will be used as classification label in the next step. To label the pairs $\{(\mathbf{X}_i, \mathbf{Z}_i)\}_i$ we selected K-medoids with the Grassmannian metric

$$d((X_i,Z_i),(X_j,Z_j)) = \|Z_i - Z_j\|_F, \tag{2.23}$$

where $\|\cdot\|_F$ is the Frobenius distance, in case $\mathbf{Z}_i$ represents the local active subspace or spectral clustering in case $\mathbf{Z}_i$ is the local active subspace dimension. In the last case, the labels correspond to the connected components of the graph built on the nodes $\{(\mathbf{X}_i, \mathbf{Z}_i)\}_i$ with adjacency list corresponding to the nearest nodes with respect to the distance

$$d((X_i,Z_i),(X_j,Z_j)) = \begin{cases} \infty, & Z_i \neq Z_j \\ \|X_i - X_j\|, & Z_i = Z_j \end{cases}, \tag{2.24}$$

where $\|\cdot\|$ is the Euclidean metric in $\mathbb{R}^n$. The connected components are obtained from the eigenvectors associated with the eigenvalue 0 of the discrete Laplacian of the graph [182].

3. A classification method is applied to the inputs-labels pairs $\{(\mathbf{X}_i, l_i)\}_i$. Generally, for our relatively simple applications, we apply a multilayer perceptron with 1000 hidden nodes and 2 layers.

Remark 6 (Grassmann distance). In general regarding the definition 4, the dimension p could be varying among samples $\mathbf{X}_i$ and one could use a more general distance with respect to the one from eq. (2.23) that can have as arguments two vectorial subspaces of possibly different and arbitrary large dimensions.

Remark 7 (Gradient-free active subspace). In general, both the response surface design and the classification procedure above can be carried out from the pairs $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_i$ of inputs, outputs instead of the sets $\{(\mathbf{X}_i, \mathbf{dY}_i)\}_i$ of inputs, gradients. In fact, the gradients $\{\mathbf{dY}_i\}$ can be approximated in many different ways [60] from $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_i$. In the numerical results in section 2.5 when the gradients are not available they are approximated with the gradients of the local one-dimensional polynomial regression built on top of the neighboring samples.

Algorithm 4: Classification with local features from the active subspaces information.

```
input  : inputs-gradients pairs $\{(\mathbf{X}_i, \mathbf{dY}_i)\}_{i\in I}$ as training dataset with index set $I$,
         local features based on AS information $\{\mathbf{Z}_i\}_{i\in I}$, see 3 and 4,
         labelling method based on the distance $d$ from 2.23 or from 2.24,
         classification method that takes as input the inputs-labels pairs $\{(\mathbf{X}_i, l_i)\}_{i\in I}$
output : predictor for new test inputs and classes of the training dataset.

1 foreach $i \in I$ do
2     evaluate the feature $\mathbf{Z}_i$ from $(\mathbf{X}_i, \mathbf{dY}_i)$ and the neighbouring points of $\mathbf{X}_i$.
3 initialize the $|I| \times |I|$ distance matrix $M$ associated to the pairs $\{(\mathbf{X}_i, \mathbf{Z}_i)\}_{i\in I}$.
4 foreach $i \in I$ do
5     foreach $i \leq j \in I$ do
6         $M(i, j) = d((\mathbf{X}_i, \mathbf{Z}_i), (\mathbf{X}_j, \mathbf{Z}_j))$
7 use the labelling method with input $M$, to assign a label $l_i$ for each pair $(\mathbf{X}_i, \mathbf{Z}_i)$.
8 train the classification method with the inputs-labels training pairs $\{(\mathbf{X}_i, l_i)\}_{i\in I}$.
```

2.5 Numerical results

In this section, we apply the proposed localized AS method to some datasets of increasing complexity. We emphasize that the complexity is not only defined by the number of parameters but also by the intrinsic dimensionality of the problem. We compare the clustering techniques presented in section 2.3, and we show how the active subspaces-based distance metric outperforms the Euclidean one for those functions which present a global lower intrinsic dimensionality. We remark that for hierarchical top-down clustering, we can use both metrics, and we always show the best case for the specific dataset.

We start with a two-dimensional example for which we can plot the clusters and the regressions, and compare the different techniques. Even if it is not a case for which one should use parameter space dimensionality reduction we think it could be very useful for the reader to understand also visually all the proposed techniques. For the higher dimensional examples, we compare the accuracy of the methods in terms of R^2 score and classification performance. All the computations regarding AS are done with the open-source Python package[1] called `ATHENA` [218], for the classification algorithms we use the scikit-learn package [41], and for the Gaussian process regression GPy [98].

We suppose the domain $\mathcal{X}$ to be a n-dimensional hyperrectangle. we are going to rescale the input parameters $\mathbf{X}$ to $[-1,1]^n$.

2.5.1 Some illustrative two-dimensional examples

We start by presenting two two-dimensional test cases to show every aspect of the methodology together with illustrative plots. First we analyze a case where a global active subspace, even if present, does not provide a regression accurate enough along the active direction, in section 2.5.1. Then we consider a radial symmetric function for which, by construction, an AS does not exist, in section 2.5.1, and the use of K-means is instead preferable since we cannot exploit a privileged direction in the input domain.

Quartic function

Let us consider the following two-dimensional quartic function $f(\mathbf{x}) = x_1^4 - x_2^4$, with $\mathbf{x} = (x_1, x_2) \in [0,1]^2$. In fig. 2.1 we can see the contour plot of the function, the active subspace direction — translated for illustrative reasons — and the corresponding sufficient summary plot of the global active subspace, computed using 400 uniformly distributed samples. With sufficient summary plot we intend $f(\mathbf{x})$ plotted against the input parameters projected onto the active subspace, that is $W_1^T\mathbf{x}$. It is clear how, in this case, a univariate regression does not produce any useful prediction capability.

Let us apply the clustering techniques introduced in the previous sections fixing the number of clusters to 4. In fig. 2.2 we can clearly see how the supervised distance metric in eq. (2.21) acts in dividing the input parameters. On the left panel, we apply K-means which clusters the data into 4 uniform quadrants, while in the middle and right panels we have K-medoids and hierarchical top-down, respectively, with a subdivision aligned with the global AS. We notice that for this simple case, the new metric induces an identical clustering of the data. In fig. 2.3 we plotted the sufficient summary plots for each of the clusters individuated by K-medoids or hierarchical top-down in fig. 2.2.

[1] Available at https://github.com/mathLab/ATHENA/.

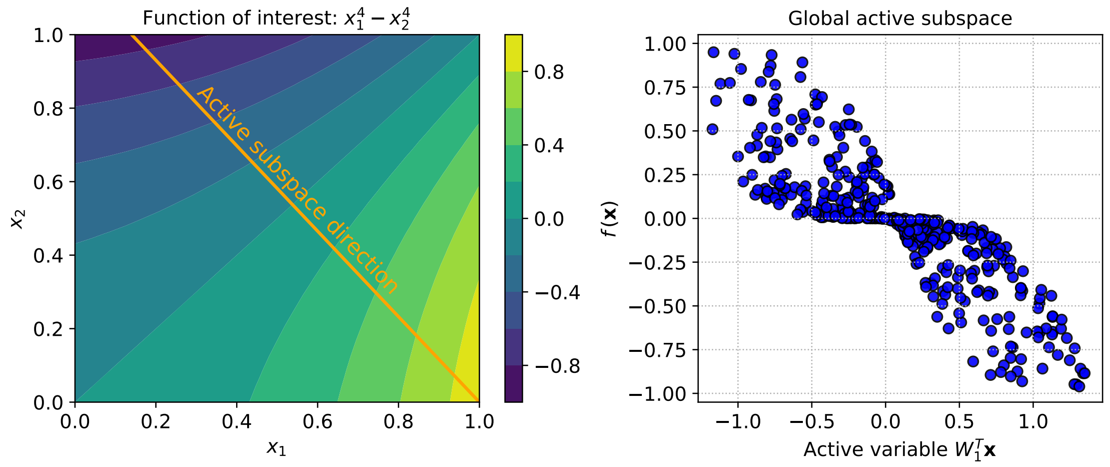


Fig. 2.1 On the left panel the contour plot of the quartic function and in orange the global active subspace direction. On the right panel the sufficient summary plot resulting in projecting the data onto the global AS.

By using a single univariate regression for each cluster the R^2 score improves a lot with respect to a global approach (see right panel of fig. 2.1).

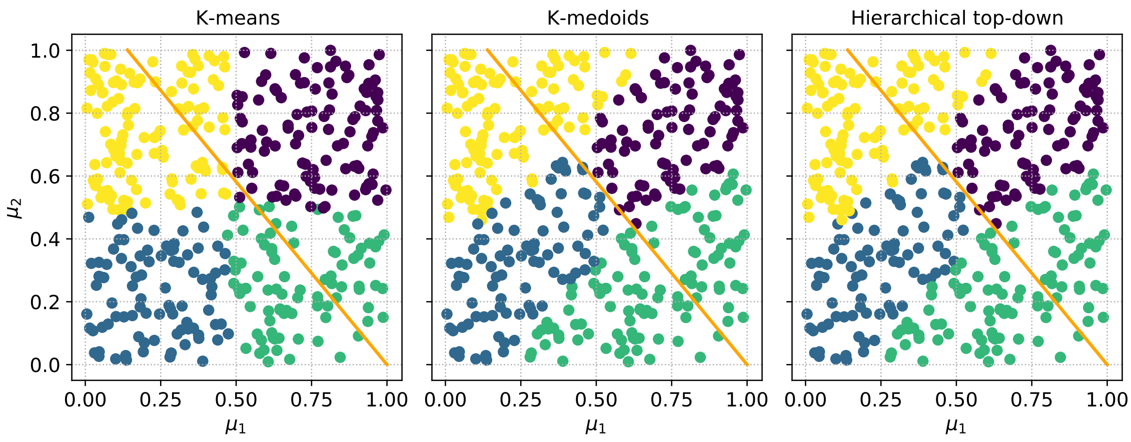


Fig. 2.2 Comparison between the different clusters obtained by K-means (on the left), K-medoids (middle panel), and hierarchical top-down (on the right) with AS induced distance metric defined in eq. (2.21) for the quartic test function. In orange the global active subspace direction. Every cluster is depicted in a different color.

We can also compare the R^2 scores for all the methods, using a test dataset of 600 samples. In fig. 2.4 we report the scores for K-means, K-medoids and for hierarchical top-down with AS-based distance metric. The score for the global AS, which is 0.78, is not reported in fig. 2.4 for illustrative

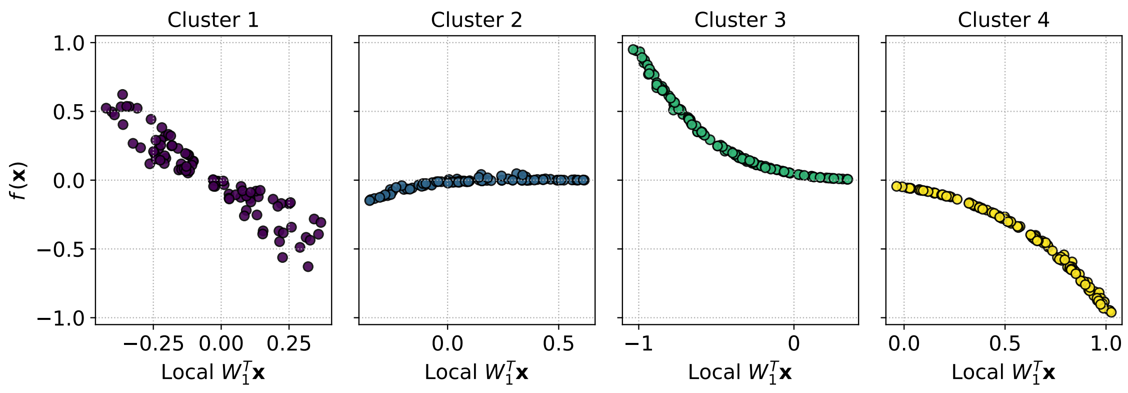


Fig. 2.3 Local sufficient summary plots for the 4 clusters individuated by K-medoids or hierarchical top-down in fig. 2.2 (colors correspond).

reasons. The results are very similar due to the relatively simple test case, but we can see that even with 2 clusters the gain in accuracy is around 23% using the metric in eq. (2.21).

The hierarchical top-down clustering method was run with the following hyperparameters: the total number of clusters is increasing from 2 to 10, the minimum number of children equal to the maximum number of children equal to 3, uniform normalization of the clusters, the minimum size of each cluster is 10 elements, the clustering method is K-medoids with AS distance, the maximum active subspace dimension is 1.

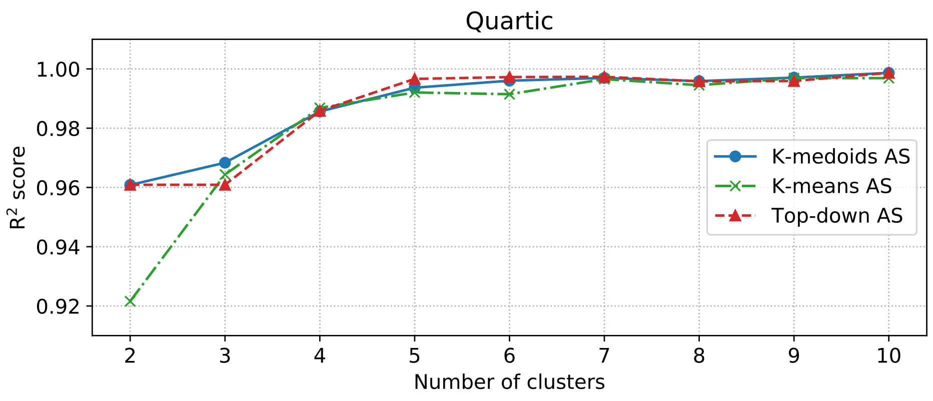


Fig. 2.4 R^2 scores comparison between local versions varying the number of clusters for the quartic function. Global AS has a score equal to 0.78.

Then we want to increase the accuracy of the regression for a fixed number of clusters equal to 3, loosing in some regions the reduction in the parameter space. Starting from the clustering with hierarchical top-down and 3 clusters of dimension 1, the AS dimension of each of the 3 clusters is increased if the threshold of 0.95 on the local R^2 score is not met. In general, the local R^2 score is

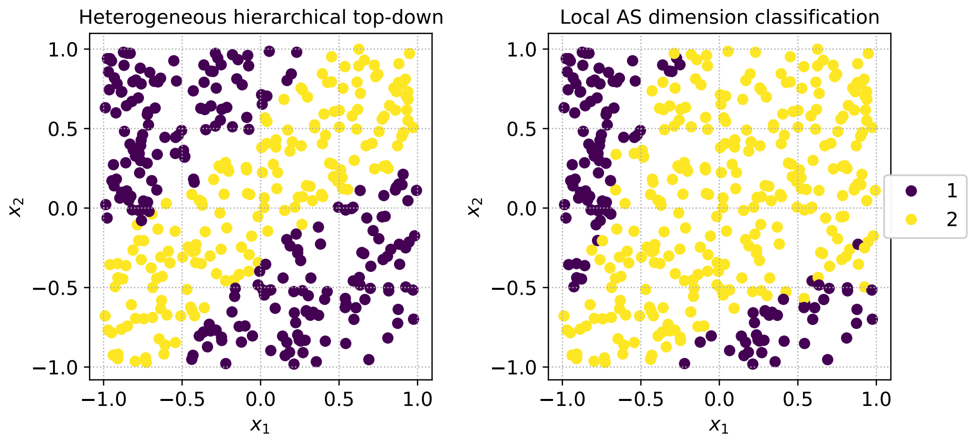


Fig. 2.5 On the left panel the hierarchical top-down clustering with heterogeneous AS dimension and R^2 score equal to 1. On the right panel the labels of the local AS dimension from definition 3.

evaluated on a validation set, for which predictions from the local response surfaces are obtained, after each validation sample is classified into one of the 3 clusters.

The 3 clusters are reported in fig. 2.5 on the left. The R^2 score on the test set is 1, instead of around 0.97 from fig. 2.4. To obtain this result, the central cluster AS dimension is increased from 1 to 2. We compare the clustering with respect to the classification of the local AS dimension with algorithm 4 using as features the local AS dimension as defined in definition 3, on the right of fig. 2.5. Actually, algorithm 4 is stopped after the plotted labels are obtained as the connected components of the underlying graph to which spectral clustering is applied: no classification method is employed, yet. It can be seen that hierarchical top-down clustering with heterogeneous AS dimension is more efficient with respect to the classes of algorithm 4, regarding the number of samples associated with a response surface of dimension 2.

Radial symmetric cosine

This example addresses the case for which an active subspace is not present. This is due to the fact that there are no preferred directions in the input domain since the function f has radial symmetry. For this case, the exploitation of the supervised distance metric does not provide any significant gain and K-means clustering works better on average since it does not use the global AS structure. The model function we consider is $f(\mathbf{x}) = \cos(\|\mathbf{x}\|^2)$, with $\mathbf{x} \in [-3,3]^2$.

In fig. 2.6 we compare the R^2 scores for K-means, K-medoids with AS-based metric, and hierarchical top-down with Euclidean metric. We used 500 training samples and 500 test samples. We see K-medoids has not a clear behavior with respect to the number of clusters, while the other methods present a monotonic trend and better results on average, especially K-means. On the other hand, local models improve the accuracy considerably, even for a small number of clusters, with respect to a global model. In this case, the specifics of hierarchical top-down clustering are: the

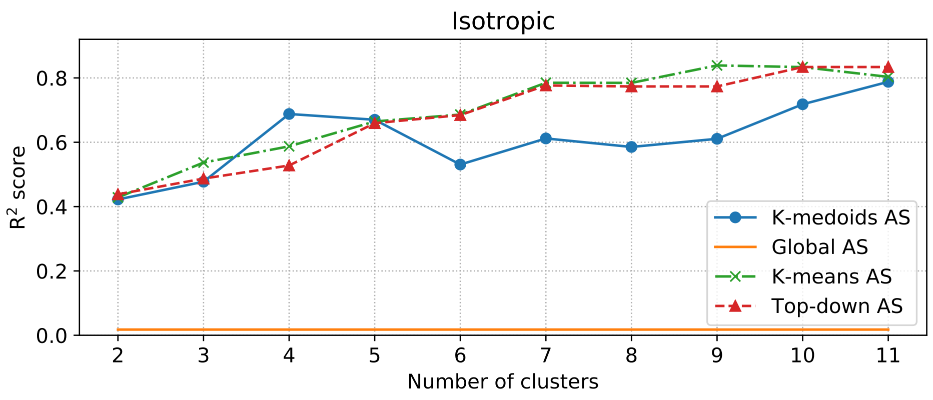


Fig. 2.6 R^2 scores comparison between global AS and local versions varying the number of clusters for the isotropic model function. Global AS corresponds to no clustering.

minimum number of children is equal to the maximum, the minimum number of elements per cluster is 10, the clustering method chosen is K-means, the normalization employed it the uniform one, and the total number of clusters is increasing from 2 to 11.

2.5.2 Higher-dimensional datasets

In this section, we consider more interesting benchmarks, for which dimension reduction in the parameter space is useful since the starting dimension of the parameter space is higher. We test the classification procedure in algorithm 4 with an objective function with 6 parameters and defined piecewise as a paraboloid with different AS dimensions. We also test the procedure of response surface design with local AS, with a classical 8-dimensional epidemic benchmark model.

Multi-dimensional hyper-paraboloid

The objective function $f : [-4,4]^6 \to \mathbb{R}$ we consider is defined piecewise as follows

$$f(x) = \begin{cases} x_1^2 & \text{if } x_1 > 0 \text{ and } x_2 > 0, \\ x_1^2 + x_2^2 & \text{if } x_1 < 0 \text{ and } x_2 > 0, \\ x_1^2 + x_2^2 + x_3^2 & \text{if } x_1 > 0 \text{ and } x_2 < 0, \\ x_1^2 + x_2^2 + x_3^2 + x_4^2 & \text{if } x_1 < 0 \text{ and } x_2 < 0. \end{cases} \tag{2.25}$$

In the 4 domains in which f is defined differently, we expect an AS dimension ranging from 1 to 4, respectively. We employed algorithm 4 using the local AS dimensions as additional features, from definition 3: the values of the hyperparameters are the following: $\varepsilon = 0.999$, $N = 6$, $p = 4$. In fig. 2.7 we plot the accuracy of the classification of the labels, associated with the connected components of the graph built as described in algorithm 4, and also the accuracy of the classification of the local active subspace dimension, that takes the values from 1 to 4. The test dataset for both the classification errors has size 1000. The score chosen to assess the quality of the classification is the mean accuracy, that is the number of correctly predicted labels over the total number of labels. For both the classification tasks 100 train samples are enough to achieve a mean accuracy above 80%.

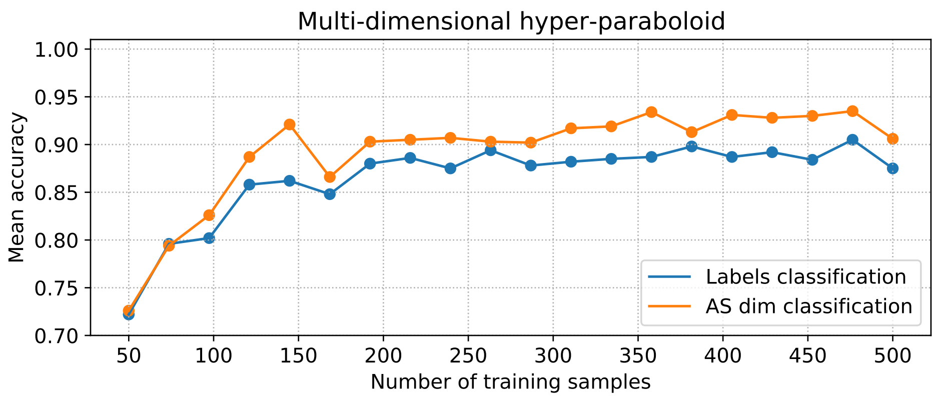


Fig. 2.7 Mean accuracy study for a training dataset increasing in size from 50 to 500 samples. The test set is made of 1000 independent samples. The classification accuracy for the procedures of connected component classification (in blue) and local AS dimension classification (in orange) are both shown.

We remark that every step is applied to a dataset of samples in a parameter space of dimension 6, even if, to get a qualitative idea of the performances of the method, in fig. 2.8 we show only the first two components of the decision boundaries of the 4 classes for both the previously described classification problems.

Ebola epidemic model

In this section, we examine the performance of the proposed methods over the dataset created with the SEIR model for the spread of Ebola[2]. The output of interest in this case is the basic reproduction number R_0 of the SEIR model, described in [78], which is computed using 8 parameters as follows

$$R_0 = \frac{\beta_1 + \frac{\beta_2 \rho_1 \gamma_1}{\omega} + \frac{\beta_3}{\gamma_2}\psi}{\gamma_1 + \psi}. \tag{2.26}$$

As shown in previous works, this function has a lower intrinsic dimensionality, and thus a meaningful active subspace, in particular of dimension 1. To evaluate the performance of the local AS we compute the R^2 score, as in eq. (2.17), varying the number of clusters from 2 to 10 for all the methods presented. The test and training datasets are composed by 500 and 300, respectively, uniformly distributed and

[2]The dataset was taken from https://github.com/paulcon/as-data-sets.

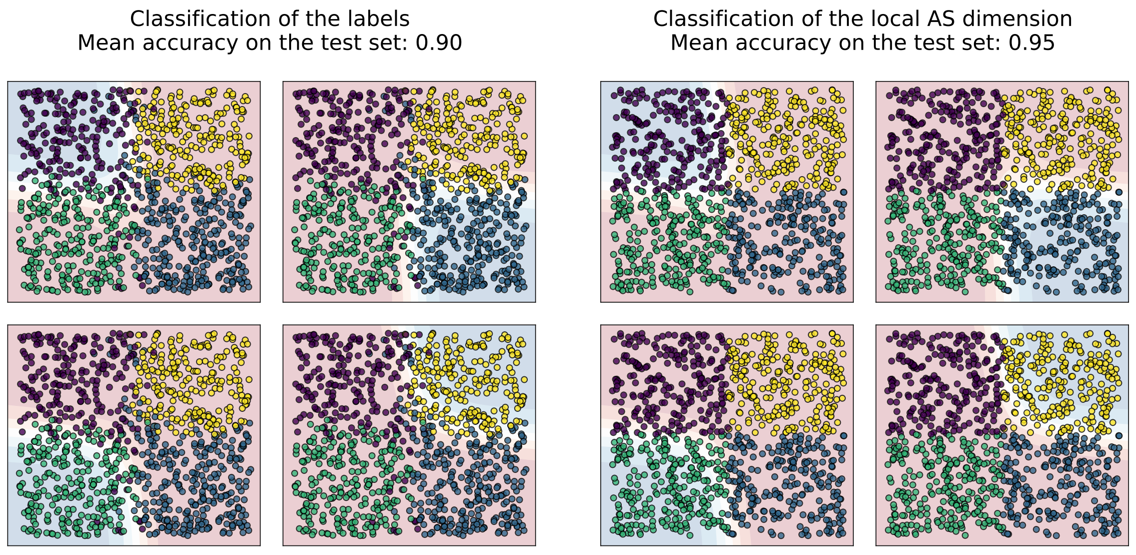


Fig. 2.8 On the left panel the decision boundaries of the 4 classes associated with the connected components of the graph built as described in algorithm 4. On the right panel the decision boundaries of the 4 classes associated to the local AS dimension from 1 to 4. The dataset has dimension 6, only the first two components of the decision boundaries and of the test samples are plotted.

independent samples. The results are reported in fig. 2.9, where as baseline we reported the R^2 for the GPR over the global AS. We can see how the use of the AS-based distance metric contributes the most with respect to the actual clustering method (compare K-medoids and hierarchical top-down in the plot). K-means, instead, does not guarantee an improved accuracy (for 4 and 9 clusters), and in general the gain is limited with respect to the other methods, especially for a small number of clusters which is the most common case in practice, since usually we work in a data scarcity regime. The results for K-medoids and top-down are remarkable even for a small number of clusters with an R^2 above 0.9 and an improvement over 10% with respect to the global AS, which means that no clustering has been used.

The hyperparameters for the hierarchical top-down algorithm are the following: the maximum local active subspace dimension is 1, the maximum number of children is equal to the number of total clusters, the minimum number of children is 2 at each refinement level, the minimum number of elements per cluster is 10, and the clustering method for each refinement is K-medoids with AS distance.

2.5.3 Datasets with vectorial outputs

In this section, we want to show how hierarchical top-down clustering and the classification procedure of algorithm 4 can be combined to improve the overall reduction in the parameter space, for a fixed

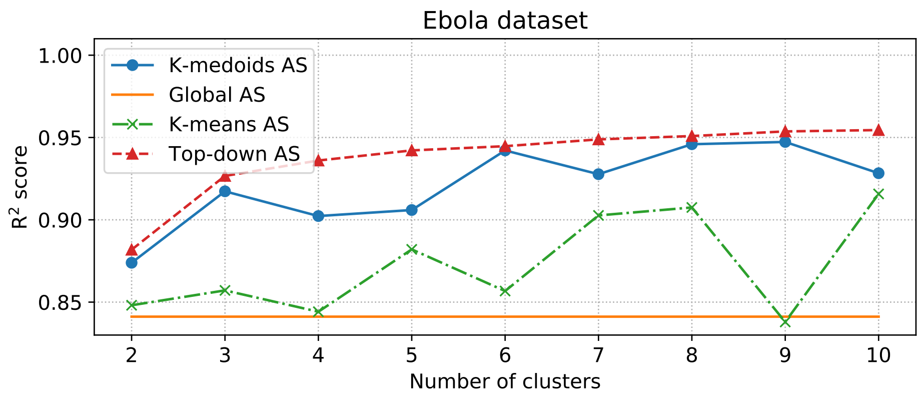


Fig. 2.9 R^2 scores comparison between global AS and local versions varying the number of clusters for the Ebola spread model. Global AS corresponds to no clustering.

lower threshold in the R^2 score. For the response surface design with active subspaces for vectorial outputs we refer to [281, 221].

Poisson equation with random diffusivity

Let us consider the stochastic Poisson problem on the square $\mathbf{x} = (x, y) \in \Omega := [0,1]^2$, defined as:

$$\begin{cases} -\nabla \cdot (\kappa \nabla u) = 1, & \mathbf{x} \in \Omega, \\ u = 0, & \mathbf{x} \in \partial\Omega_{\text{top}} \cup \partial\Omega_{\text{bottom}}, \\ u = 10y(1-y), & \mathbf{x} \in \partial\Omega_{\text{left}}, \\ \mathbf{n} \cdot \nabla u = 0, & \mathbf{x} \in \partial\Omega_{\text{right}}, \end{cases} \tag{2.27}$$

with homogeneous Neumann boundary condition on $\partial\Omega_{\text{right}}$, and Dirichlet boundary conditions on the remaining part of $\partial\Omega$. The diffusion coefficient $\kappa : (\Omega, \mathcal{A}, P) \times \Omega \to \mathbb{R}$, with $\mathcal{A}$ denoting a σ-algebra, is such that $\log(\kappa)$ is a Gaussian random field, with covariance function $G(\mathbf{x}, \mathbf{y})$ defined by

$$G(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\beta^2} \right), \quad \forall \mathbf{x}, \mathbf{y} \in \Omega, \tag{2.28}$$

where the correlation length is $\beta = 0.03$. We approximate this random field with the truncated Karhunen–Loève decomposition as

$$\kappa(s, \mathbf{x}) \approx \exp\left(\sum_{i=0}^{m} X_i(s) \gamma_i \boldsymbol{\psi}_i(\mathbf{x}) \right), \qquad \forall (s, \mathbf{x}) \in \Omega \times \Omega, \tag{2.29}$$

where $(X_i)_{i \in 1,\dots,m}$ are independent standard normal distributed random variables, and the eigenpairs of the Karhunen–Loève decomposition of the zero-mean random field κ are denoted with $(\gamma_i, \boldsymbol{\psi}_i)_{i \in 1,\dots,d}$.

The parameters $(X_i)_{i\in 1,\dots,m=10}$ sampled from a standard normal distribution are the coefficients of the Karhunen-Loève expansion truncated at the first 10 modes, so the parameter space has dimension $m = 10$. This test case is implemented in tutorial 5 of the `ATHENA` [218] package.

The domain Ω is discretized with a triangular unstructured mesh $\mathcal{T}$ with 3194 triangles. The simulations are carried out with the finite element method with polynomial order 1. The solution u is evaluated at 1668 degrees of freedom, thus the output is vectorial with dimension $d = 1688$. As done in [281, 221], the output is enriched with the metric induced by the Sobolev space $H^1(\Omega)$ on to the finite element space of polynomial order 1: the metric is thus represented by a $d \times d$ matrix M obtained as the sum of the mass and stiffness matrices of the numerical scheme, and it is involved in the AS procedure when computing the correlation matrix $E\left[Df\, M\, Df^T\right]$, where Df is the $m \times d$ Jacobian matrix of the objective function $f : \mathbb{R}^{10} \to \mathbb{R}^d$.

Since the output is high-dimensional we classified algorithm 4 the output space in 6 clusters, using the Grassmann distance from eq. (2.23), as shown in fig. 2.10.

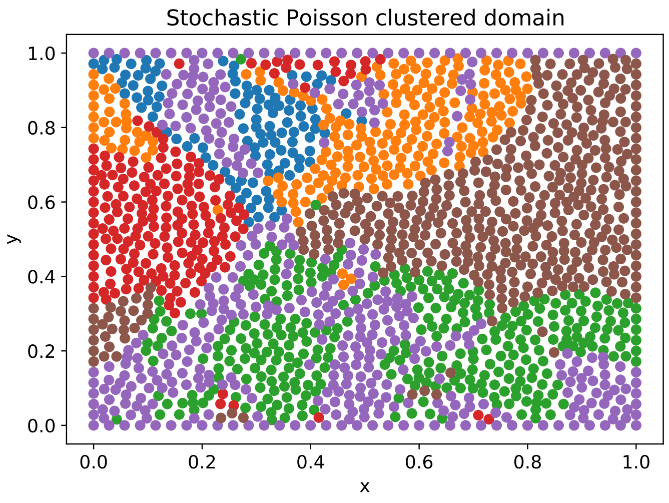


Fig. 2.10 Subdivision of the spatial domain Ω in 6 clusters based on the Grassmann distance from definition 4, i.e. the clusters correspond to the connected components of the graph built on top of the degrees of freedom with adjacency list determined using as distance definition 4.

Afterwards, we applied hierarchical top-down clustering to every one of the 6 triplets of inputs-outputs-gradients, obtained restricting the outputs and the gradients to each one of the 6 clusters. The specifics of hierarchical top-down clustering we employed are the following: the minimum and maximum number of children for each refinement is equal to the total number of clusters, which is 4, the minimum number of elements in each cluster is 10, and the clustering algorithm chosen is K-medoids with the AS distance. The size of the training and test datasets is respectively 500 and 150. The gradients are evaluated with the adjoint method. Since the output is vectorial we employed the mean R^2 score, where the average is made among the components of the vectorial output considered.

Then for every lower threshold on the R^2 score, we increase one by one the dimension of the 6×4 local clusters, until all the R^2 scores of each of the 6 triplets are above the fixed threshold. The same procedure is applied to the whole dataset of inputs-outputs-gradients but executing hierarchical top-down clustering just once, for all the output's components altogether.

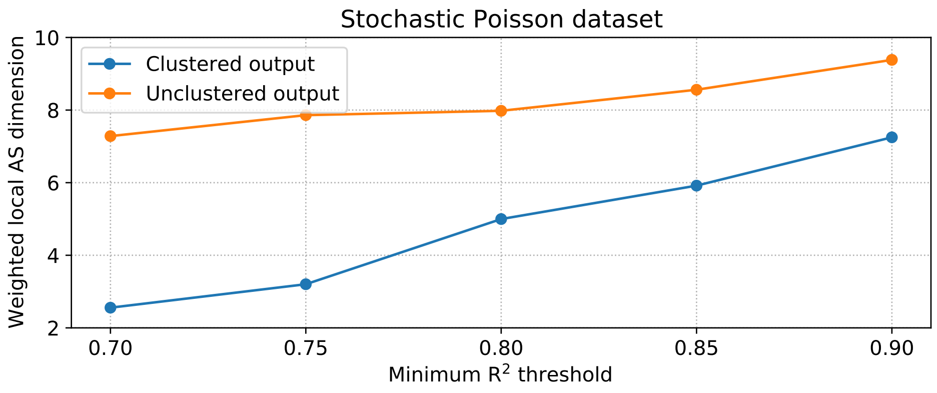


Fig. 2.11 In orange the local AS dimensions weighted on the number of elements of each of the 4 clusters in the parameter space, obtained with hierarchical top-down clustering. In blue the local AS dimensions weighted on the number of elements of each of the 4 clusters in the parameter space, obtained with hierarchical top-down clustering, times 6 clustered outputs (see fig. 2.10) for a total of 24 terms in the weighted average.

The results are reported in fig. 2.11. In the case of the clustered outputs, the local dimension of each one of the 6 clustered outputs times 4 local clusters in the parameter space, for a total of 24 local clusters, are weighted with the number of elements of each cluster. In the same way the 4 clusters of the case with unclustered outputs are weighted with the number of the elements of each one of the 4 clusters. It can be seen that for every fixed threshold, there is an evident gain, with respect to the dimension reduction in the parameter space, in clustering the outputs and then performing hierarchical top-down clustering in the parameter space.

Shape design of an airfoil

For this vectorial test case, we consider the temporal evolution of the lift coefficient of a parametrized NACA airfoil. Here we briefly present the problem we solve to create the dataset, we refer to [250] for a deeper description.

Let us consider the unsteady incompressible Navier-Stokes equations described in an Eulerian framework on a parametrized space-time domain $S(\boldsymbol{\mu}) = \Omega(\boldsymbol{\mu}) \times [0,T] \subset \mathbb{R}^2 \times \mathbb{R}^+$. The vectorial velocity field $\mathbf{u} : S(\boldsymbol{\mu}) \to \mathbb{R}^2$, and the scalar pressure field $p : S(\boldsymbol{\mu}) \to \mathbb{R}$ solve the following parametric

PDE:

$$\begin{cases} \mathbf{u_t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot 2\nu \nabla^s \mathbf{u} = -\nabla p & \text{in } S(\boldsymbol{\mu}), \\ \nabla \cdot \mathbf{u} = \mathbf{0} & \text{in } S(\boldsymbol{\mu}), \\ \mathbf{u}(t, \mathbf{x}) = \mathbf{f}(\mathbf{x}) & \text{on } \Gamma_{\text{in}} \times [0, T], \\ \mathbf{u}(t, \mathbf{x}) = \mathbf{0} & \text{on } \Gamma_0(\boldsymbol{\mu}) \times [0, T], \\ (\nu \nabla \mathbf{u} - p\mathbf{I})\mathbf{n} = \mathbf{0} & \text{on } \Gamma_{\text{out}} \times [0, T], \\ \mathbf{u}(0, \mathbf{x}) = \mathbf{k}(\mathbf{x}) & \text{in } S(\boldsymbol{\mu})_0 \end{cases} . \tag{2.30}$$

Here, $\Gamma = \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \cup \Gamma_0$ denotes the boundary of $\Omega(\boldsymbol{\mu})$ composed of inlet boundary, outlet boundary, and physical walls, respectively. With $\mathbf{f}(\mathbf{x})$ we indicate the stationary non-homogeneous boundary condition, and with $\mathbf{k}(\mathbf{x})$ the initial condition for the velocity at $t = 0$. The geometrical deformation is applied to the boundary $\Gamma_0(\boldsymbol{\mu})$. The undeformed configuration corresponds to the NACA 4412 wing profile [3, 139]. To alter such geometry, we adopt the shape parametrization and morphing technique proposed in [128], where 5 shape functions are added to the airfoil profiles. They are commonly called Hicks-Henne bump functions. Let y_u and y_l be the upper and lower ordinates of the profile, respectively. The deformation of such coordinates is described as follows

$$y_u = \overline{y_u} + \sum_{i=1}^{5} c_i r_i, \qquad y_l = \overline{y_l} - \sum_{i=1}^{5} d_i r_i, \tag{2.31}$$

where the bar denotes the reference undeformed state. The parameters $\boldsymbol{\mu} \in \mathbb{D} \subset \mathbb{R}^{10}$ are the weights coefficients, c_i and d_i, associated with the shape functions r_i. In particular, we set $\mathbb{D} := [0, 0.03]^{10}$. The explicit formulation of the shape functions can be found in [128]. For these datasets, the Reynolds number is $Re = 50000$. The time step is $dt = 10^{-3}$ s. For other specifics regarding the solver employed and the numerical method adopted we refer to [250].

As outputs, we considered the values of the lift coefficient, every 15 time steps, from 100 ms to 30000 ms, for a total of 1994 components. Even in this case, the output is classified with algorithm 4 with distance defined in definition 3. The values of the lift coefficient physically interesting are collected at last, after an initialization phase. Nonetheless, for the purpose of having a vectorial output we considered its value from the time instant 100 ms. The procedure finds two classes and splits the ordered output components in two parts: from the component 0 to 996, the local AS dimension is 1, for the remaining time steps it is higher. So we can expect an improvement in the efficiency of the reduction in the parameter space when considering separately these two sets of outputs components as fig. 2.12 shows. The weighted local AS dimension is in fact lower when using clustering, for every minimum R^2 threshold.

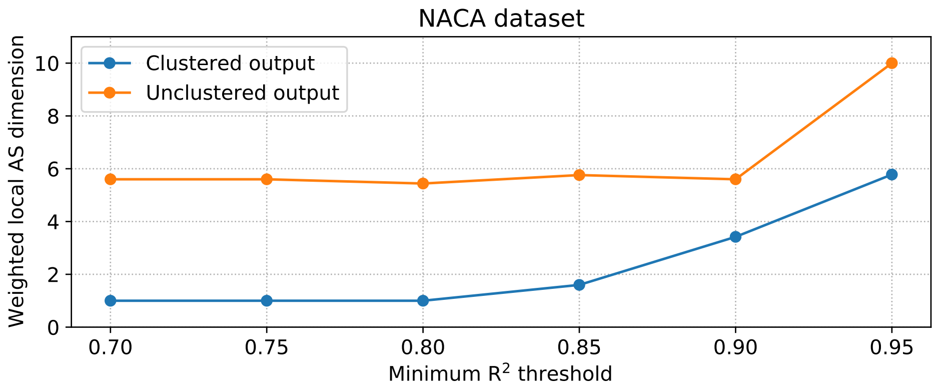


Fig. 2.12 In orange the local AS dimensions weighted on the number of elements of each of the 2 clusters in the parameter space, obtained with hierarchical top-down clustering. In blue the local AS dimensions weighted on the number of elements of each of the 2 clusters in the parameter space, obtained with hierarchical top-down clustering, times 2 clustered outputs for a total of 4 terms in the weighted average.

2.6 Conclusions and perspectives

In this work, we present a new local approach for parameter space reduction which exploits supervised clustering techniques, such as K-means, K-medoids, and hierarchical top-down, with a distance metric based on active subspaces. We called this method local active subspaces (LAS). The proposed metric tend to form the clusters transversally with respect to the active subspace directions thus reducing the approximation error induced by the choice of the non-optimal profile.

The theoretical formulation provides error estimates for the construction of response surfaces over the local active subspaces. We also present a classification approach to capture the optimal AS dimension for each cluster and can be used as a preprocessing step, both for the inputs and the vectorial outputs, for the construction of more accurate regressions and surrogate modelling. The proposed approach is very versatile, especially the hierarchical top-down clustering which can incorporate quite different criteria. The methodology has been validated over a vast range of datasets, both scalar and vector-valued, showing all the strengths and possible weaknesses, in the case of radial symmetric functions. In all the test cases LAS achieved superior performance with respect to the classical global approach.

Possible future lines of research can focus on the study of the extension of these methods to nonlinear parameter space reduction techniques, or on the use of more advanced clustering criteria.

Chapter 3

Multi-fidelity nonlinear regression with Active Subspaces

Multi-fidelity models are of great importance due to their capability of fusing information coming from different numerical simulations, surrogates, and sensors. The focus, in this case, is on the approximation of high-dimensional scalar functions with low intrinsic dimensionality. By introducing a low dimensional bias we can fight the curse of dimensionality affecting these quantities of interest, especially for many-query applications. A gradient-based reduction of the parameter space through linear active subspaces or nonlinear transformations of the input space is sought. Low-fidelity response surfaces based on such reduction are built, thus enabling nonlinear autoregressive multi-fidelity Gaussian process regression without the need of running new simulations with simplified physical models. This has great potential in the data scarcity regime affecting many engineering applications. In this work, it is presented a new multi-fidelity approach that involves active subspaces and the nonlinear level-set learning method. The proposed framework is tested on two high-dimensional benchmark functions and on a more complex car aerodynamics problem involving the Reynolds Averaged Navier-Stokes equations. It is shown how a low intrinsic dimensionality bias can increase the accuracy of Gaussian process response surfaces.

Contents

3.1 Literature review

The curse of dimensionality affects the realization of reliable models for high-dimensional function approximation. This problem is particularly evident in the data scarcity regime which characterizes many industrial and engineering applications. We address this issue by exploiting parameter space reduction techniques in a multi-fidelity setting.

Gaussian processes (GP) [271] have spread in many fields as a reliable regression (GPR) method, especially for optimization and inverse problems. Many extensions stemmed from the original formulation, such as for kernel methods [142], and for big data and memory limitations [159, 164]. On the other hand the exploitation of multi-fidelity models had a huge impact in the scientific computing community thanks to the possibility to integrate simulations and data coming from different models and sources. For an overview of different applications we suggest [145, 91, 211, 31, 32, 151]. A particularly promising nonlinear autoregressive multi-fidelity Gaussian process regression (NARGP), was proposed in [200]. Recent advancements in the context of physics-informed neural networks [212] in a multi-fidelity setting for function approximation and inverse PDE problems, can be found in [178].

These models achieve increased expressiveness with some kind of nonlinear approach extending GP models to non-GP processes at the cost of an additional computational load. In this direction, some works aim to obtain computationally efficient heteroscedastic GP models using a variational inference approach [158], or a nonlinear transformation [234]. This approach is extended to multi-fidelity models starting from the linear formulation presented by Kennedy and O'Hagan [145] towards deep GP [68] and NARGP.

Classical low-fidelity models obtained by coarse grids or simplified physical models still suffer the curse of dimensionality when used for high-dimensional GP construction. Linear parameter space reduction with Active Subspaces (AS) [60] can fight such curse using input-output couples obtained by high-fidelity simulations. Successful applications of parameter space reduction with active subspaces can be found in many engineering fields: naval and nautical problems [249], shape optimization [170, 101, 73, 72], car aerodynamics studies [188], inverse problems [62, 183], cardiovascular studies coupled with intrusive model order reduction [248], for the study of high-dimensional parametric PDEs [186], and in CFD problems in a data-driven setting [71, 250], among others. New extensions of AS have also been developed in recent years such as AS for multivariate vector-valued functions [281], a kernel approach for AS for scalar and vectorial functions [221], a localization extension for both regression and classification tasks [220], and sequential learning of active subspaces [273]. The multi-fidelity setting has been used to find an active subspace given different fidelity models [154].

Other nonlinear techniques for parameter space reduction include manifold learning [131, 205], active manifolds [33] and nonlinear level-set learning (NLL) [283]. NLL adopts a Reversible Neural Networks (RevNet) architecture to learn an effective parameter space deformation to capture the geometry of the objective function level-sets and parametrize them.

With this contribution, we show how to integrate linear and nonlinear parameter space dimensionality reduction within a multi-fidelity regression scheme based on Gaussian processes to increase the accuracy of high-dimensional response surfaces. The low-fidelity models are built with AS or NLL and incorporated in the NARGP framework, following the preliminary results obtained in [219]. An extensive automotive test case is presented with different configurations.

This work is organized as follows: in Section 3.2 we introduce multi-fidelity Gaussian process regression starting from the building block of a single fidelity up to the NARGP method; in Section 3.3 we focus on the parameter space reduction with active subspaces and nonlinear level-set learning which are going to be used to construct the low-fidelity models; Section 3.4 shows how to add the low-intrinsic dimensionality bias into the NARGP framework, accompanied by pseudocode; in Section 3.5 we present the numerical results of the proposed approach applied to two benchmark models, and to an automotive application; finally Section 3.6 draw the conclusions and some future research lines.

3.2 Multi-fidelity Gaussian process regression

In this section, we are going to briefly recall the Gaussian process regression (GPR) technique in order to better characterize the nonlinear autoregressive multi-fidelity Gaussian process regression (NARGP) introduced in [200]. NARGP represents the main framework for our proposed multi-fidelity method. We are going to consider the general setting with multiple levels of fidelity.

3.2.1 Gaussian process regression

Gaussian process regression is a supervised technique to approximate unknown functions given a finite set of input/output pairs $\mathcal{S} = \{x_i, y_i\}_{i=1}^N$. Let $f : \mathcal{X} \subset \mathbb{R}^m \to \mathbb{R}$ be the scalar function of interest. The set $\mathcal{S}$ is generated through f with the following relation: $y_i = f(x_i)$, which are the noise-free observations. We assigned a prior to f with mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}'; \theta)$, that is $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'; \theta))$. The prior expresses our beliefs about the function before looking at the observed values. From now on we consider zero mean $\mathcal{GP}$, that is $m(\mathbf{x}) = \mathbf{0}$, and we define the covariance matrix as $\mathbf{K}_{i,j} = k(x_i, x_j; \theta)$, with $\mathbf{K} \in \mathbb{R}^{N \times N}$. In order to make predictions using the Gaussian process we still need to find the optimal values for the hyperparameters vector θ by maximizing the log likelihood:

$$\log p(\mathbf{y}|\mathbf{x},\theta) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| - \frac{N}{2}\log 2\pi. \tag{3.1}$$

Let $\mathbf{x}_*$ be the test samples, and $\mathbf{K}_{N*} = k(\mathbf{x},\mathbf{x}_*;\theta)$ be the matrix of the covariances evaluated at all pairs of training and test samples, and in a similar fashion $\mathbf{K}_{*N} = k(\mathbf{x}_*,\mathbf{x};\theta)$, and $\mathbf{K}_{**} = k(\mathbf{x}_*,\mathbf{x}_*;\theta)$. By conditioning the joint Gaussian distribution on the observed values we obtain the predictions f_* by sampling the posterior as

$$f_*|\mathbf{x}_*,\mathbf{x},\mathbf{y} \sim \mathcal{N}(\mathbf{K}_{*N}\mathbf{K}^{-1}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_{*N}\mathbf{K}^{-1}\mathbf{K}_{N*}). \tag{3.2}$$

3.2.2 Nonlinear multi-fidelity Gaussian process regression

In this section, we briefly present the nonlinear autoregressive multi-fidelity Gaussian process regression (NARGP) scheme [200]. It extends the concepts present in [145, 160] to nonlinear correlations between the different available fidelities.

The procedure is purely data-driven. We start from the input/output pairs corresponding to p levels of increasing fidelity, that is

$$\mathcal{S}_q = \{x_i^q, y_i^q\}_{i=1}^{N_q} \subset \mathcal{X} \times \mathbb{R} \subset \mathbb{R}^m \times \mathbb{R}, \qquad \text{for } q \in \{1,\dots,p\}, \tag{3.3}$$

where $y_i^q = f_q(x_i^q)$. With p we indicate the highest fidelity. We also assume that the design sets have a hierarchical structure:

$$\pi(S_p) \subset \pi(S_{p-1}) \subset \cdots \subset \pi(S_1), \tag{3.4}$$

where $\pi : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ is the projection onto the first m coordinates. Due to this hierarchy, when the fidelities of the available datasets cannot be neatly assessed, it is reasonable to consider the cost needed to produce them as ordering criterion, see Remark 10.

The NARGP formulation assigns a Gaussian process to each fidelity model f_q, so they are completely defined by the mean field m_q, with the constant zero field as prior, and by their kernel k_q, as follows:

$$y_q(\bar{x}) - \varepsilon \sim \mathcal{GP}(f_q(\bar{x}))|m_q(\bar{x}), k_q(\theta_q)) \quad \forall q \in \{1,\dots,p\}\,, \tag{3.5}$$

where $\varepsilon \sim \mathcal{N}(0,\sigma^2)$ is a noise term and

$$\bar{x} := \begin{cases} (\mathbf{x}, f_{q-1}(\mathbf{x})) \in \mathbb{R}^m \times \mathbb{R}, & q > 1 \\ \mathbf{x} \in \mathbb{R}^m, & q = 1 \end{cases}. \tag{3.6}$$

The definition of the kernel $k_q(\theta_q)$ implements the auto-regressive characteristic of the method since it depends on the previous fidelity model f_{q-1}:

$$k_q((x, f_{q-1}(x)), (x', f_{q-1}(x')); \theta_q) = k_q^\rho(x,x';\theta_q^\rho) \cdot k_q^f(f_{q-1}(x), f_{q-1}(x'); \theta_q^f) + k_q^\delta(x,x';\theta_q^\delta)\,. \tag{3.7}$$

The hyperparameters to be tuned are represented by $\theta_q \equiv (\theta_q^\rho, \theta_q^f, \theta_q^\delta)$ and are associated respectively to the multiplicative kernel k_q^ρ, the auto-regressive kernel k^f, and the kernel k_q^δ, which corresponds to the non auto-regressive part in the sum of Equation 3.7. For our applications, we employ the radial basis function kernel with automatic relevance determination (RBF-ARD) [271], but there are other possible choices.

The presence of the multiplicative kernel k_q^ρ allows nonlinear interdependencies between subsequent fidelities to be modeled, surpassing a linear auto-regressive multi-fidelity scheme. The latent manifold that relates the inputs, the lower fidelity posterior and the high-fidelity posterior is in this case nonlinear [200].

We use the notation$(\mathbf{x}, y_{q-1}(\mathbf{x}))$ for the training set and $\mathbf{x}_*$ for the new input. So in order to evaluate the predictive mean and variance for a new input $\mathbf{x}_*$ we have to integrate the posterior $p(f_q(\mathbf{x}_*)|f_{q-1}, \mathbf{x}_*, \mathbf{x}_q, y_q)$ defined as

$$
\begin{aligned}
& f_q(\mathbf{x}_*|f_{q-1}, \mathbf{x}_*, \mathbf{x}_q, y_q) \sim \mathcal{N}(\mathbf{K}_{*N}^q(\mathbf{K}^q)^{-1}y_q, \mathbf{K}_{**}^q - \mathbf{K}_{*N}^q(\mathbf{K}^q)^{-1}\mathbf{K}_{N*}^q), && (3.8)\\
& \mathbf{K}_{*N}^q = k_q((\mathbf{x}_*, f_{q-1}(\mathbf{x}_*), (\mathbf{x}_{q-1}, y_{q-1}); \theta_q), && (3.9)\\
& \mathbf{K}_{N*}^q = k_q((\mathbf{x}_{q-1}, y_{q-1}), (\mathbf{x}_*, f_{q-1}(\mathbf{x}_*); \theta_q), && (3.10)\\
& \mathbf{K}^q = k_q((\mathbf{x}_{q-1}, y_{q-1}), (\mathbf{x}_{q-1}, y_{q-1}); \theta_q), && (3.11)
\end{aligned}
$$

over the Gaussian distribution of the prediction at the previous level $f_{q-1}(\mathbf{x}_*) \sim \mathcal{N}(m_{q-1}(\mathbf{x}_*), k_{q-1}(\mathbf{x}_*))$. Apart from the first level of fidelity $q = 1$ the posterior probability distribution given the previous fidelity models is no longer Gaussian. So, in practice, the following integral is approximated with recursive Monte Carlo at each fidelity level, for all $q \in \{2, \ldots, p\}$,

$$
\begin{aligned}
p(f_q^{\text{post}}(\mathbf{x}_*)) &:= p(f_q(\mathbf{x}_*)|f_{q-1}, \mathbf{x}_*, \mathbf{x}_q, y_q) = \\
&= \int_{\mathcal{X}} p(f_q(\mathbf{x}_*)|s, \mathbf{x}_*, \mathbf{x}_q, y_q) d\mathcal{L}_{f_{q-1}^{\text{post}}(\mathbf{x}_*)}(s) && (3.12)\\
p(f_1^{\text{post}}(\mathbf{x}_*)) &:= p(f_1(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{x}_1, y_1) \sim \mathcal{N}(m_1(\mathbf{x}_*), k_1(\mathbf{x}_*)), && (3.13)
\end{aligned}
$$

where $\mathcal{L}_{f_{q-1}^{\text{post}}(\mathbf{x}_*)}$ is the probability law of $f_{q-1}^{\text{post}}(\mathbf{x}_*) \sim \mathcal{N}(m_{q-1}(\mathbf{x}_*), k_{q-1}(\mathbf{x}_*))$. In the applications, we always use 200 to 10000 Monte Carlo samples, since the results do not vary much increasing them for our test cases.

The hyperparameters θ_q are optimized (non-recursively) with maximum log-likelihood estimation for each GP model $\mathcal{GP}(f_q|\mathbf{0}, k^q(\theta_q))$, for all $q \in \{1, \ldots, p\}$,

$$
\operatorname*{argmin}_{\theta_q} - \log p(f_q(\mathbf{x}_q)|\mathbf{x}_q, y_q, y_{q-1}, \theta_q) \propto \frac{1}{2}\log|K^q(\theta_q)| + \frac{1}{2}y_q^T(K^q(\theta_q))^{-1}y_q \,, \tag{3.14}
$$

this is why a hierarchical dataset is needed. The hyperparameters tuning is achieved maximizing the log-likelihood with the gradient descent optimizer L-BFGD in GPy [98]. For some test cases, the training procedure is subject to relevant perturbations relative to the number of restarts, this is especially true in higher dimensions of the parameter space.

3.3 Parameter space reduction

Our aim is to test multi-fidelity Gaussian process regression models to approximate objective functions which depend on inputs/parameters sampled from a high-dimensional space. Low-fidelity models relying on a physics-based or numerical model reduction — for example a coarse discretization or a more specific numerical model order reduction — still suffer from the high dimensionality of the input space. In our approach, we try to tackle these problems by searching for a surrogate (low-fidelity) model accounting for the complex correlations among the input parameters that concur to the output of interest. With this purpose in mind, in this section we are going to briefly present the active subspaces (AS) [60], and the nonlinear level-set learning (NLL) method [283] for parameter space reduction in order to design response surfaces with Gaussian process regression.

3.3.1 Active subspaces

Let $\mathbf{X}$ be an absolutely continuous random variable with probability density ρ, such that $\operatorname{supp}(\rho) = \mathcal{X} \subset \mathbb{R}^m$. The variable $\mathbf{X}$ represents the inputs, and m denotes the dimension of the input parameter space. With simple Monte Carlo we can approximate the uncentered covariance matrix of the gradients of the function of interest as

$$E_\rho[\nabla_{\mathbf{x}} f(\nabla_{\mathbf{x}} f)^T] \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathbf{x}} f(\mathbf{X}_i)(\nabla_{\mathbf{x}} f(\mathbf{X}_i))^T, \tag{3.15}$$

where N denotes the number of samples. We are looking for the highest spectral gap $\lambda_r - \lambda_{r+1}$ in the sequence of ordered eigenvalues of the approximated correlation matrix. The active subspace is the eigenspace corresponding to the first r eigenvalues $\lambda_1, \dots, \lambda_r$ and it is denoted with the matrix $\hat{W}_r \in \mathcal{M}(m \times r)$ whose columns are the corresponding r active eigenvectors. The inactive subspace is defined as the span of the remaining eigenvectors. On it f is almost flat on average, so we can safely discard such component without compromising too much the accuracy. We can thus build a response surface $\mathcal{R}$ using a Gaussian process regression trained with N_{train} pairs $\{\hat{W}_r^T \mathbf{x}_i, y_i\}_{i=1}^{N_{\text{train}}}$ of active inputs and outputs.

The mean square regression error is bounded a priori [60] by

$$E_\rho\left[(f(\mathbf{X}) - \mathcal{R}(\hat{W}_r^T \mathbf{X}))^2\right] \leq C_1(1+N^{-1/2})^2 \left(\varepsilon(\lambda_1 + \cdots + \lambda_r)^{1/2} + (\lambda_{r+1} + \cdots + \lambda_m)^{1/2}\right)^2 + C_2\delta, \tag{3.16}$$

where C_1 and C_2 are constants, ε quantifies the error in the approximation of the true active subspace W_r with $\hat{W}_r$ obtained from the Monte Carlo approximation, and $C_2\delta$ is a bound on the mean squared error of the Gaussian process regression over the active subspace:

$$E_{\rho_{\hat{W}_r \mathbf{X} | \hat{W}_{m-r} \mathbf{X}}} \left[\left(\overline{E_\rho \left[f(\mathbf{X}) | \sigma(\hat{W}_r^T \mathbf{X}) \right]} - \mathcal{R}(\hat{W}_r^T \mathbf{X}) \right)^2 \right] \leq C_2 \delta, \tag{3.17}$$

where $\rho_{\hat{W}_r \mathbf{X} | \hat{W}_{m-r} \mathbf{X}}$ is the probability of the active variables conditioned on the inactive ones, and $\overline{E_\rho \left[f(\mathbf{X}) | \sigma(\hat{W}_r^T \mathbf{X}) \right]}$ is the random variable $f(\mathbf{X})$ conditioned on the σ-algebra generated by $\hat{W}_r^T \mathbf{X}$ and approximated with the Monte Carlo method.

3.3.2 Nonlinear level-set learning method

This method seeks a bijective nonlinear transformation $g_{\text{NLL}} : \mathcal{X} \to \tilde{\mathcal{X}} \subset \mathbb{R}^m$ to capture the geometry of level sets and parametrize them in a low-dimensional space. To this end in [283] they employ reversible networks (RevNets) [47] to learn the transformation g_{NLL}. The designed loss function uses samples of the gradients of the target function to encourage the transformed function to be sensitive to only a few active coordinates.

To construct the RevNet, the following architecture [113], which is reversible by definition, is employed:

$$\begin{cases} u_{n+1} = u_n + h K_{n,1}^T \sigma(K_{n,1} v_n + b_{n,1}), \\ v_{n+1} = v_n - h K_{n,2}^T \sigma(K_{n,2} u_n + b_{n,2}), \end{cases} \qquad \text{for } n = 0, 1, \dots, N-1, \tag{3.18}$$

where u and v are partitions of the states, h is a scalar time step, the matrices K contain the weights, b represent the biases, and σ is the activation function. We remark that the original coordinates and the transformed ones are split in two in u and v.

3.4 Multi-fidelity data fusion with active subspaces

Our study regards the design of a nonlinear autoregressive multi-fidelity Gaussian process regression (NARGP) [200] with two fidelities: the high-fidelity corresponds to a relatively accurate and costly model, for example, a numerical model which requires computationally intensive simulations to obtain a scalar output for each parameter sample; and the low-fidelity level which comes from a response surface built through a parameter space reduction technique — here we focus on active subspaces but little modifications are required in order to use NLL as we are going to show. We consider models with high-dimensional input space but with a low intrinsic dimensionality. This setting characterizes many industrial applications [188, 249, 170].

In fact, the inductive biases we impose come mainly from two sources: the kernel of the Gaussian process (lengthscale, noise, regularity of the stochastic process) and the low-fidelity intrinsic dimensionality assumption (presence of a dominant linear or nonlinear active subspace). The key feature of the method is the imposition of the latter on the multi-fidelity model design: we expect that a hint towards the presence of an active subspace will be transferred from the low-fidelity to the high-fidelity level through the discovery of nonlinear correlations between the low-fidelity predictions, and the high-fidelity inputs/outputs dataset. In this way, the accuracy should increase in the data-scarcity regime, i.e. when the number of high-fidelity samples is not enough to obtain an accurate single-fidelity regression. The overhead with respect to the original procedure [200] is the evaluation of the active subspace from the high-fidelity inputs and the training of the whole multi-fidelity model; this costs are usually negligible as shown in section 3.5.2.

In Figure 3.1 we present an illustrative scheme of the proposed NARGP-AS method; the underlying objective function is an hyperbolic paraboloid $f:[0,1]^2\subset\mathbb{R}^2\to\mathbb{R},\ f(x_1,x_2)=x_1^2-x_2^2$ and is shown only for the purpose of representing the procedure more clearly. The high-fidelity flow field belongs to the automotive application of section 3.5.2.

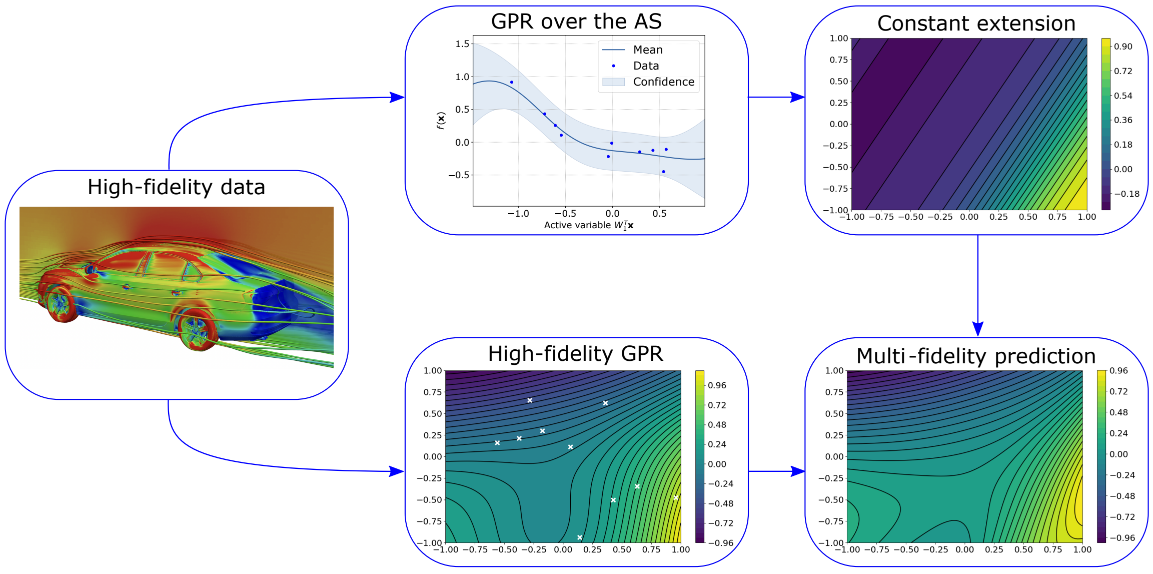


Fig. 3.1 Illustrative scheme of the NARGP-AS method. Starting from 10 high-fidelity data (depicted with blue dots and white crosses) we construct as low-fidelity model a response surface that is constant along the inactive subspace.

For clarity, we will use the letters H and L as labels for the high-fidelity and low-fidelity models respectively, instead of the fidelity levels $q=1$ and $q=2$. Changing, as just described, the notations of subsection 3.2.2, we consider

$$S^L=\{x_i^L,y_i^L\}_{i=1}^{N_L}\subset\mathbb{R}^m\times\mathbb{R},$$
$$S^H=\{x_i^H,y_i^H\}_{i=1}^{N_H}\subset\mathbb{R}^m\times\mathbb{R},$$

and additionally $\{dy_i^H\}_{i=1}^{N_2}\subset\mathbb{R}^m$ the gradients corresponding to the high-fidelity dataset $S^H=\{x_i^H,y_i^H\}_{i=1}^{N_H}$: in principle, the gradients can be directly obtained from the model of interest (with adjoint methods in case of PDE models for example) or approximated from the input-output pairs S^H. For the influence of the gradients' approximation on the regression error, see [60]. For our test cases, we employ the exact gradients when available (for the benchmarks in sections 3.5.1 and 3.5.1) or we approximate them from the high-fidelity GPR (in the test cases in sections 3.5.2, 3.5.2 and 3.5.2). This does not results in additional costs, since the HF GPR is needed in the NARGP-AS procedure.

The high-fidelity dataset S^H (and the corresponding gradients) represents by itself all the necessary ingredients: S^L is built from S^H through a response surface on the linear or nonlinear active subspace. For this purpose the dataset S^H is employed, with the corresponding gradients, to find an active subspace $\hat{W}_r$ or train a RevNet, as described in subsections 3.3.1 and 3.3.2.

Then, since $S^H \subset S^L$, we write

$$S^L \setminus S^H = \{x_i^L, y_i^L\}_{i=1}^{N_L} \setminus \{x_i^H, y_i^H\}_{i=1}^{N_H} = \{\tilde{x}_i^L, \tilde{y}_i^L\}_{i=1}^{N_L-N_H}. \tag{3.19}$$

The additional low-fidelity inputs $\{\tilde{x}_i^1\}_{i=1}^{N_L-N_H}$ are sampled independently from the inputs' probability distribution, while the additional low-fidelity outputs $\{\tilde{y}_i^L\}_{i=1}^{N_L-N_H}$ are the predictions associated to the active components of the additional low-fidelity inputs $\{\hat{W}_r^T \tilde{x}_i^L\}_{i=1}^{N_L-N_H}$, obtained from the response surface trained on

$$\{\hat{W}_r^T x_i^H, y_i^H\}_{i=1}^{N_H} \subset \mathbb{R}^r \times \mathbb{R}.$$

The response surface is trained as a Gaussian process regression as described in subsection 3.3.1. The procedure is synthetically reviewed through Algorithm 5. The number of low-fidelity samples is chosen until a good approximation of the low-fidelity response surface is obtained. As it is experimentally shown in Figure 3.8, additional low-fidelity samples do not improve the accuracy of the multi-fidelity model afterward.

Remark 8 (Nonlinear level-set learning as LF model). If NLL is employed to build the low-fidelity level, only the first step of Algorithm 5 is changed. For our applications, the GPR designed with NLL has dimension one.

Algorithm 5: NARGP-AS response surface design algorithm.

```
input ::
        training high-fidelity inputs, outputs, gradients triplets:
          $\{(\mathbf{x}_i^H, y_i^H, dy_i^H)\}_{i=1}^{N_H} \subset \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^m$
        low-fidelity inputs $\{\mathbf{x}_i^L\}_{i=1}^{N_L} \subset \mathbb{R}^m$
output ::
        multi-fidelity model:
          $\big((f_H|x_i^H, y_i^H),\ (f_L|x_i^L)\big) \sim (\mathcal{GP}(f_H|m_H, k_H), \mathcal{GP}(f_L|m_L, k_L))$
1 Compute the active subspace $\hat{W}_r$ with the high-fidelity gradients $\{dy_i^H\}_{i=1}^{N_H}$
2 Build the response surface $\mathcal{R}(\hat{W}_r^T \mathbf{X})$ with a GP regression from $\{(\hat{W}_r^T \mathbf{x}_i^H, y_i^H)\}_{i=1}^{N_H}$
3 Predict the low-fidelity outputs $\{y_i^L\}_{i=1}^{N_L}$ at $\{\mathbf{x}_i^L\}_{i=1}^{N_L}$ and the training high-fidelity inputs
    $\{y_i^H\}_{i=1}^{N_H}$ at $\{\mathbf{x}_i^H\}_{i=1}^{N_H}$ with the response surface
4 Train the multi-fidelity model at the low-fidelity level $f_L$ with the training dataset
    $\{(\mathbf{x}_i^L, y_i^L)\}_{i=1}^{N_L} \cup \{(\mathbf{x}_i^H, y_i^H)\}_{i=1}^{N_H}$
5 Train the multi-fidelity model at the high-fidelity level $f_H$ with the training dataset $\{((\mathbf{x}_i^H, y_i^H), y_i^H)\}_{i=1}^{N_H}$
```

Remark 9 (Markov property). Theoretically the observations $\{y_i^q\}$ should be noiseless for each level of fidelity q in order to preserve the Markov property [200]. However, in practice, it could be beneficial in some applications to add noise at each fidelity level, or constraint the noise levels from below in order to avoid overfitting.

3.5 Numerical results

In this section, we are going to present the results obtained with the NARGP-AS and the NARGP-NLL method over two benchmark test problems (Piston 3.5.1 and Ebola 3.5.1 models), and over a more complex car aerodynamics problem (Jetta-6 3.5.2, Jetta-12-RANS 3.5.2, Jetta-12-DDES 3.5.2). The library employed to implement the NARGP model is Emukit [190] while for the active subspace and NLL response surface design we used the open source Python package[1] called `ATHENA` [218], and GPy [98].

The computational times of the prediction and training of the NARGP-AS method are reported in Table 3.1. In particular, it is shown how the number HF test samples and of Monte Carlo (MC) samples affect the MF prediction times. The training costs are mainly affected by the number of restarts of the optimization with L-BFGD, instead.

Table 3.1 Computational times of the training of the multi-fidelity models and evaluation of the predictions. The label "pred" stands for prediction and "HF tr samples" stands for the number of high-fidelity training samples.

Test case	HF tr samples	training	restarts	MC samples	HF test samples	pred MF	pred HF
Piston model 3.5.1	150	24 [s]	10	100	10000	10 [s]	0.123 [s]
Ebola model 3.5.1	150	21 [s]	10	100	10000	10 [s]	0.450 [s]
Jetta-6 3.5.2	76	229 [s]	150	100	25	0.056 [s]	0.0006 [s]
Jetta-12-RANS 3.5.2	185	50 [s]	10	10000	51	14.2 [s]	0.0006 [s]
Jetta-12-DDES 3.5.2	65	20 [s]	10	1000	50	0.02 [s]	0.0003 [s]

3.5.1 Benchmark test problems

The first benchmark test problem presents a 7-dimensional input parameter space and the quantity of interest is the time a cylindrical piston takes to complete a cycle[2]. The second one is a 8-dimensional model for the spread of Ebola in Western Africa [78]. These tests have been chosen because of the presence of an active subspace and they indeed present a low intrinsic dimensionality. The sufficient summary plot is plotted for both cases together with a one-dimensional Gaussian process regression built over the AS. We also show the correlation between the low-fidelity level and the high-fidelity level of the multi-fidelity model. We compare the performance of the different fidelities by looking

[1]Available at https://github.com/mathLab/ATHENA.
[2]The piston dataset was taken from https://github.com/paulcon/active_subspaces.

at the corresponding R^2 scores. This score is chosen to show how the obtained regressions compare with respect to a constant predictor equal to the function average ($R^2 = 0$). With LF we denote the low-fidelity model represented by a GP regression on the low-fidelity input/output couples, with HF the high-fidelity model represented by a GP regression built on the full space, and with MF the proposed multi-fidelity model. The number of low-fidelity samples is kept fixed at 200 for both test cases, while we study the accuracy varying the number of high-fidelity training samples used. For both the benchmark problems the models were tested over a dataset comprising 10000 samples, selected with Latin hypercube sampling (LHS). The nonlinear autoregressive fidelity fusion approach achieves better performance with a consistent increase in the R^2 score.

The piston model

For this model, the scalar target function of interest represents the time it takes the piston to complete a cycle, depending on a 7-dimensional parameters vector. For its evaluation, a nonlinear function has to be computed. The input parameters are uniformly distributed. For a detailed description of the parameters' ranges, the reader can refer to [61]. The algebraic cylindrical piston model appeared as a test for statistical screening in [23], while in [61] they describe an active subspaces analysis.

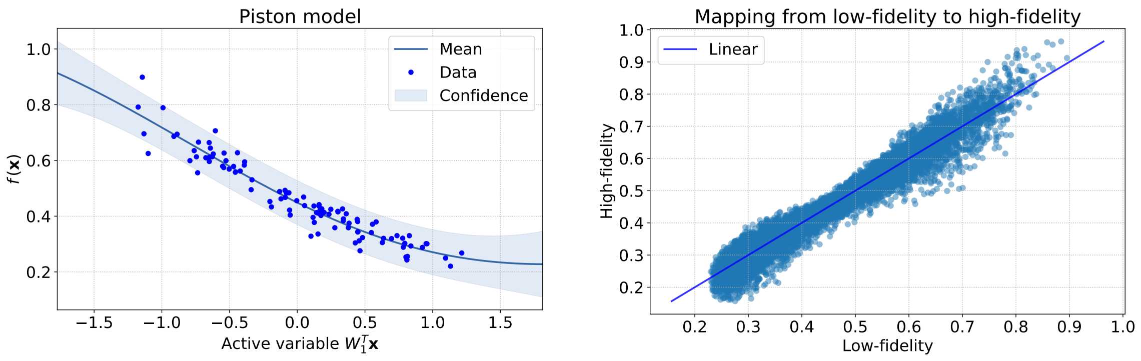


Fig. 3.2 Left: sufficient summary plot of the surrogate model built with active subspaces. 100 samples were used to build the AS surrogate model shown. Right: correlations among the low-fidelity level and the high-fidelity level of the multi-fidelity model, evaluated at the 10000 test samples.

From the sufficient summary plot reported in the left panel of Figure 3.2 we can conclude that a one-dimensional active subspace is able to describe the input-output dependency with sufficient accuracy. This is also supported by the GPR built over the AS. Moreover, the ordered eigenvalues of the covariance matrix of the gradients exhibit a spectral gap between the first and the second eigenvalue. In the right panel of Figure 3.2 we present the correlation between the high- and low-fidelity of the NARGP model.

Figure 3.3 shows on the left the mean R^2 scores of the MF model built as described in Section 3.4 varying the number of high-fidelity data. This is done over 10 training restarts of the MF, LF and HF

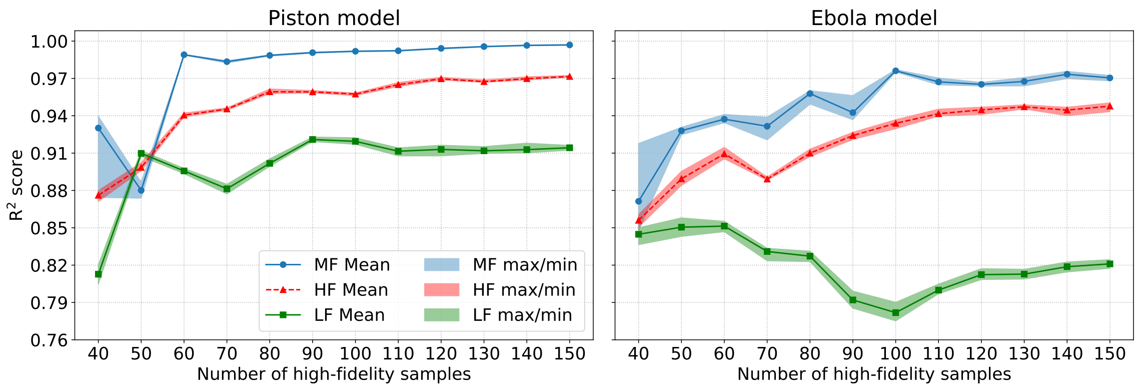


Fig. 3.3 R^2 score of the posterior of the multi-fidelity (MF), high-fidelity (HF) and low-fidelity (LF) models against the number of high-fidelity samples used to find the active subspace and build the Gaussian process regressions of the MF, HF, LF models. The 10000 test samples are distributed with Latin hypercube sampling (LHS). In the left panel the results for the piston model, while on the right the Ebola spread model.

models: moreover each GPR training is restarted 10 times for the HF and LF models and 20 times for the MF model at each fidelity level, inside the GPy package. We show also the minimum and maximum R^2 scores over the outer 10 training restarts to show the stability of the procedure. When we have a scarce amount of data the models are not so robust as we can see in the left part of the plot for 50 and 60 high-fidelity samples. After that point, we have very stable results which account for a relative gain in the 3–5% range with respect to the high-fidelity regression.

Modified SEIR model for Ebola

Now we consider the modified SEIR model for the spread of Ebola in Liberia, presented in [78], which depends on 8 parameters. As scalar output of interest we take the basic reproduction number R_0. It can be computed with the following formula:

$$R_0 = \frac{\beta_1 + \frac{\beta_2 \rho_1 \gamma_1}{\omega} + \frac{\beta_3}{\gamma_2}\psi}{\gamma_1 + \psi}, \tag{3.20}$$

with parameters range taken from [78], where they conducted a global sensitivity analysis with AS. For a kernel-based active subspaces comparison the reader can refer to [221].

In this case, a one-dimensional Gaussian process response surface is not able to achieve the same good accuracy of the previous case, as can be seen in the left panel of Figure 3.4. This is also confirmed by the correlation between the low- and high-fidelity levels of the NARGP, depicted in the right panel of Figure 3.4. The corresponding R^2 scores in the right panel of Figure 3.3 reflect this behavior of worse performance with respect to the piston test case, where better correlations among

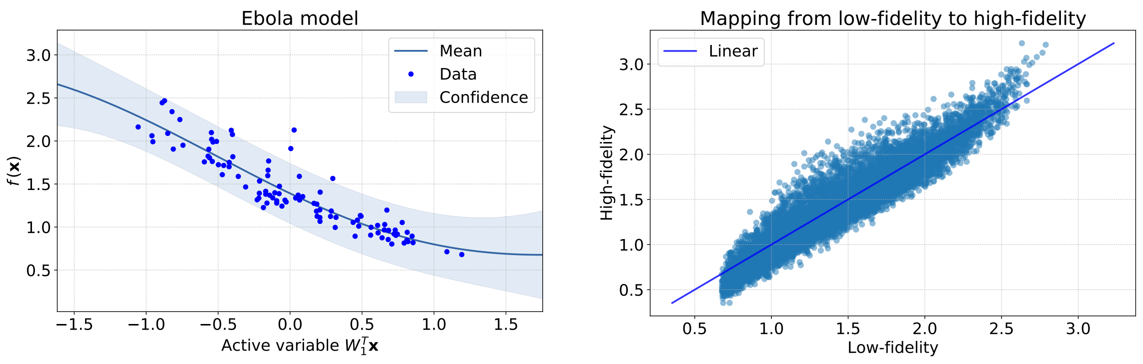


Fig. 3.4 Left: sufficient summary plot of the Ebola model, 100 samples were used to build the AS surrogate model shown. Right: correlations among the low-fidelity level and the high-fidelity level of the multi-fidelity model, evaluated at the 10000 test samples.

the fidelities were identified. The relative gain is in the 3–4% range with respect to the high-fidelity regression.

3.5.2 Automotive application

Two different test cases from the world of automotive aerodynamics are investigated in order to demonstrate the applicability of the presented method to real-life problems. The first one (named hereafter **Jetta-6**) is taken from [180], where it is described in detail. It consists of a 6-dimensional geometric parameterization of the Volkswagen Jetta VI. The parameters (see Table 3.2) were generated by free-form deformation and focus on the rear part of the car. A Latin Hypercube with 101 samples was created, and the aerodynamic flow fields were computed with `OpenFoam` [264] via Delayed Detached Eddy Simulations (DDES). An illustrative example can be seen in Figure 3.5. The physical simulation time was four seconds, and the fields were averaged over the last two seconds before integrating them over the vehicle surface to obtain the drag coefficient c_D. With mesh sizes being of the order of 100M cells, each variant required about 23,000 CPU-core-hours.

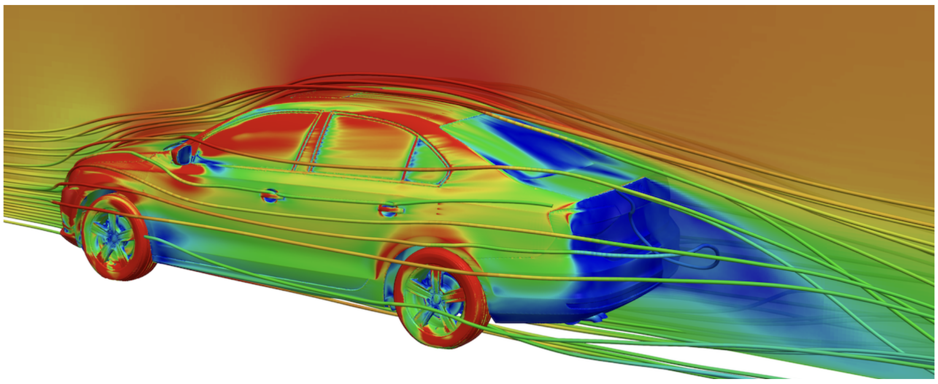


Fig. 3.5 Visualization of the averaged flow field around the Jetta.

Table 3.2 Parameters' description of the Jetta-6 test case [180].

Parameter	Description	Lower bound	Upper bound
$\boldsymbol{\mu}_1$	Rear roof lowering	0 mm	50 mm
$\boldsymbol{\mu}_2$	Trunk height	-30 mm	30 mm
$\boldsymbol{\mu}_3$	Trunk length	-50 mm	100 mm
$\boldsymbol{\mu}_4$	Rear lateral tapering	-60 mm	50 mm
$\boldsymbol{\mu}_5$	Rear end edge position	-70 mm	30 mm
$\boldsymbol{\mu}_6$	Rear end depression	-15 mm	0 mm

The second automotive test case (named hereafter **Jetta-12**) is based on the same car model and was created within the EC project UPSCALE [2]. The parameterization consists of 12 geometric modifications all around the vehicle (see Figure 3.6 and Table 3.3). Besides the baseline geometry, a Sobol sequence of 300 additional samples was created and computed with `OpenFoam`. To reduce the required computational budget to an affordable amount, Reynolds-Averaged-Navier-Stokes (RANS) computations were carried out instead of DDES runs. This allowed to use coarser meshes of 52M cells and resulted in 1700 CPU-core-hours for a single run for the 4000 iterations, of which the last 1000 were averaged to obtain the drag coefficient c_D.

Table 3.3 Parameters' description of the Jetta-12 test case.

Parameter	Description	Lower bound	Upper bound
$\boldsymbol{\mu}_1$	Spoiler Y Angle	-5.0°	0.0°
$\boldsymbol{\mu}_2$	Spoiler Slide Translation	0.0 mm	30.0 mm
$\boldsymbol{\mu}_3$	Tail Light Y Span	-15.0 mm	5.0 mm
$\boldsymbol{\mu}_4$	Tail Light X Translation	-10.0 mm	10.0 mm
$\boldsymbol{\mu}_5$	Rear Window X Translation	-100.0 mm	100.0 mm
$\boldsymbol{\mu}_6$	Rear Window Z Translation	-30.0 mm	0.0 mm
$\boldsymbol{\mu}_7$	Rear End Taper Ratio	-1.0°	3.0°
$\boldsymbol{\mu}_8$	Front Window X Translation	-100.0 mm	100.0 mm
$\boldsymbol{\mu}_9$	Front Window Z Translation	-30.0 mm	0.0 mm
$\boldsymbol{\mu}_{10}$	Rear End Z Translation	-30.0 mm	30.0 mm
$\boldsymbol{\mu}_{11}$	Grill Slide Translation	-50.0 mm	50.0 mm
$\boldsymbol{\mu}_{12}$	Bumper Y Translation	-20.0 mm	20.0 mm

In Figure 3.7 we depicted the eigenvalues decay for the automotive test cases. The largest spectral gap is always between the first and the second eigenvalue. This justifies the choice of a low-fidelity model built from a one-dimensional regression.

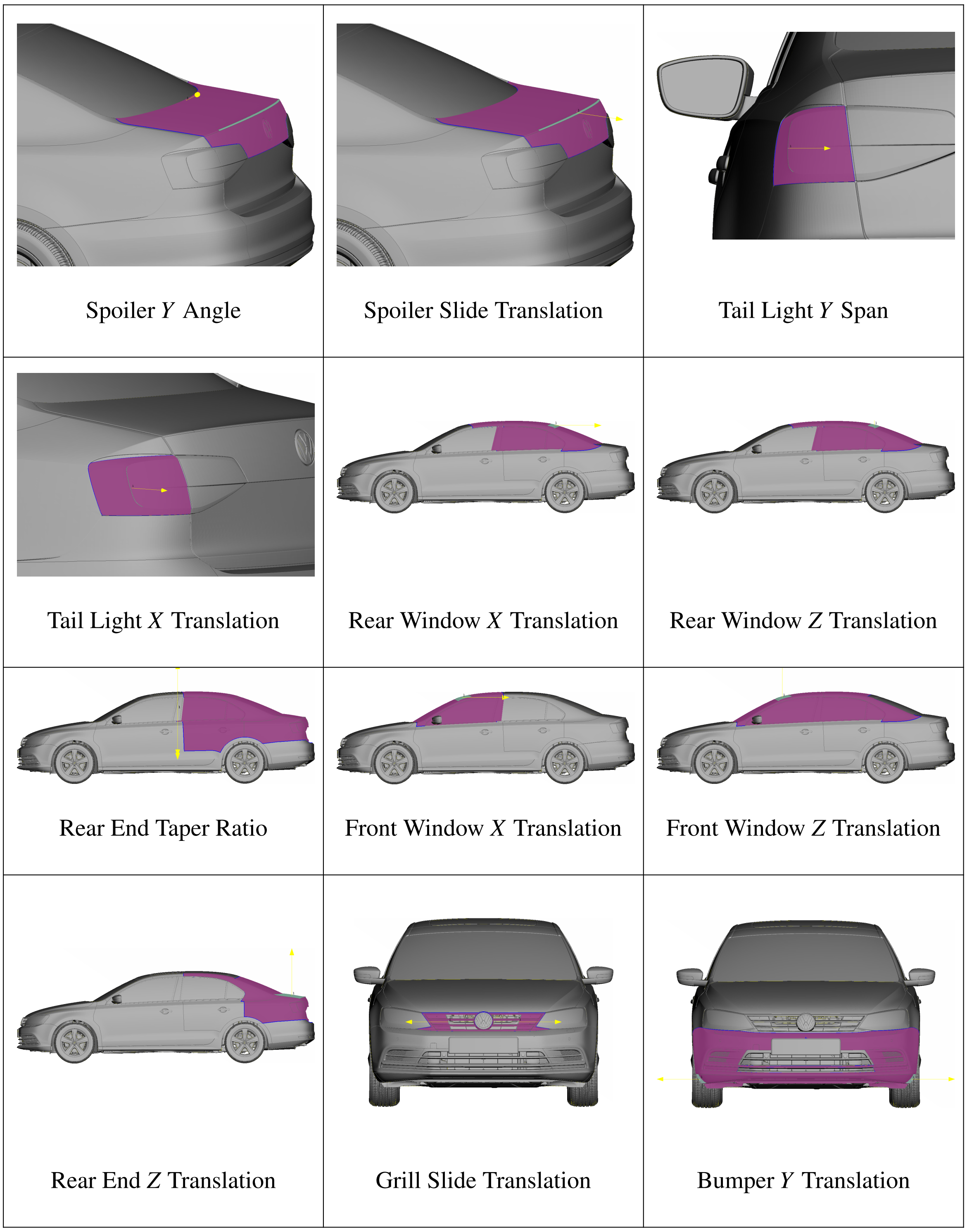

Spoiler Y Angle
Spoiler Slide Translation
Tail Light Y Span
Tail Light X Translation
Rear Window X Translation
Rear Window Z Translation
Rear End Taper Ratio
Front Window X Translation
Front Window Z Translation
Rear End Z Translation
Grill Slide Translation
Bumper Y Translation

Fig. 3.6 Affected areas by the geometrical parameters for the Jetta-12 test case. The ranges of each parameter can be gleaned from Table 3.3.

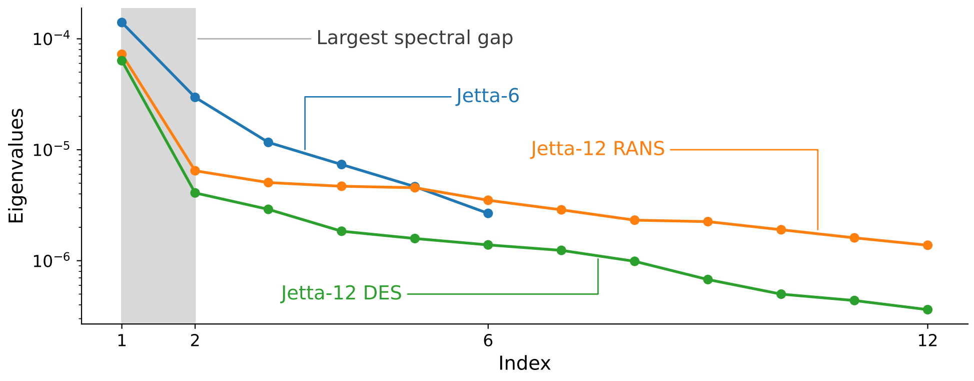


Fig. 3.7 Eigenvalues decay of the covariance matrix of the gradients for the Jetta-6 and Jetta-12 test cases.

Multi-fidelity response surface design Jetta-6

In this test case, the low-fidelity model chosen is the response surface trained on the active latent variables obtained with the NLL method: instead of prolonging along the orthogonal directions the one-dimensional regression built on the active subspace, a GPR is trained on the deformed high-fidelity inputs $\{g_{\text{NLL}}(\mathbf{x}_i^H)\}_{i=1}^{N_H} \subset \tilde{\mathcal{X}}$. We remark that the map g_{NLL} does not preserve in general convexity of the domain $\mathcal{X}$ or orthogonality of the boundaries. Nonetheless, this is not problematic for this application since we are not interested in backmapping the active latent variables from $\tilde{\mathcal{X}}$ to $\mathcal{X}$, but only in forwarding the inputs from $\mathcal{X}$ to $\tilde{\mathcal{X}}$ and than evaluating the predictions with the GPR.

The employed RevNet has 10 layers. It was trained for 20000 epochs on a dataset of 76 training samples and 25 test samples, with ADAM stochastic optimization method [147], with an initial learning rate of 0.03. The high-fidelity samples were obtained with LHS method. The architecture is implemented in PyTorch [196] inside the `ATHENA` [218] Python package. We perform a study on the number of additional LF samples, distributed uniformly on the domain, from 100 to 400 with a step of 50. The results are shown in Figure 3.8.

The maximization of the log-likelihood is performed with 10 restarts for the HF and LF models, and 100 restarts for the MF model, all inside GPy optimization algorithm. All training procedures are moreover restarted 10 times, testing the stability of the optimization process for each fidelity model. This is done in order to show, in Figure 3.8 with blue lines, that the MF training presents some small instabilities with respect to the HF and LF training, as expected. The LF and HF models are designed over the same HF inputs-outputs datasets, so they are not influenced by the additional LF samples.

Remark 10 (Reversing the fidelities order). A natural question that may arise regards the correct ordering of the HF and LF models in the MF when the accuracy is higher for the LF as in Figure 3.8. We perform a study with respect to the number of additional samples from the HF GPR (not from the numerical simulations), now the lowest fidelity in the MF model. Moreover, in order to reach a desirable accuracy, we add to each of the 2 levels of fidelity of the MF model 200 uniformly

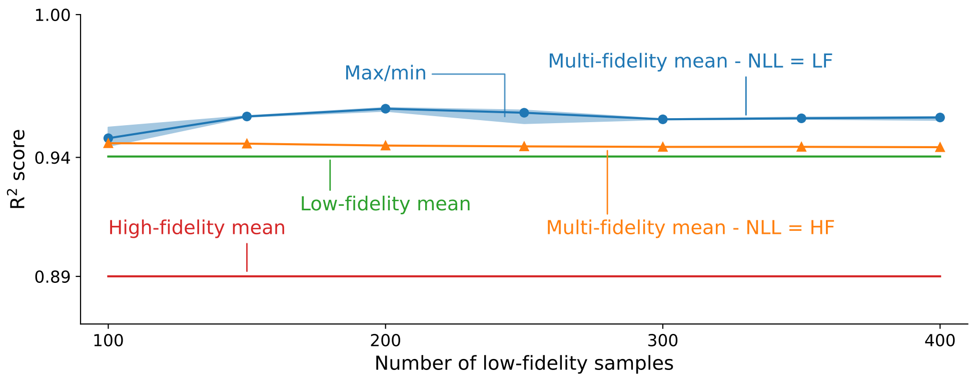


Fig. 3.8 R^2 score evaluated on the 25 test samples obtained from LHS on the domain $\mathcal{X}$, varying the number of LF samples. The mean R^2 score over 10 restarts of the training of the GPR is shown. For the MF also the minimum and maximum values are reported. The orange line identifies the results obtained by reversing the fidelities order, so the number of LF samples corresponds to the HF GPR. The LF and HF R^2 scores are not influenced by the number of additional LF samples.

sampled input-output pairs: the highest fidelity is the NLL GPR built with $76+200$ training data; the lowest fidelity is the HF GPR built with training data equal to 76 from numerical simulations $+200$ fictitiously from the HR GPR (not from numerical simulations) $+$ additional samples from 100 to 400 with a step of 50 from HF GPR (not numerical simulations). The results are reported in Figure 3.8 with orange lines (NLL = HF). The R^2 score is lower than the previous case. Generally, the ordering of the fidelities depends on the availability of data and the cost of obtaining them.

We also perform cross-validation (CV) with leave-one-out strategy for the Jetta-6 test case to assess the robustness of the result with respect to the test dataset, in Figure 3.9. We reported the mean and confidence intervals at 95% among the 25 batches of the leave-one-out strategy for a test set of 25 samples: each batch has 24 test samples. For each abscissa, the batches corresponding to the lowest R^2 score for the MF and highest R^2 score for the LF are found, so that with respect to these two selected batches the R^2 scores of the LF and MF models, respectively, can be computed and compared: we want to remark that batch-wise the MF R^2 score is always higher to the LF R^2 score.

Multi-fidelity response surface design Jetta-12

For this test case with additional 6 parameters with respect to the previous one, for a total of 12, a one-dimensional NLL response surface does not perform better than a one-dimensional AS response surface, so we preferred the latter as LF model. In this case, we also added Gaussian noise at each fidelity level in order to achieve a better accuracy, losing the Markov property, see Remark 9. Moreover, to avoid overfitting we restricted the variance of the Gaussian noise to the interval $[0.01, 0.1]$ at each fidelity of the multi-fidelity model.

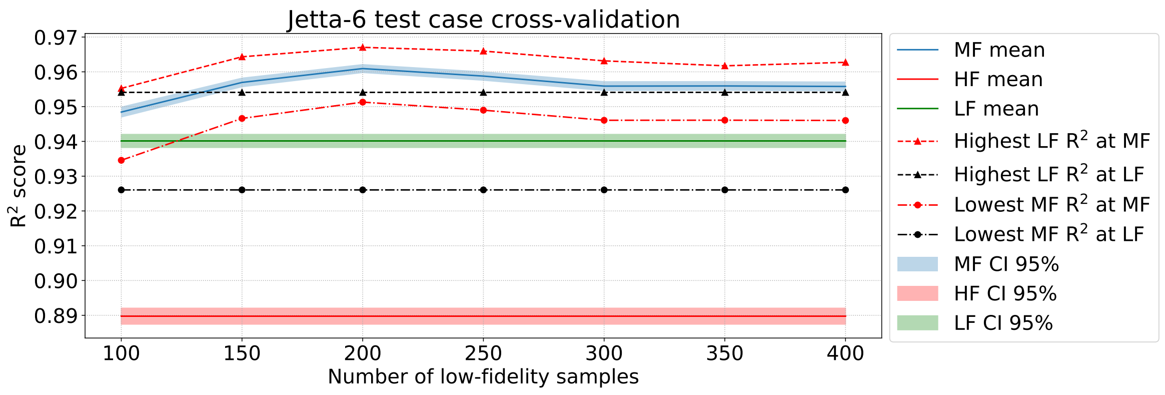


Fig. 3.9 Cross-validation with leave-one-out strategy and confidence bounds at 95%. The labels **lowest MF R^2 at LF** stands for the R^2 score of the batch associated with the lowest R^2 score for the MF model in the CV procedure, but evaluated at the predictions of the LF model. The other labels are analog.

We perform a study on the number of high-fidelity samples from 45 to 225, obtained from a Sobol’ sequence. The test set has 51 samples obtained with LHS instead. The number of additional LF samples is 100. The results are reported in Figure 3.10. As for the Jetta-6 test case, we perform 10 outer training restarts for the LF, HF, and MF models: the 100 additional LF samples are resampled every time. Moreover, the optimization procedures of the GPRs are restarted 10 times for the LF, HF, and MF model. We employed also a validation dataset of additional independent 25 samples from the continuation of the Sobol’ sequence: the markers in Figure 3.10 correspond to the best HF and MF models with respect to the validation set. We also report maximum and minimum R^2 scores for the outer loop training restarts of the MF model to show that the validation process is fairly effective, at least when employing 45 to 155 high-fidelity samples. We emphasized in the plot the three distinct areas corresponding to the scarce data, low data, and abundance of data regimes.

It can be seen a gain of around 4% on average on the R^2 score of the MF model, with respect to the other two, in the abscissae range from 45 to 155. This time the procedure is much less stable with respect to the optimization process, probably due to the higher dimension of the input space. The decreasing behaviour of the R^2 score of the MF models from the abscissa 135 to 225 can be ascribed to the prevalence of the HF model: in this case the LF model influences less the predictions of the MF model, which are more stable and close to the HF ones. The low HF R^2 score at abscissa 55 is almost constant for each outer training step and is not related to overfitting, but it can be associated with a high sensitivity of the regression when employing a small dataset relative to the problem at hand.

We want also to remark that in this test case the training of the LF, HF, and MF models takes less than 10 minutes for each number of high-fidelity samples, with increasing costs from 45 to 225 samples, considering altogether the outer loop training restarts. Compared with the costs for a high-fidelity simulation, the MF training cost is negligible.

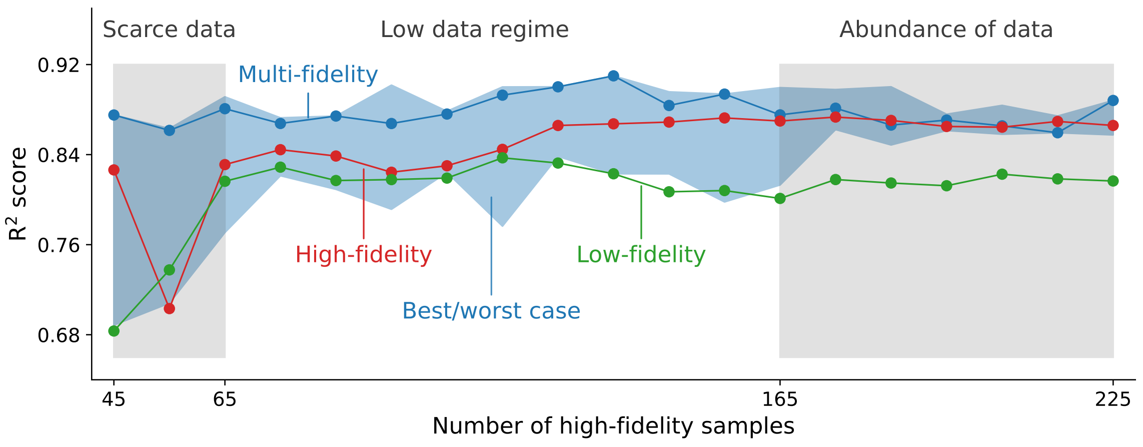


Fig. 3.10 Jetta-12: R^2 score evaluated on the 51 test samples obtained from LHS on the domain $\mathcal{X}$, varying the number of HF samples. The mean R^2 score over 10 restarts of the training of the GPR is shown. For the MF model the minimum and maximum values are shown, differently from the HF and LF models, since the perturbations are not sensible. The markers associated with the MF and HF models represent the R^2 scores on the test set of the best HF and MF models with respect to an independent validation set of 25 samples.

High-fidelity model choice for Jetta-12

In principle, any model of the same phenomenon originating from a different physical approximation, numerical method, or discretization, can be employed to produce a multi-fidelity model. In the case of the Jetta-12 automotive test case, computations can be carried out with the more accurate DDES runs, as in the Jetta-6 test case. Then, we have 3 models at our disposal: the response surfaces built on the DDES outputs, RANS outputs, or AS predictions.

We train the DDES-AS and DDES-RANS two-fidelity models as described in Section 3.2, and consider also the DES and AS single fidelity models. We compute 75 DDES training input-output pairs, and 50 DDES test input-output pairs, both sampled with LHS. Since the DDES simulations represent the highest fidelity, the DDES test samples will be used to evaluate the R^2 scores of all the other single and multi-fidelity models considered. All the 300 RANS training data available from the previous test case will be employed for the DDES-RANS model, and 100 additional LHS sampled input-output pairs obtained from the AS response surface will be used to train the DDES-AS two-fidelity model.

The R^2 errors on the test set are reported in Figure 3.11. Also in this case we use cross-validation with leave-one-out and leave-two-out strategy to assess the robustness of the results with respect to the test set: we show the mean, minimum, maximum, and standard deviation (std) with respect to the sets of $50 = \binom{50}{1}$ and $1225 = \binom{50}{2}$ cross-validation batches. When the two models are integrated into the DDES-AS MF model, the accuracy sensibly rises as observed previously. Only for the MF model, we extracted 10 validation samples from the 75 training dataset, so we trained it with exactly 65 samples and selected the best model looking at the R^2 score of the validation set. The effectiveness of the validation procedure is shown in Figure 3.12. Also in this case, we constrained the Gaussian noise levels of the MF model to belong to the interval $[0.01, 0.1]$.

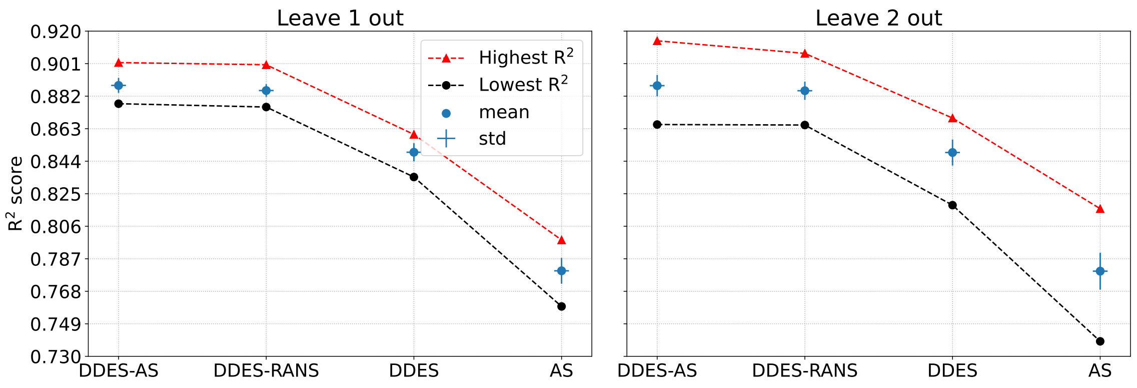


Fig. 3.11 Cross-validation (CV) with leave-one-out (Left) and leave-two-out strategy (Right) on the test set. The labels **Lowest $\mathbf{R^2}$** and **Highest $\mathbf{R^2}$** stand for the R^2 score of the batch associated with the lowest and highest R^2 score, respectively. The mean and standard deviation shown (std) are computed with respect to the sets of $50 = \binom{50}{1}$ and $1225 = \binom{50}{2}$ cross-validation batches, respectively.

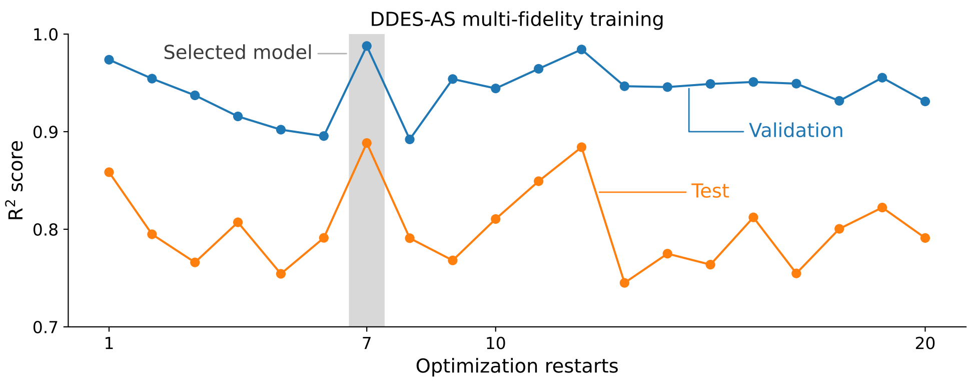


Fig. 3.12 Validation process of the DDES-AS multi-fidelity model over 20 outer training restarts changing every time the additional 100 additional low-fidelity samples. The selected multi-fidelity model corresponds to abscissa 7.

The accuracy is comparable to the DDES-RANS MF model, implying that the AS response surface can indeed be used as a low-fidelity purely data-driven model in the process of design of a multi-fidelity model, along with more standard models based on different physical or numerical approximations of the phenomenon under study. It must be said that the RANS outputs are poorly correlated with respect to the DDES as can be seen from Figure 3.14: in fact the converged GPR built

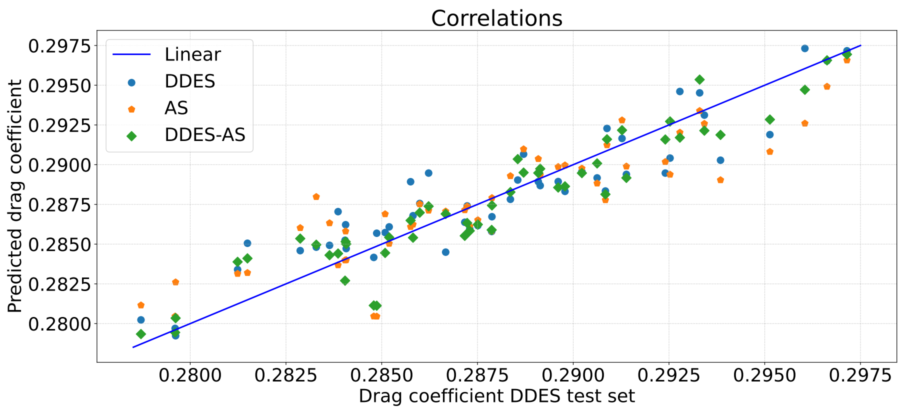


Fig. 3.13 Comparison of the correlations between the predictions of the DDES model with the test DDES outputs and the correlations between the DDES-AS MF model with the test DDES outputs.

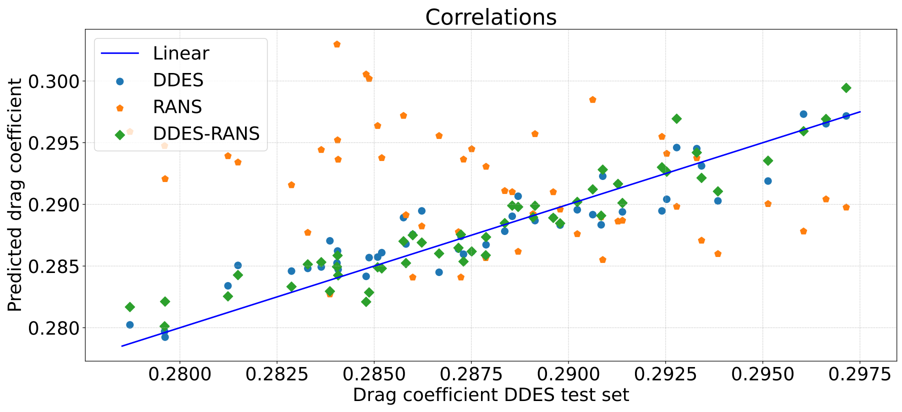


Fig. 3.14 Comparison of the correlations between the predictions of the DDES model with the test DDES outputs, the correlations between the RANS model with the test DDES outputs, and the correlations between the DDES-RANS model with the test DDES outputs.

upon the RANS training dataset have a mean R^2 score below 0 on the DDES test set. Nonetheless, the multi-fidelity model DDES-RANS achieves an accuracy higher than the single-fidelity DDES model.

3.6 Conclusions and future perspectives

The approximation of high-dimensional scalar quantities of interest is a challenging problem in the context of data scarcity, which is typical in engineering applications. We addressed this problem by proposing a nonlinear multi-fidelity method that does not necessitate the simulation of simplified models, but instead constructs a low-fidelity surrogate introducing a low-intrinsic dimensionality bias through active subspaces or nonlinear level-set learning methods. Our approach is data-efficient since it extracts new information from the high-fidelity simulations. We construct different Gaussian processes using the autoregressive scheme called NARGP. The proposed multi-fidelity approach results in better approximation accuracy over the entire parameter space as demonstrated with two benchmark problems and an automotive application.

NARGP-AS was able to achieve better performance with respect to the single-fidelity GP over the high-fidelity data, resulting in a relative gain on the R^2 score around 3–5% for the piston model, and around 3–4% for the Ebola model, depending on the number samples used. NARGP-NLL was used for the Jetta-6 test case, reaching an accuracy gain around 2% with respect to the low-fidelity model, and around 4% with respect to the high-fidelity model. We also presented a comparison switching the two fidelities. Finally, for the Jetta-12 test case we obtained a relative gain on the R^2 score around 3%.

Future research lines should investigate the use of different active subspaces-based methods, such as kernel AS [221], or local AS [220], which exploit kernel-based and localization techniques, respectively. This multi-fidelity framework has also the potential to be integrated with other reduced order modelling techniques [227, 54, 228] to further increase the accuracy in the resolution of parametric problems, especially for high-dimensional surrogate-based optimization [252].

Mandatory for real applications is a model management strategy providing theoretical guarantees and establishing accuracy and/or convergence of outer-loop applications. Some attempts towards multi-source Bayesian optimization/Experimental design are being studied. Moreover increasing the number of fidelities in the multi-fidelity model is a possible direction of investigation, especially when the phenomenon of interest allows many cheap low-fidelity approximations.

Chapter 4

Constrained Generative Models

Real-world applications of computational fluid dynamics often involve the evaluation of quantities of interest for several distinct geometries that define the computational domain or are embedded inside it. For example, design optimization studies require the realization of response surfaces from the parameters that determine the geometrical deformations to relevant outputs to be optimized. In this context, a crucial aspect to be addressed is the limited resources at disposal to computationally generate different geometries or to physically obtain them from direct measurements. This is the case for patient-specific biomedical applications for example. When additional linear geometrical constraints need to be imposed, the computational costs increase substantially. Such constraints include total volume conservation, barycenter location and fixed moments of inertia. It is developed a new paradigm that employs generative models from machine learning to efficiently sample new geometries with linear and multilinear constraints. A consequence of our approach is the reduction of the parameter space from the original geometrical parametrization to a low-dimensional latent space of the generative models. Crucial is the assessment of the quality of the distribution of constrained geometries obtained with respect to physical and geometrical quantities of interest. Non-intrusive model order reduction is enhanced since smaller parametric spaces are considered. The methodology is tested on two academic test cases: a mixed Poisson problem on the 3d Stanford bunny with fixed barycenter deformations and the turbulent Reynolds Averaged Navier-Stokes equations on the Duisburg test case with fixed volume deformations of the naval hull.

Contents

4.1 Literature review

In the last years, there has been an increasing interest in using Deep Neural Networks to approximate distributions of 3D objects [246, 263, 52, 243, 266] via generative models. In different contexts, new methods that define deformation maps of 3d objects with the preservation of some geometrical properties, like the volume [129, 262, 4, 114, 83], are studied. The possibility to enforce geometrical constraints on the domains of computational fluid dynamics simulations is of great interest, especially for industrial and real-world applications. For example, in naval engineering, the fast generation of geometries of naval ship's hulls such that the volume of the submerged part is preserved is fundamental for ydrodynamic stability. On the other hand, biomedical applications that involve patient-specific numerical models often struggle to obtain new valid geometries that also preserve some geometrical properties of interest since the experimental data are scarce.

We try to close this gap by implementing constrained generative models (cGMs) that are able to reproduce distributions of constrained free from deformations (cFFD) [230]. The main advantage of our novel methodology is that the computational costs are substantially reduced in the predictive phase with respect to classical numerical methods that define volume-preserving maps for example. Another great advantage is the dimension reduction of the space of parameters from the original space of deformations to the latent space of generative models.

Constrain preserving deformations One of the first works that introduce volume-preserving deformations in computer graphics is Hirota et al. [129] who developed a variation of Free Form Deformation that conserves the volume of the original domain exploiting an augmented Lagrangian formulation. This argument has been developed further by Hahmann et al.[114], that take a restriction on the possible deformations and obtain an explicit solution using a sequence of three quadratic programming problems. Von funck [262] et al. (2006) propose a method for creating deformations using path line integration of the mesh vertices over divergence-free vector fields, thus preserving the volume. Eisemberg et al. [83] extend this method for shape interpolation, using the eigenvectors of the Von Neumann-Laplace equation; the interpolation is done using the Karhunen-Loéve expansion. Cerverò et al. [4] obtain a volume-preserving method using a cage-based deformation scheme with generalized barycenter coordinates. They also propose a measure of the local stress of the deformed volume.

Generative models for 3D meshes In the last years, there has been an increasing number of new generative models' architectures for 3D mesh deformation. Qtan et al.[243] develop a Variational Autoencoder in which the input data is encoded into a rotation invariant mesh difference representation, which is based on the concept of deformation gradient and it is rotation invariant. They also propose an extended model in which the prior distribution can be altered. Ranjan et al.[213] develop a Convolutional Mesh Autoencoder (CoMA) which is based on Chebyscev Spectral Convolution and on a new sampling method which is based on the hierarchical mesh representation. Rana Hanocka et al. [120] introduce some new original convolutional and pooling layers for mesh data points based on their intrinsic geodesic connections. Yuan et al. [279] extend the previous works, by implementing a variational autoencoder that works for meshes with the same connectivity but supported on different geometries. Hahner et al. [115] develop an autoencoder model for semiregular meshes with different sizes, which have a locally regular connectivity and hierarchical meshing. They are also able to apply the same autoencoder to different datasets. Cheng et al.[52] develop a new mesh convolutional GAN model based on [27].

The work is organized as follows. In section 4.2, we describe the classical free form deformation method along with its variant to impose linear and multilinear constraints in subsection 4.2.2. In section 4.3, the generative models we are going to use are presented. In subsection 4.3.2, our novel approach to enforce linear and multilinear constraints on generative models is introduced. Since we also show how model order reduction can be performed efficiently in this context, a brief overview of the non-intrusive proper orthogonal decomposition with interpolation method (PODI) is described in section 4.4 along with radial basis functions interpolation to deform the computational meshes. We test the new methodology on two academic benchmarks in section 4.5: a mixed Poisson problem for the 3d Stanford bunny [257] **SB** with fixed barycenter deformations in section 4.5.1 and the incompressible Reynolds' Averaged Navier-Stokes equations with volume-preserving deformations of the Duisburg's test case [268] **HB** naval bulb in section 4.5.2.

4.2 Constrained Free Form Deformation

Free Form Deformation (FFD) was introduced in [230]. It is successfully employed in optimal shape design [249] as a technique to geometrically parametrize the domain through a basis of Bernstein polynomials and a set of bounding control points. When applied to deform computational meshes, some challenges related to FFD, include the preservation of the regularity of the mesh for relatively large deformations, the possibly high dimensional parameter space employed and the loss of conservation of geometrical or physical quantities of interest such as the volume. In this work, we address the last two points.

4.2.1 Free Form Deformation

We consider the FFD of 3d shapes in $\mathbb{R}^3$. Let $D = [0,1]\times[0,1]\times[0,1] \subset \mathbb{R}^3$ be the bounding box inside which the deformation is performed and $P = \{\mathbf{P}_{ijk}\}_{i,j,k=1}^{m,n,o} = \{\left[\frac{i}{m}, \frac{j}{n}, \frac{k}{o}\right]\}_{i,j,k=1}^{m,n,o}$ the lattice of control points with $m,n,o > 0$.

The bases employed for the interpolation are the trivariate Bernstein polynomials $\{B_{ijk}^{\lambda\mu\nu}\}_{i,j,k=0}^{\lambda\mu\nu} \subset \mathbb{P}^{\boldsymbol{\alpha}}(D)$ of polynomial degree $\boldsymbol{\alpha} = (\lambda, \mu, \nu) \in \mathbb{N}^3$ and support $D \subset \mathbb{R}^3$ defined from the one-dimensional basis $\{b_s^\kappa\}_{s=0}^\kappa \subset \mathbb{P}^\kappa([0,1])$:

$$b_s^\kappa(x) = \binom{\kappa}{s} x^s(1-x)^{\kappa-s}, \qquad \{b_s^\kappa\}_{s=0}^\kappa \subset \mathbb{P}^\kappa([0,1]), \tag{4.1}$$

$$B_{ijk}^{\lambda\mu\nu}(u,v,w) = b_i^\lambda(u) b_j^\mu(v) b_k^\nu(w), \qquad \{B_{ijk}^{\lambda\mu\nu}\}_{\nu=0}^{n} \subset \mathbb{P}^{\boldsymbol{\alpha}}(D). \tag{4.2}$$

Other bases such as B-splines can be employed [114].

Given the set of displacements of the control points $\delta P = \{\delta\mathbf{P}_{ijk}\}_{i,j,k=1}^{m,n,o}$, the deformation map $T_P : D \subset \mathbb{R}^3 \to \mathbb{R}^3$ is defined as follows:

$$T_Q(u,v,w) = (u,v,w) + \sum_{i,j,k=0}^{m,n,o} b_i^m(u) b_j^n(v) b_k^o(w) \delta\mathbf{P}_{ijk}, \quad \forall (u,v,w) \in D \subset \mathbb{R}^3 \tag{4.3}$$

mapping the domain $D = [0,1]\times[0,1]\times[0,1] \subset \mathbb{R}^3$ onto the bounding box of control points $T_P(D) = K$. The displacement field defined through the displacements of the control points δP is interpolated inside $D \subset \mathbb{R}^3$ by the second term of Equation (4.3).

To apply the FFD deformation, represented by the map T_P, to arbitrary 3d meshes, point clouds (CAD or STL files) or general subdomains $\Omega \subset \mathbb{R}^3$, an affine map $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ is evaluated to map the subdomain $D \subset \mathbb{R}^3$ into a parallelepiped $\tilde{K} \subset \mathbb{R}^3$ that intersects such 3d meshes, point clouds or general subdomains, $\tilde{K} \cap \Omega \neq \varnothing$. In this way, we define the FFD map as the composition $\tilde{T}_P = \varphi \circ T_P \circ \varphi^{-1} : \tilde{K} \cap D \subset \mathbb{R}^3 \to \mathbb{R}^3$:

$$\begin{tikzcd}
\tilde{K} \cap \Omega \subset \mathbb{R}^3 \arrow[d, "\varphi^{-1}"] & \varphi(K) \subset \mathbb{R}^3 \\
D \subset \mathbb{R}^3 \arrow[r, "T_P"] & T(D) = K \subset \mathbb{R}^3 \arrow[u, "\varphi"']
\end{tikzcd}$$

notice that, generally, the deformation can affect the boundaries $\partial\left(\tilde{K} \cap \Omega\right)$ of the intersection $\tilde{K} \cap \Omega \subset \mathbb{R}^3$ and not only its interior. We will thus employ it to deform 3d objects that will be embedded afterward into a computational mesh, see section 4.4. The control points $P \subset D \subset \mathbb{R}^3$ are mapped by φ into $\tilde{P} = \{\varphi(\mathbf{P}_{ijk})\}_{i,j,k=1}^{m,n,o} \subset \tilde{K} \cap \Omega \subset \mathbb{R}^3$ and deformed from $\tilde{P} \subset \Omega$ to $\tilde{P} + \delta\tilde{P}$, that is $\forall i \in$

$\{0,\dots,m\}, j \in \{0,\dots,n\}, k \in \{0,\dots,o\}$:

$$\tilde{T}_P(\tilde{\mathbf{P}}_{ijk}) = \tilde{\mathbf{P}}_{ijk} + \delta\tilde{\mathbf{P}}_{ijk}, \qquad \delta\tilde{\mathbf{P}}_{ijk} = \sum_{i,j,k=0}^{m,n,o} b_i^m(u) b_j^n(v) b_k^o(w) A_\varphi(\delta\mathbf{P}_{ijk}), \tag{4.4}$$

where $A_\varphi \in \mathbb{R}^3 \times 3$ is the matrix of the affine transformation $\varphi(\mathbf{x}) = A_\varphi \mathbf{x} + b_\varphi$ for all $\mathbf{x} \in \mathbb{R}^3$. In practice, we only need to fix the control points $\tilde{P}$ and their deformations $\delta\tilde{P}$. For this basic form of FFD we use the open-source Python package [251].

Fixed the control points $\tilde{P}$ and the displacements $\delta\tilde{P}$, the map $\tilde{T}_P : \tilde{K} \cap D \subset \mathbb{R}^3 \to \mathbb{R}^3$ can also be defined similarly to what has been done previously. We make explicit the dependency on the set of displacements δP with the notation

$$\tilde{T}_P(\mathbf{Q}, \{\delta\mathbf{P}_{ijk}\}_{ijk=0}^{m,n,o}) = \mathbf{Q} + \sum_{i,j,k=0}^{m,n,o} b_i^m(Q_x) b_j^n(Q_y) b_k^o(Q_z) A_\varphi(\delta\mathbf{P}_{ijk}), \qquad \tilde{T}_P(\mathbf{Q}, \delta P) : \mathbb{R}^3 \times \mathbb{R}^{(m\cdot n\cdot o)\times 3} \to \mathbb{R}^3 \tag{4.5}$$

with $\mathbf{Q} = (Q_x, Q_y, Q_z) \in \mathbb{R}^3$.

4.2.2 Free Form Deformation with multilinear constraints

One of the main objectives of this work is to approximate distributions of point clouds along with some of their geometrical properties such as the volume or the barycenter if they are kept constant. If we want to employ generative models for this purpose, in section 4.3, we must first collect a dataset of geometries that satisfy the constraint we want to impose.

Regarding volume-preserving deformations, many strategies involve the definition of divergence-free vector fields [262, 83]. Other volume-preserving methods are directly applied to FFD, like [129] with an augmented Lagrangian nonlinear multi-level formulation and [114] with a least-squares formulation. We will follow the last work for the simple closed-form enforcement of linear and multilinear constraints. Multiple linear constraints can be enforced at the same time as well with this method. In general, for the whole methodology presented in section 4.3 to work, only the constraint itself needs to be linear, how the training dataset representing the distribution to be approximated with GM is obtained is not relevant. In section 4.3 are reported other linear and multilinear constraints that could be imposed apart from the volume and the barycenter that are effectively conserved with our procedure as shown in the results section 4.5.

Let us suppose we are given a constraint that is linear with respect to the displacements $\delta P = \{\delta\mathbf{P}_{ijk}\}_{i,j,k=1}^{m,n,o}$. Chosen a subset of N points $Q \in \mathbb{R}^{N\times 3}$ of the undeformed subdomain $Q = \{\mathbf{Q}_i\}_{i=0}^N \subset \Omega \subset \mathbb{R}^3$ $\Omega \subset \mathbb{R}^3$ with $\mathbf{Q}_i \in \mathbb{R}^3$, $\forall i \in \{0,\dots,N\}$, we can write the linear constraint as

$$\mathbf{c} = A_c \mathrm{vec}(\tilde{T}_P(Q, \delta P)) \tag{4.6a}$$

where the displacements δP of the control points P have been made explicit through the notation of equation (4.5) and $\mathrm{vec}(\tilde{T}_P) \in \mathbb{R}^{3N}$ is the rowwise vectorization of the matrix $\tilde{T}_P \in \mathbb{R}^{N\times 3}$. We have introduced the vector $\mathbf{c} \in \mathbb{R}^n$ and the matrix $A_c \in \mathbb{R}^{n\times 3N}$ representing the linear constraint. We can expand the previous equation in

$$\mathbf{c} = A_c \text{vec}(\tilde{T}_P(Q, \delta P)) = A_c \text{vec}(Q) + A_c \text{vec}(\delta Q), \qquad \delta \mathbf{Q}_l = \sum_{i,j,k=0}^{m,n,o} b_i^m(Q_x^l) b_j^n(Q_y^l) b_k^o(Q_z^l) A_\varphi(\delta \mathbf{P}_{ijk}) \tag{4.6b}$$

where $\{\delta \mathbf{Q}_l\}_{l=1}^N = \delta Q = \in \mathbb{R}^{N\times 3}$ with $\delta \mathbf{Q}_l \in \mathbb{R}^3$ and $\mathbf{Q}_l = (Q_x^l, Q_y^l, Q_z^l) \in \mathbb{R}^3$, for all $l \in \{1, \dots .N\}$. The linearity with respect to $\delta P = \{\delta \mathbf{P}_{ijk}\}_{i,j,k=0}^{m,n,o}$ is thus made explicit.

A perturbation of the displacement of the control points $\delta d_{\text{cFFD}} = \{\delta \mathbf{d}_{ijk}\}_{i,j,k=1}^{m,n,o} \subset \mathbb{R}^3$ is found solving the least-squares problem

$$\delta d_{\text{cFFD}} = \underset{\delta d = \{\delta \mathbf{d}_{ijk}\}_{i,j,k=1}^{m,n,o} \subset \mathbb{R}^3}{\text{argmin}} \|\delta d\|_2 \quad \text{such that} \quad \mathbf{c} = \mathbf{A}_c \text{vec}(\tilde{T}_P(\mathbf{Q}, \delta P + \delta d)), \tag{4.7}$$

that can be effectively solved in closed form.

If the constraints are linear in each component (x, y, z) they can be imposed component-wise finding first the x-components $\delta d_{\text{cFFD},x}$ of the perturbations $\delta d_{\text{cFFD}} = \{\delta \mathbf{d}_{ijk}\}_{i,j,k=1}^{m,n,o} \subset \mathbb{R}^3$ and subsequently the y- and z-components, $\delta d_{\text{cFFD},y}$ and $\delta d_{\text{cFFD},z}$, respectively.

This strategy is successfully applied for constraints on the volumes of triangulations that define 3d objects in [114]. Other linear constraints that can be imposed are the position of the barycenter (linear) and the surface area (bi-linear).

Sometimes additional constraints on the position of the control points must be enforced: for example for the Duisburg test case **HB** of section 4.5.2 we deform only the region of the bulb of a whole ship's hull. This region is extracted from the hull's STL file and two straight cuts are thus introduced. To keep the displacements null close to these cuts additional constraints must be enforced on the cFFD deformations. We do so with the employment of a weight matrix $M = \{\omega_{ijk}\}_{i,j,k=1}^{m,n,o} \subset \mathbb{R} \cap \{x \geq 0\}$ that multiplies δd:

$$\delta d_{\text{cFFD}} = \underset{\delta d = \{\delta \mathbf{d}_{ijk}\}_{i,j,k=1}^{m,n,o} \subset \mathbb{R}^3}{\text{argmin}} \|M\delta d\|_2 \quad \text{such that} \quad \mathbf{c} = \mathbf{A}_c \text{vec}(\tilde{T}_P(\mathbf{Q}, \delta P + \delta d)), \tag{4.8}$$

with $M\delta d = \{\omega_{ijk} \delta \mathbf{d}_{ijk}\}_{i,j,k=1}^{m,n,o}$.

4.3 Constrained Generative Models

In the following sections, we introduce the architectures we employ for generative modelling (GM) and the novel constrained generative models (cGMs).

4.3.1 Generative modelling

Generative models (GMs) [106] have been successfully applied for computer graphics' tasks such as 3d objects generation. Their employment in surrogate modelling is especially beneficial when the computational costs needed to create a new geometry from real observations are reduced. One disadvantage is that depending on the complexity of the distribution to be approximated, their training requires a large amount of data. In this work, we focus on studying the possible employment of GMs for the reduction of the space of the parameters that specify geometrical deformations. In our case, the parameters are the displacements of cFFD. Another objective is to reduce the computational costs needed to generate new linear constrained geometries with cFFD.

We denote with $p_X(x)$ the probability density of the distribution of 3d objects and with $X : (A, \mathcal{A}, P) \to \mathbb{R}^M$ the random variable representing it, with M the number of points or degrees of freedom of the 3d mesh. The triple $(A, \mathcal{A}, P)$ corresponds to the space of events, sigma algebra and probability measure of our setting. The variable x may represent 3d point clouds, 3d meshes or general subdomains of $\mathbb{R}^3$. We are going to focus on generative models that approximate $p_X(x)$ with $p_\theta(x)$ where θ are parameters to be found and that can be factorized through latent variables z such that

$$p_\theta(x) = \int_Z p_\theta(x|z)p_\theta(z)dz \tag{4.9}$$

where z is typically low dimensional with respect to x, with dimension $R \ll M$. These models are called latent variable models[253]. Given a set of training samples of the distribution associated to X obtained with cFFD (section 4.2.2), the objective is to sample new 3d objects with quality comparable to the one generated using cFFD. It is crucial that the new samples satisfy exactly the linear or multilinear constraints imposed with cFFD.

As the subject of generative models is well-known in the literature we summarize briefly the architectures we are going to employ.

Simple Autoencoder As a toy model for pure benchmarking we first implement a very simple Autoencoder [1], composed of two parts: an Encoder $Enc_\psi : \mathbb{R}^M \to \mathbb{R}^R$ which encodes the mesh in the latent space, and a Decoder $Dec_\theta : \mathbb{R}^R \to \mathbb{R}^M$ that takes a point of the latent space and returns it to the data space. Autoencoders were originally created for learning the latent representations of the data and have many applications [1], like denoising. Some recent research has empirically shown [102, 67] that they can indeed be used as generative models. They are trained on the L^2 loss

$$L_{\theta,\psi}(x) = ||x - Dec_\theta(Enc_\psi(x))||_2, \tag{4.10}$$

the associated generative model is

$$p_\theta(x|z) = Dec_\theta(z) \tag{4.11}$$

where z is sampled from a multivariate normal distribution.

Beta Variational Autoencoder A Variational Autoencoder [148] is a probabilistic version of the autoencoder. It is based on the concept of Evidence Variational Lower Bound (ELBO) that can be employed in place of the log-likelihood $p_\theta(x)$ when it is not directly computable. Given a variational distribution $q_\psi(z|x)$ the ELBO is

$$ELBO_{\psi,\theta}(x) = E_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right)\right] = E_{q_\phi(z|x)}\left[\log\left(p_\theta(x \mid z)\right)\right] - KL\left(q_\phi(z \mid x)\|p_\theta(z)\right) \quad (4.12)$$

it follows that

$$ELBO_{\psi,\theta}(x) \leq \log p_\theta(x). \quad (4.13)$$

Variational autoencoders model $q_\psi(z|x)$ and $p_\theta(x|z)$ as two normal distributions:

$$q_\psi(z|x) = \mathcal{N}(a_\psi(x), b_\psi(x)) \quad (4.14)$$

where $a_\psi, b_\psi : \mathbb{R}^M \to \mathbb{R}^R$ are encoders and

$$p_\theta(x|z) = \mathcal{N}(Dec_\theta(z), \sigma), \quad (4.15)$$

Notice that, the KL term in the ELBO is a regularizing term on the variational distribution. Depending on the strength of $p_\theta(x|z)$ the Variational autoencoders can suffer from two problems: if it is dominant then there is no regularizing effect and so the samples from $p_\theta(z)$ will have poor quality. Otherwise, the KL will go to 0, so a_ψ, b_ψ will degenerate to constants and the model will start behaving autoregressively. To solve these two problems, an α term is added to the ELBO:

$$L_{\theta,\phi} = E_{q_\phi(z|x)}\left[\log\left(p_\theta(x \mid z)\right)\right] - \alpha KL\left(q_\phi(z \mid x)\|p_\theta(z)\right), \quad (4.16)$$

this technique is also used in [243].

Adversarial Autoencoder Another class of Generative Models are Generative Adversarial Networks which have the same structure of an autoencoder

$$p_\theta(x|z) = Dec_\theta(z) \quad (4.17)$$

but they are trained with the help of a discriminator D_λ in the following min-max game:

$$\min_\theta \max_\lambda V(\theta,\lambda) = E_{p_X(x)}\left[\log\left(D_\lambda(x)\right)\right] + E_{p_\theta(z)}\left[\log\left(1 - D_\lambda\left(Dec_\theta(z)\right)\right)\right]. \quad (4.18)$$

In this context, the decoder is also called generator. The discriminator is used to distinguish between the data and the generator samples while the generator wants to fool it. A problem of generative adversarial networks is that the min-max game can bring instability, for the details of this we refer to

[11]. A solution is to combine them with variational autoencoders to get the following loss:

$$\max_{\theta,\phi}\min_{\lambda} L_{\theta,\phi,\lambda} = E_{q_\phi(z,x)}\left[\log\left(p_\theta(x \mid z)\right)\right] - E_{p_\theta(z)}\left[\log\left(1 - D_\lambda(z)\right)\right] + E_{q_\phi(z)}\left[\log\left(D_\lambda(z)\right)\right], \quad (4.19)$$

which is the ELBO with the GAN loss instead of the KL term. This model is called Adversarial Autoencoder [175]. The associated generative model is

$$p_\theta(x|z) = \mathcal{N}(Dec_\theta(z), \sigma), \quad \text{with} \quad z \sim \mathcal{N}(0, I_R) \quad (4.20)$$

Boundary Equilibrium GAN Another solution to the GAN problems is the Boundary Equilibrium GAN [27]. In this particular model, the discriminator is an autoencoder

$$D_\lambda(x) = Dec_\lambda(Enc_\lambda(x)) \quad (4.21)$$

and the objective is to learn the autoencoder loss function

$$f_\lambda = ||x - D_\lambda(x)||_2 \quad (4.22)$$

while maintaining a balance between the data loss and the generator loss to prevent instabilities caused by the min-max training. Let G_θ the generator, this is achieved in three steps using control theory:

- Solve $\min_\lambda E_{p_X(x)}[f_\lambda(x)] - k_t E_{p_X(x)}[f_\lambda(G_\theta(Enc_\lambda(x)))]$ over λ.
- Solve $\max_\theta E_{p_Z(z)}[f_\lambda(G_\theta(z)]$.
- Update $k_t = k_{t-1} + a(\gamma E_{p_X(x)}[f_\lambda(x)] - E_{p_Z(z)}[f_\lambda(G_\theta(z)])$.

The associated generative model is

$$p_\theta(x|z) = G_\theta(z), \quad \text{with} \quad z \sim \mathcal{N}(0, I_R). \quad (4.23)$$

4.3.2 Generative modelling with multilinear constraints

In this section, we introduce our new framework to impose linear or multilinear constraints on GMs. The idea is to rely on the generalization properties of NNs. The cGMs layers' objective is to approximate the training distributions while correcting the unwanted deformations that do not satisfy exactly the geometrical constraints. Multi-linear constraints are enforced subsequently in the same way as linear ones. We will show that the volume and the barycenter position of 3d objects can be preserved in our numerical experiments section 4.5.

The best architectures that adapt to datasets structured on meshes are graph neural networks [34]. Unfortunately, compared to convolutional neural networks for data structured on Cartesian grids, they are quite heavy to train for large problems supported on computational meshes. Thus, we leave to further studies the implementation of our methodology with GNNs and we focus on generative models that employ principal component analysis (PCA) to perform dimension reduction as a preprocessing step. After having collected a training and test dataset of 3d point clouds,

$$\mathcal{X}_{\text{train}} = \begin{pmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{n_{M\times\text{train}}} \\ | & | & & | \end{pmatrix}^T \in \mathbb{R}^{n_{\text{train}}\times M}, \qquad \mathcal{X}_{\text{test}} = \begin{pmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{n_{M\times\text{test}}} \\ | & | & & | \end{pmatrix}^T \in \mathbb{R}^{n_{\text{test}}\times M}, \tag{4.24}$$

PCA is applied to obtain a set of r modes of variation $U_{\text{PCA}} \in \mathbb{R}^{M\times r}$ with $M \gg r > 0$, such that fixed a tolerance $1 \gg \varepsilon > 0$ for the reconstruction error in Frobenious norm $\|\cdot\|_F$, we have:

$$\|(I_M - U_{\text{PCA}}U_{\text{PCA}}^T)\mathcal{X}_{\text{train}}\|_F \leq \varepsilon. \tag{4.25}$$

The notation M for the dimension of a single point cloud includes the 3 (x,y,z) components of each point stacked on each other on the same vector. Without considering the enforcement of the geometrical constraints, the GMs could now be trained to approximate the lower dimensional distributions $U_{\text{PCA}}^T\mathcal{X}_{\text{train}} \in \mathbb{R}^{r\times n_{\text{train}}}$ with a great saving in terms of computational cost since $M \gg r$. However, this reasoning may not work with more complex distributions that cannot be accurately approximated as combinations of PCA modes. A direct consequence would be that the generalization error on the test dataset would perform significantly worse than the training error. Fortunately, this is not the case of our numerical studies: $r = 140$ and $r = 30$ PCA modes are sufficient for the Stanford's bunny and DTCHull test cases, respectively.

We denote with $Y = U_{\text{PCA}}X$ the random variable associated with the dataset at hand projected on the PCA's modes. With $\mathbf{y} \in \mathbb{R}^r$ we represent its realizations. To reconstruct the point clouds on the full space, a matrix multiplication $U_{\text{PCA}}\mathbf{y} = \tilde{\mathbf{x}} \in \mathbb{R}^M$ is needed. We denote with $\tilde{X} : (A, \mathcal{A}, P) \to \mathbb{R}^M$ the random variable associated with the GMs, that is used to approximate X without the enforcement of geometrical constraints.

Our methodology to impose linear or multilinear constraints on GMs, affect both the training and predictive stages. It consists in the addition of a final constraint enforcing layer $l_{\text{enforcing}} : \mathbb{R}^M \to \mathbb{R}^M$ that acts on the reconstructed outputs $\tilde{\mathbf{x}} \in \mathbb{R}^M$. The final layer $l_{\text{enforcing}}$ perturbs the outputs $\tilde{\mathbf{x}} + \delta\tilde{\mathbf{x}} = \tilde{\tilde{\mathbf{x}}} \in \mathbb{R}^M$ such that the geometrical constraints with fixed values represented by the vector $\mathbf{c} \in \mathbb{R}^{n_c}$ are exactly satisfied through the solution of the least-squares problem

$$\delta\tilde{\mathbf{x}} = \underset{\delta\mathbf{x}\in\mathbb{R}^M}{\text{argmin}}\|\delta\mathbf{x}\|_2, \qquad \text{s.t.} \qquad A_c(\tilde{\mathbf{x}} + \delta\mathbf{x}) = \mathbf{c}, \tag{4.26}$$

where $A_c \in \mathbb{R}^{n_c\times M}$ is the matrix representing the linear geometrical constraints and $\delta\tilde{\mathbf{x}} \in \mathbb{R}^M$ are the point clouds perturbations. Multi-linear constraints, like the volume, are imposed subsequently component after component. The final outputs $\tilde{\tilde{\mathbf{x}}} \in \mathbb{R}^M$ that satisfy exactly the geometrical constraints are then forwarded to the loss function during the training or used directly in the predictive phase.

It is crucial to remark that the linear constraints are imposed exactly on the outputs of the GMs, after solving a least-squares problem similar to equation 4.7. The main difference is that the perturbations affect directly the cloud of points coordinates and not the displacements of the control points as in the constrained FFD presented in section 4.2.2. Figure 4.1 shows a sketch of the methodology, including the application of PCA and of the constraints enforcing layer, for the simple autoencoder GM. The procedure is easily adapted for the other GMs of the previous section 4.3.1.

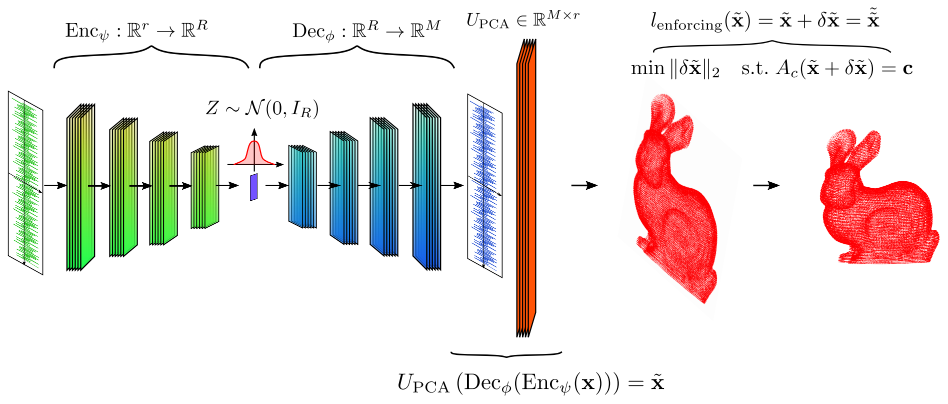

$\mathrm{Enc}_\psi : \mathbb{R}^r \to \mathbb{R}^R$
$\mathrm{Dec}_\phi : \mathbb{R}^R \to \mathbb{R}^M$
$U_{\mathrm{PCA}} \in \mathbb{R}^{M\times r}$
$l_{\mathrm{enforcing}}(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}} + \delta\tilde{\mathbf{x}} = \tilde{\tilde{\mathbf{x}}}$
$\min \|\delta\tilde{\mathbf{x}}\|_2 \quad \text{s.t. } A_c(\tilde{\mathbf{x}} + \delta\tilde{\mathbf{x}}) = \mathbf{c}$
$Z \sim \mathcal{N}(0, I_R)$
$U_{\mathrm{PCA}}\,(\mathrm{Dec}_\phi(\mathrm{Enc}_\psi(\mathbf{x}))) = \tilde{\mathbf{x}}$

Fig. 4.1 Training of the constrained autoencoder generative model composed by a parametrized encoder Enc_ψ, a decoder Dec_ϕ, a reconstruction layer that employs PCA modes $U_{\mathrm{PCA}} \in \mathbb{R}^{M\times r}$ and a final multilinear constraint enforcing layer $l_{\mathrm{enforcing}}$. The generalization to other generative models of is straight-forward.

Despite this quite arbitrary correction $\tilde{\mathbf{x}} + \boldsymbol{\delta}\tilde{\mathbf{x}} = \tilde{\tilde{\mathbf{x}}} \in \mathbb{R}^M$ during the training of the GMs, the NNs' layers manage to correct the outputs $y \in \mathbb{R}^r$ to balance the PCA reconstruction and the perturbation $\boldsymbol{\delta}\tilde{\mathbf{x}} \in \mathbb{R}^M$. The simplicity of this strategy together with the generalization capabilities of GMs make our methodology effective and relatively easy to implement. We remark that the methodology is not strictly linked with FFD and cFFD: arbitrary techniques, like [262, 83], can be employed to generate the training and test datasets as long as the geometrical constraints are linear or multilinear.

Our method cannot be applied to nonlinear constraints. A possible way to implement them is to use substitute the linear constraints enforcing layer with nonlinear optimization layers [5]. The disadvantage is the higher computational cost and the fact that the nonlinear constraints would not be satisfied exactly.

4.4 Surrogate modelling

In our numerical experiments in section 4.5, we validate the distributions of the trained cGMs with some geometrical and physical metrics of interest. To evaluate these metrics we need to compute some physical fields with numerical simulations on computational domains affected by the newly sampled 3d objects from the cGMs. A Poisson problem is considered in section 4.5.1 and the Navier-Stokes equations in section 4.5.2.

The first test case in section 4.5.1 considers as domain the deformed Stanford bunny [257] with fixed barycenter location. The second test case in section 4.5.2 considers as domain a parallelepiped with two separated phases (water and air) in which is embedded a ship hull whose bulb is deformed with cGMs while keeping its volume fixed.

We show how reduced order modelling [126, 228] can benefit from the employment of cGMs. The main advantage is dimension reduction of the parameter space. In fact, the space of parameters that define the geometrical deformations changes from a possibly high-dimensional space associated with the cFFD method to the usually smaller latent space of the cGMs. The consequence of this dimension reduction is the increased efficiency of non-intrusive ROMs based on interpolation methods from the space of parameters to the coefficients of proper orthogonal decomposition (POD) modes.

The outputs of cGMs are 3d point clouds organized into STL files. To obtain a mesh from each STL we use interpolation with radial basis functions (RBF) [70] of a reference computational mesh: the 3d point clouds generated by the cGM are used to define an interpolation map that will deform the reference computational mesh. We proceed in this way because it is simpler to design ROMs on computational meshes with the same number of degrees of freedom. In fact, with RBF interpolation, each mesh generated from a STL file has the same number of cells of the reference mesh. A possible solution is to evaluate projection and extrapolation maps from a fine mesh common to all the others associated with different sampled 3d objects with possibly different numbers of dofs. Since MOR is not our main focus, we employ RBF interpolation, knowing that in this way, the geometrical constraints are not imposed exactly anymore but subject to the level of discretization of the mesh and to the accuracy of the RBF interpolation.

In this section, we briefly summarize the employment of RBF interpolation and present the non-intrusive ROMs we use in the results' section 4.5: proper orthogonal decomposition with interpolation (PODI) performed by Gaussian process regression (GPR) [214], radial basis functions (RBF) or feed-forward neural networks (NNs) [106]. The active subspaces method (AS) [60] is also introduced as reference dimension reduction method for the space of parameters and coupled with PODI model order reduction [248].

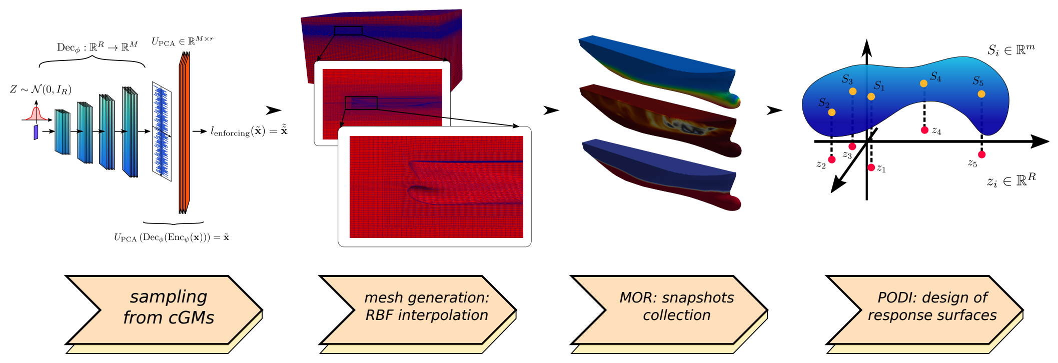

$\mathrm{Dec}_\phi : \mathbb{R}^R \to \mathbb{R}^M$
$U_{\mathrm{PCA}} \in \mathbb{R}^{M\times r}$
$Z \sim \mathcal{N}(0, I_R)$
$l_{\mathrm{enforcing}}(\tilde{\mathbf{x}}) = \tilde{\tilde{\mathbf{x}}}$
$U_{\mathrm{PCA}}\left(\mathrm{Dec}_\phi(\mathrm{Enc}_\psi(\mathbf{x}))\right) = \tilde{\mathbf{x}}$
$S_i \in \mathbb{R}^m$
S_1 S_2 S_3 S_4 S_5
z_1 z_2 z_3 z_4 z_5
$z_i \in \mathbb{R}^R$
sampling from cGMs
mesh generation: RBF interpolation
MOR: snapshots collection
PODI: design of response surfaces

Fig. 4.2 Pipeline for non-intrusive reduced order modelling: from 3d object generation with constrained generative models (cGMs) to mesh deformation with radial basis functions (RBF) to surrogate modelling with proper orthogonal decomposition with interpolation (PODI).

4.4.1 Mesh interpolation with radial basis functions

After having obtained a new 3d object as STL file from the trained cGMs represented by the 3d points cloud $x^1 = \{\mathbf{x}^1_i\}_{i=1}^{\tilde{M}} \in \mathbb{R}^3$, we define a RBF interpolation map $s : \mathbb{R}^3 \to \mathbb{R}^3$,

$$s(\mathbf{x}) = \mathbf{q}(\mathbf{x}) + \sum_{i=1}^{N_b} \beta_i \xi(\|\mathbf{x} - \mathbf{x}_{b_i}\|), \qquad \forall \mathbf{x} \in \mathbb{R}^3, \tag{4.27}$$

where $\{\mathbf{x}_{b_i}\}_{i=1}^{N_b} \subset \mathbb{R}^3$ are the RBF control points, N_b is the number of RBF control points, $\xi : \mathbb{R}^3 \to \mathbb{R}^3$ is a radial basis function and $\mathbf{q} \in [\mathbb{P}^1(\mathbb{R}^3)]^3$ a vector-valued polynomial of degree 1 with coefficients $\delta = \{\delta^1_0, \delta^1_1, \delta^2_0, \delta^2_1, \delta^3_0, \delta^3_1\}$ to be defined. The coefficients of the RBF interpolation map s are $\beta = \{\beta_i\}_{i=1}^{N_b} \subset \mathbb{R}^3$. The parameters of the RBF interpolation map to be defined are β and δ. Given a reference mesh, the STL file from which it was obtained is employed as reference STL, with associated 3d point cloud $x^2 = \{\mathbf{x}^2_i\}_{i=1}^{\tilde{M}} \in \mathbb{R}^3$. Since the newly sampled STL files and the reference one have the same number of points $\tilde{M}$, the following interpolation problem gives β and δ:

$$s(\mathbf{x}^2_j) = q(\mathbf{x}^1_j) + \sum_{i=1}^{N_b} \beta_i \xi(\|\mathbf{x}^1_j - \mathbf{x}_{b_i}\|), \qquad \forall j \in \{1, \dots, \tilde{M}\} \tag{4.28a}$$

$$0 = \sum_{i=1}^{N_b} \beta_i q(\mathbf{x}_{b_i}). \tag{4.28b}$$

In practice, to each point cloud x^1 and x^2 is usually added a set of points to keep fixed such that the reference mesh is only deformed in a limited region of the computational domain. This is the reason why we employed the notation $\tilde{M} > M$ to differentiate between the dimension M of the 3d point clouds that are the output of the cGMs and the dimension $\tilde{M}$ of the RBF control points that are enriched with additional points to be kept fixed. An example of reference (in red) and deformed

$M = 5000$ (in blue) control points from the constrained autoencoder generative model is shown in Figure 4.3.

We remark that due to the RBF interpolation, the multilinear constraints are not imposed exactly anymore, but only approximately. To avoid this problem, the same discretization of the 3d object embedded on the computational mesh must be used converted as point cloud also for the training of the cGMs. This is effectively done for the Stanford's bunny test case, but not for the DTCHull's one. In fact, we decided to reduce the resolution of the hull for the numerical simulations in order to lower the computational costs for these preliminary studies.

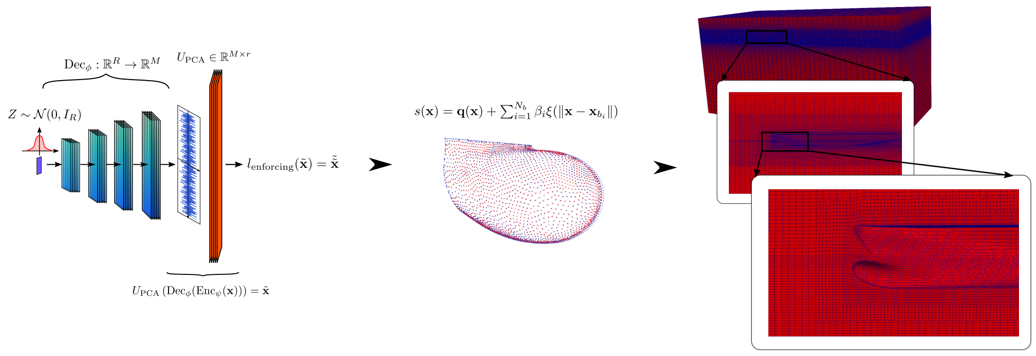

$\mathrm{Dec}_\phi : \mathbb{R}^R \to \mathbb{R}^M$
$U_{\mathrm{PCA}} \in \mathbb{R}^{M\times r}$
$Z \sim \mathcal{N}(0, I_R)$
$l_{\mathrm{enforcing}}(\tilde{\mathbf{x}}) = \tilde{\tilde{\mathbf{x}}}$
$U_{\mathrm{PCA}}\left(\mathrm{Dec}_\phi(\mathrm{Enc}_\psi(\mathbf{x}))\right) = \tilde{\mathbf{x}}$
$s(\mathbf{x}) = \mathbf{q}(\mathbf{x}) + \sum_{i=1}^{N_b} \beta_i \xi(\|\mathbf{x} - \mathbf{x}_{b_i}\|)$

Fig. 4.3 Pipeline for the deformation of the reference computational mesh for the Duisburg test case **HB**. The numerical results are reported in section 4.5.2. In red the 3d point cloud of the reference STL, in blue the 3d point cloud of a deformed bulb generated by a constrained autoencoder used as cGM on the left. From these two points cloud x^2 and x^1 a RBF interpolation map is defined from equation (4.28) and used to deform the reference mesh on the right. The bulb location with respect to the whole computational domain is shown.

4.4.2 Proper orthogonal decomposition with interpolation

After having obtained a deformed mesh with RBF interpolation, as for the DTCHull test case, or directly without RBF interpolation, as for the Stanford's bunny test case, numerical simulations can be performed. For each one of the newly sampled geometries from the cGMs, some physical fields of interest are collected and organized in a snapshots matrix $S \in \mathbb{R}^{m\times n_{\mathrm{ROM,train}}}$, where m is the number of degrees of freedom while the rest $n_{\mathrm{ROM,test}} = n_{test} - n_{\mathrm{ROM,train}}$ is used for testing the accuracy of the reduced order models (ROMs). We remind that n_{test} is the number of newly sampled geometries from the cGMs, divided in $n_{\mathrm{ROM,train}}$ training and $n_{\mathrm{ROM,test}}$ test datasets to validate our model order reduction procedure.

We employ the finite volumes method (FVM) [260] implemented in the open-source software library `OpenFoam` [264], for both test cases in section 4.5. For the Stanford's bunny we set a simple Poisson problem, while for the DTCHull we start from the DTCHull multiphase tutorial [268] for the Reynolds Averaged Navier-Stokes equations (RANS) with associated solver *interFoam*.

Proper orthogonal decomposition (POD), already introduced as PCA previously, is employed to compute a set of reduced basis $U_{\mathrm{ROM}} \in \mathbb{R}^{m\times r_{\mathrm{ROM}}}$. For our studies, we will employ only non-intrusive ROMs: new values of the physical fields are obtained with interpolations or regressions of the expansion with respect to the POD modes U_{ROM}, this technique is generally called POD with interpolation (PODI). The input-output dataset of the interpolation or regression are the latent coordinates of the cGMs $\{z_i\}_{i=1}^{n_{\text{ROM,train}}} \in \mathbb{R}^R$ or the displacements δP of cFFD for the inputs and the coefficients $U_{\text{ROM}}^T S \in \mathbb{R}^{r_{\text{ROM}} \times n_{\text{ROM,train}}}$ for the outputs:

$$\{(z_l, S_l)\}_{p=1}^{n_{\text{ROM,train}}} \subset \mathbb{R}^R \times \mathbb{R}^m, \qquad \text{(inputs-outputs of PODI for the cGMs)} \tag{4.29a}$$

$$\{(\delta P^l, S_l)\}_{p=1}^{n_{\text{ROM,train}}} \subset \mathbb{R}^p \times \mathbb{R}^m, \qquad \text{(inputs-outputs of PODI for the cFFD)} \tag{4.29b}$$

where p is the number of cFFD displacements $\{\delta P^l\}_{l=1}^p = \{\{\delta \mathbf{P}^l_{i,j,k}\}_{i,j,k=0}^{\tilde{m},\tilde{n},\tilde{o}}\}_{l=1}^p$ different from 0, $\tilde{m},\tilde{n},\tilde{o}$ count the non-zero displacements, possibly less than the m,n,o control points of the lattice of FFD. The notation $S_j \in \mathbb{R}^m$ refers to the rows of the snapshots matrix $S \in \mathbb{R}^{m \times n_{\text{ROM,train}}}$.

Once the interpolation or regression maps are defined from the training input-output datasets, the new $n_{\text{ROM,test}}$ physical fields associated with the geometries of the test dataset, are efficiently evaluated through these interpolations or regressions, without the need for full-order numerical simulations.

The method we employ to perform the interpolation is RBF interpolation, while Gaussian process regression (GPR) and feed-forward neural networks (NNs) are employed to design a regression of the POD coefficients. These techniques are compared in section 4.5. Also, the Active subspaces method (AS) will be used to build response surfaces from the latent coordinates for the cGMs or the cFFD displacements while performing also a further reduction in the space of parameters.

4.4.3 Active subspaces method

We briefly sketch the Active Subspaces method (AS). It will be used as reference dimension reduction method in the space of parameters and as regression method to perform model order reduction through the design of response surfaces. We will define response surfaces for the ROMs built on top of both cFFD and cGMs deformed meshes: we will see that not only the initial parameter space dimension p associated with the cFFD's non-zero displacements (see equation (4.29)) can be substantially reduced for our test cases, but also the latent spaces' dimensions of the cGMs.

Given a function $f : \chi \subset \mathbb{R}^R \to \mathbb{R}$ from the space of parameters to a scalar output of interest $f : \boldsymbol{\mu} \mapsto f(\boldsymbol{\mu})$, the active subspaces are the leading eigenspaces of the uncentered covariance matrix of the gradients:

$$\Sigma = E[\nabla_{\boldsymbol{\mu}}(f \nabla_{\boldsymbol{\mu}} f)^T] = \int_{\chi} (\nabla_{\boldsymbol{\mu}} f)(\nabla_{\boldsymbol{\mu}} f)^T \rho \, d\boldsymbol{\mu}, \tag{4.30}$$

where $\rho : \chi \subset \mathbb{R}^R \to \mathbb{R}$ is the probability density function of the distribution of the inputs $\boldsymbol{\mu}$ considered as a vector-valued random variable in the probability space $(\chi, \mathcal{A}, P)$. The gradients of f are usually approximated with regression methods if they cannot be evaluated directly. The uncentered covariance matrix $\Sigma \in \mathbb{R}^{R \times R}$ is computed approximately with the simple Monte Carlo method and then the eigenvalue decomposition

$$\Sigma = W\Lambda W^T, \qquad W = [W_1, W_2], \qquad \Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_R), \qquad W_1 \in \mathbb{R}^{R\times r_{\mathrm{AS}}},\ W_2 \in \mathbb{R}^{R\times(R-r_{\mathrm{AS}})}, \tag{4.31}$$

highlights the active $W_1 \in \mathbb{R}^{R\times r_{\mathrm{AS}}}$ and the inactive $W_2 \in \mathbb{R}^{R\times(R-r_{\mathrm{AS}})}$ subspaces, corresponding to the first r_{AS} eigenvalues $\{\lambda_1, \dots, \lambda_{r_{\mathrm{AS}}}\}$ and last $R - r_{\mathrm{AS}}$ eigenvalues $\{\lambda_{r_{\mathrm{AS}}}, \dots, \lambda_R\}$.

A response surface can be obtained with the approximation

$$f(\boldsymbol{\mu}) \approx g(W_1^T\boldsymbol{\mu}) = g(\boldsymbol{\mu}_1), \tag{4.32}$$

where $g : W_1^T(\chi) \subset \mathbb{R}^{r_{\mathrm{AS}}} \to \mathbb{R}$ is a surrogate model for f from the reduced space of active variables $\boldsymbol{\mu}_1 = W_1^T\boldsymbol{\mu}$, instead of the full parameter space. We will employ Gaussian process regression to evaluate g.

4.5 Numerical results

We present two numerical studies to validate our novel methodology for constrained generative modelling previously introduced in section 4.3. The test cases we consider are the Stanford Bunny [257] (**SB**) and the bulb of the hull of the Duisburg test case [268] (**HB**). After generating new samples from the constrained generative models (cGMs), the crucial task of validating the results must be carried out. In fact, a natural metric that evaluates the quality of the generated distribution of the cGMs is not available: for each problem at hand we have to decide which criteria, summary statistics, geometrical and physical properties are most useful to validate the generated distribution. For this reason, we decide that the **SB** test case's generated 3d objects will be employed to solve a Poisson problem with fixed barycenter position and the **HB** test cases' generated bulbs will be embedded on a larger computational domain to solve the multiphase Navier-Stokes equations with fixed volume of the hull.

The computational meshes employed and the STL files used for the 3d object generation are shown in Figure 4.4. The STL files are considered as point clouds and will be the inputs and outputs of the cGMs: the number of points is constant for each training and generated geometry. The STL files of test cases **SB** and **HB**, have **145821** and **5000** points, respectively. We remark that only the deformations of the bulb are generated by our cGMs, that is only the **5000** points of the bulb over the **33866** points representing the whole hull are deformed. The computational meshes have sizes **114354** and **1335256** for the **SB** and **HB** test cases, respectively.

We check the Jensen-Shannon Distance (JSD) [182] between the two quantity of interests X and Y

$$JSD(X,Y) = \sqrt{\frac{1}{2}KL(p_X||0.5\cdot p_X + 0.5\cdot p_Y) + 0.5\cdot KL(p_Y||0.5\cdot p_X + 0.5\cdot p_Y)} \tag{4.33}$$

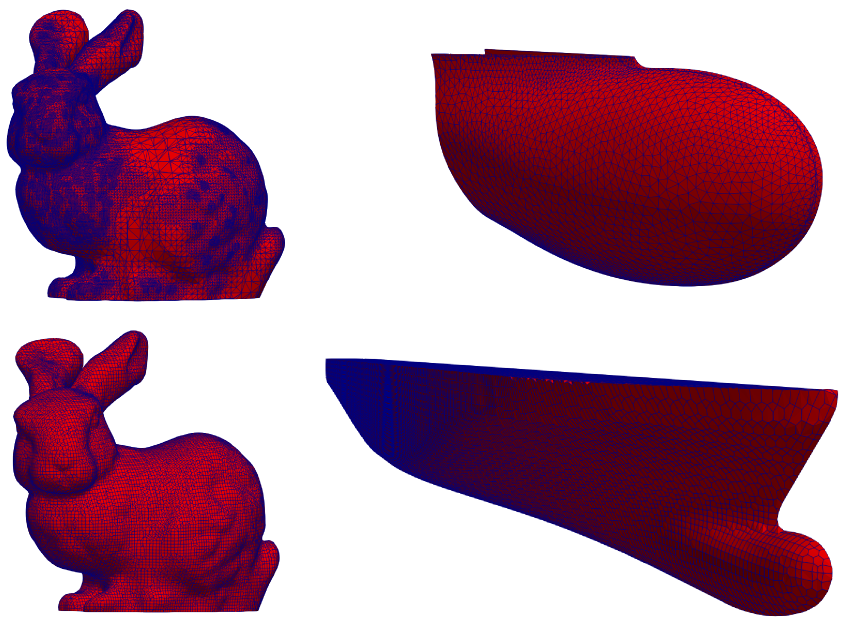


Fig. 4.4 **Top left:** STL of SB with **145821** points. **Top right:** STL of HB with **5000** points. **Bottom left:** computational mesh of SB with **M=114354** cells and dofs. **Bottom right:** computational mesh of HB with **M=1335256** cells: only the boundary of the computational domain that intersects the hull is shown.

where p_X and p_Y are the p.d.f. of X and Y respective (and are estimated from the samples using kernel density estimation) and

$$KL(p||q) = \int_{\mathbb{R}^n} p(x) log\left(\frac{p(x)}{q(x)}\right) dx \tag{4.34}$$

We choose JSD because: it is a distance on the probability space, it is bounded between 0 and 1 and it is invariant under affine transformation. These properties permit us to compare the model performances on different quantities.

For every architecture of each test case, we will also evaluate the sum of the variance (Var) of each point of the generated 3d point clouds. It will be a useful metric to determine which cGM produces the richest distribution in terms of variability of the sampled 3d point clouds.

In section 4.5.3 are reported the architectures' details and training specifics.

4.5.1 Stanford Bunny (SB)

The 3d object we employ is the Stanford bunny [257]. Our objective is to preserve the barycenter $\mathbf{x}_B \in \Omega_{\text{bunny}} \subset \mathbb{R}^3$, so we have the following set of constraints:

$$c_x = \frac{1}{n}\sum_{i=1}^{M} x_i, \qquad c_y = \frac{1}{n}\sum_{i=1}^{M} y_i, \qquad c_z = \frac{1}{n}\sum_{i=1}^{M} z_i. \tag{4.35}$$

The numerical model we are going to use to validate the results of cGMs is a mixed Poisson problem:

$$\Delta u(\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega_{\text{bunny}}, \tag{4.36a}$$

$$u(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega_{\text{bunny}} \cap \{y=0\} = \Gamma_D, \tag{4.36b}$$

$$\mathbf{n}\cdot\nabla u(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega_{\text{bunny}} \cap \{y=0\}^c = \Gamma_N, \tag{4.36c}$$

where the source term $f : \mathbb{R}^3 \to \mathbb{R}$ is :

$$f(\mathbf{x}) = \begin{cases} f(\mathbf{x}) = e^{\frac{1}{100-\|\mathbf{x}-\mathbf{x_B}\|_2^2}}, & \|\mathbf{x}-\mathbf{x_B}\|_2 < 10 \\ 0, & \text{otherwise} \end{cases}. \tag{4.36d}$$

The geometrical properties we are going to compare are the moments of inertia with uniform density equal to 1 (I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}) and the integral of the heat on the boundary with homogeneous Neumann conditions for the **SB** test case:

$$I_{xx} = \int_{\Omega_{\text{bunny}}} r_X^2(\mathbf{x})d\mathbf{x},\ I_{yy} = \int_{\Omega_{\text{bunny}}} r_Y^2(\mathbf{x})d\mathbf{x},\ I_{zz} = \int_{\Omega_{\text{bunny}}} r_Z^2(\mathbf{x})d\mathbf{x}, \qquad \text{(principal moments of inertia)} \tag{4.37a}$$

$$I_{xy} = \int_{\Omega_{\text{bunny}}} r_X(\mathbf{x})r_Y(\mathbf{x})d\mathbf{x},\ I_{xz} = \int_{\Omega_{\text{bunny}}} r_X(\mathbf{x})r_Z(\mathbf{x})d\mathbf{x},\ I_{yz} = \int_{\Omega_{\text{bunny}}} r_Y(\mathbf{x})r_Z(\mathbf{x})d\mathbf{x}, \qquad \text{(moments of inertia)} \tag{4.37b}$$

$$I_u = \int_{\partial\Gamma_N} u(\mathbf{x})\, d\mathbf{x} \qquad \text{(integral over the boundary } \partial\Gamma_N\text{)} \tag{4.37c}$$

where $r_X, r_Y, r_Z : \Omega_{\text{bunny}} \to \mathbb{R}_+$ are the distances from the x-, y- and z-axes, respectively. These quantities are evaluated on the discrete STL point cloud for the moments of inertia $I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}$ and on the computational mesh for the integral of the solution on the Neumann boundary I_u.

The number of training and test samples are $\mathbf{n}_{\text{train}} = \mathbf{400}$ and $\mathbf{n}_{\text{test}} = \mathbf{200}$, respectively. For model order reduction, the number of training and test components are instead $\mathbf{n}_{\text{ROM,train}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$, respectively. We use $\mathbf{r}_{\text{PCA}} = \mathbf{30}$ PCA modes for preprocessing inside the cGMs and $\mathbf{r}_{\text{POD}} = \mathbf{3}$ modes to perform model order reduction. The parameters' space dimension of cFFD is $\mathbf{p} = \mathbf{54}$, while the latent space dimensions of the cGMs is $\mathbf{R} = \mathbf{15}$. Some deformations of the cGMs introduced are shown in Figure 4.5: the field shown is the solution to the mixed Poisson problem on different deformed geometries.

The results with respect to the geometrical and physical metrics defined previously are shown in Table 4.1. Qualitatively the histograms of each architecture are reported in Figure 4.6 for the I_{zz} moment of inertia and in Figure 4.7 for the I_u integral of the solution of the mixed Poisson problem on the Neumann boundary Γ_N.

Non-intrusive model order reduction for the solutions of the mixed Poisson problem with GPR-PODI, RBF-PODI, NN-PODI and AS response surface design with $\mathbf{r}_{\text{AS}} = \mathbf{1}$ for the cFFD data is shown in Figure 4.8. Accurate surrogates models are built even if the dimension of the space of parameters changes from $\mathbf{p} = \mathbf{54}$ for the cFFD geometries to $\mathbf{R} = \mathbf{15}$ for the cGMs newly generated geometries. We also apply an additional level of reduction in the space of parameters with response surface design with AS in Figure 4.9. The inputs in this case are the one-dimensional active variables $\mathbf{r}_{\text{AS}} = \mathbf{1}$ also for the cGMs: the parameters space's dimension changes from $\mathbf{R} = \mathbf{15}$ to $\mathbf{r}_{\text{AS}} = \mathbf{1}$. For simplicity, we show the AS response surface with dimension $\mathbf{r}_{\text{AS}} = \mathbf{1}$, even if from the plot in Figure 4.22 of the first 20 eigenvalues of the uncentered covariance matrix from equation (4.30), the spectral gap [60] suggests $\mathbf{r}_{\text{AS}} = \mathbf{9}$.

The speedup for the generation of the geometries employing the cGMs instead of cFFD is around 60. The speedup of PODI non-intrusive model order reduction with respect to the full-order simulations is around 126000.

Table 4.1 **SB.** In this table we show the evaluation metrics of the Stanford bunny defined in equations (4.37): $I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}$ are the components of the inertia tensor, I_u is the integral of the solution of the mixed Poisson problem on the Neumann boundary Γ_N. The distribution of the components of the inertia tensor is obtained from $\mathbf{n}_{\text{test}} = \mathbf{200}$ test samples, while the physical metric I_u is based on $n_{\text{ROM,test}} = 20$ test samples. The BEGAN is overall the best model we managed to train for this test case. The model with the highest output variance is the BEGAN, even though it does not reach the total variance of the training dataset from cFFD that is **98**.

	AE	AAE	VAE	BEGAN
$JSD(I_{xx})$	8.6e-03	7.5e-03	2.1e-02	**6.3e-03**
$JSD(I_{xy})$	1.6e-02	**1.2e-02**	2.4e-02	**1.2e-02**
$JSD(I_{xz})$	**2.1e-02**	**2.1e-02**	2.6e-02	**2.1e-02**
$JSD(I_{yy})$	2.0e-02	1.2e-02	2.7e-02	**8e-03**
$JSD(I_{yz})$	1.9e-02	**1.5e-02**	2.3e-02	1.6e-02
$JSD(I_{zz})$	3.1e-02	2.2e-02	2.9e-02	**1.8e-02**
$JSD(I_u)$	4.0e-01	3.5e-01	**2.5e-01**	2.6e-01
Var	62	74	58	**76**

4.5.2 DTCHull bulb (HB)

The Duisburg test case [268] models a two-phase water-air turbulent incompressible flow over a naval hull. We start from `OpenFoam`'s [264] tutorial *DTCHull* related to the *interFoam* multiphase solver for the Reynolds Averaged Navier-Stokes equations (RANS) using the volume of fluid modelling.

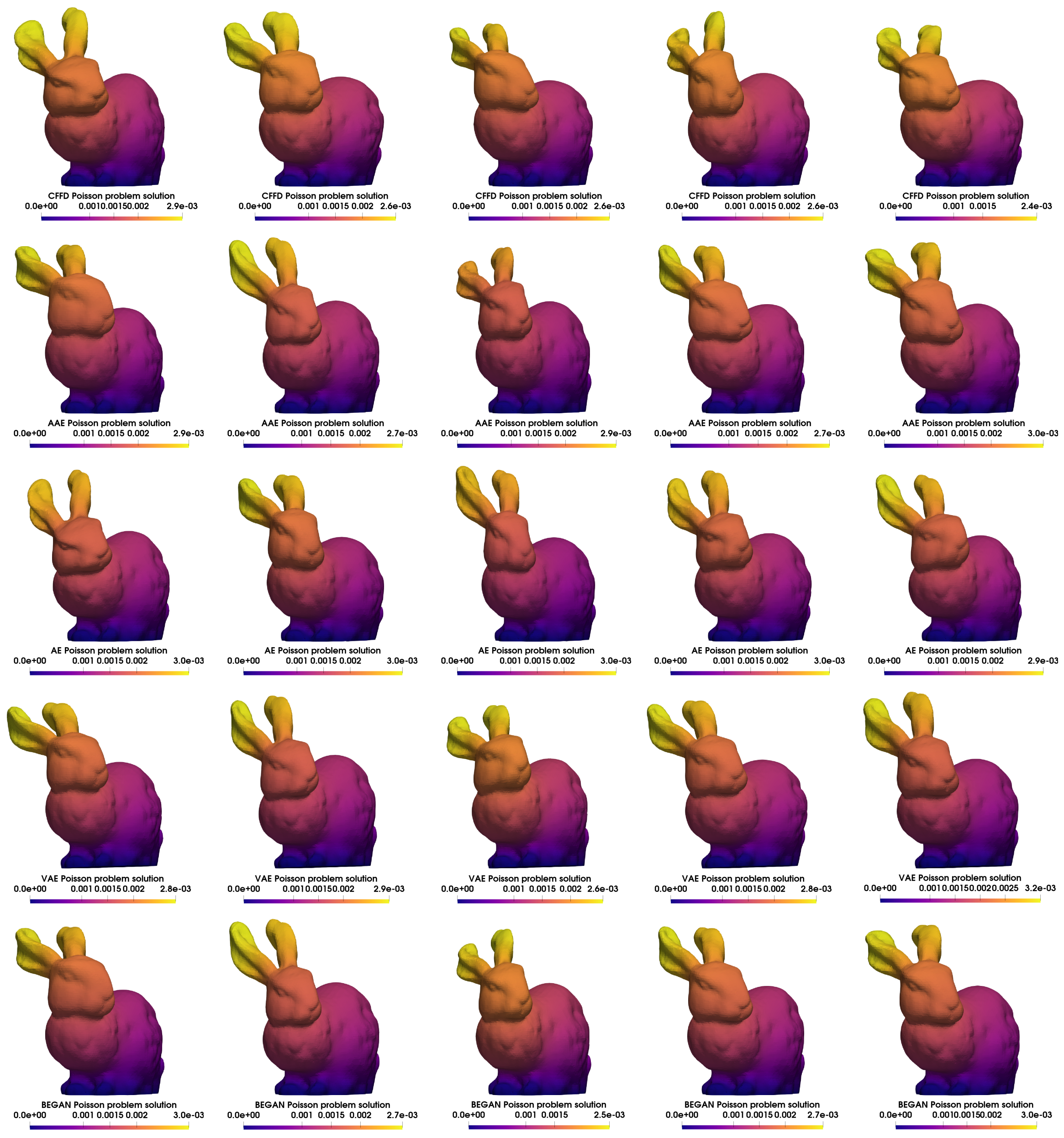

CFFD Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.9e-03
CFFD Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.6e-03
CFFD Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.6e-03
CFFD Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.6e-03
CFFD Poisson problem solution
0.0e+00 0.001 0.0015 2.4e-03
AAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.9e-03
AAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.7e-03
AAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.9e-03
AAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.7e-03
AAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
AE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
AE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
AE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
AE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
AE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.9e-03
VAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.8e-03
VAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.9e-03
VAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.6e-03
VAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.8e-03
VAE Poisson problem solution
0.0e+00 0.001 0.0015 0.002 0.0025 3.2e-03
BEGAN Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03
BEGAN Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.7e-03
BEGAN Poisson problem solution
0.0e+00 0.001 0.0015 2.5e-03
BEGAN Poisson problem solution
0.0e+00 0.001 0.0015 0.002 2.7e-03
BEGAN Poisson problem solution
0.0e+00 0.001 0.0015 0.002 3.0e-03

Fig. 4.5 **SB.** Some geometrical deformations of the Stanford bunny are shown: the first row refers to constrained cFFD of section 4.2.2, the second to the adversarial autoencoder of paragraph 4.3.1, the third to the simple autoencoder of paragraph 4.3.1, the fourth to the beta variational autoencoder of paragraph 4.3.1 and the last to the boundary equilibrium generative adversarial networks of paragraph 4.3.1. All the generative models implement the linear constraints enforcing layer of section 4.3 to preserve the position of the barycenter.

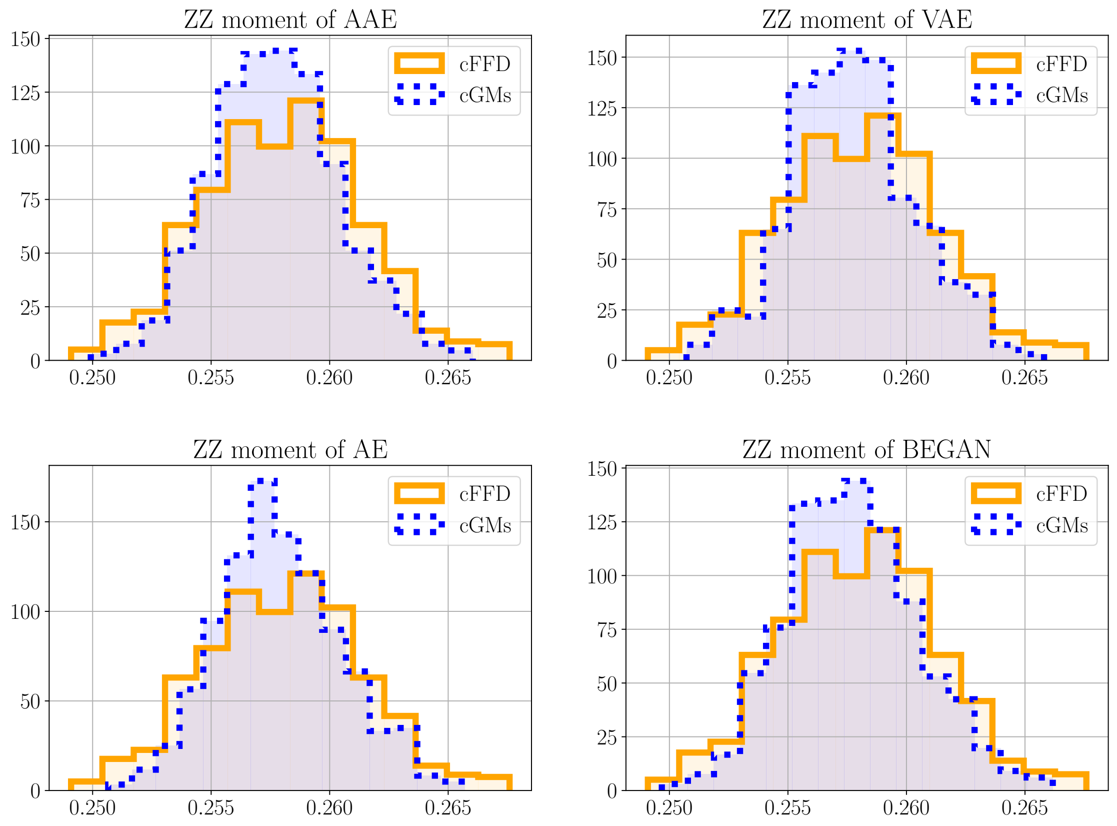


Fig. 4.6 **SB.** In this figure the histograms of the ZZ component of the inertia tensor of the cFFD and the cGMs are shown. The histogram area intersection is qualitatively more than 50%. For a quantitative measure see Table 4.1. The histograms are obtained from the $\mathbf{n}_{\text{test}} = \mathbf{200}$ test samples.

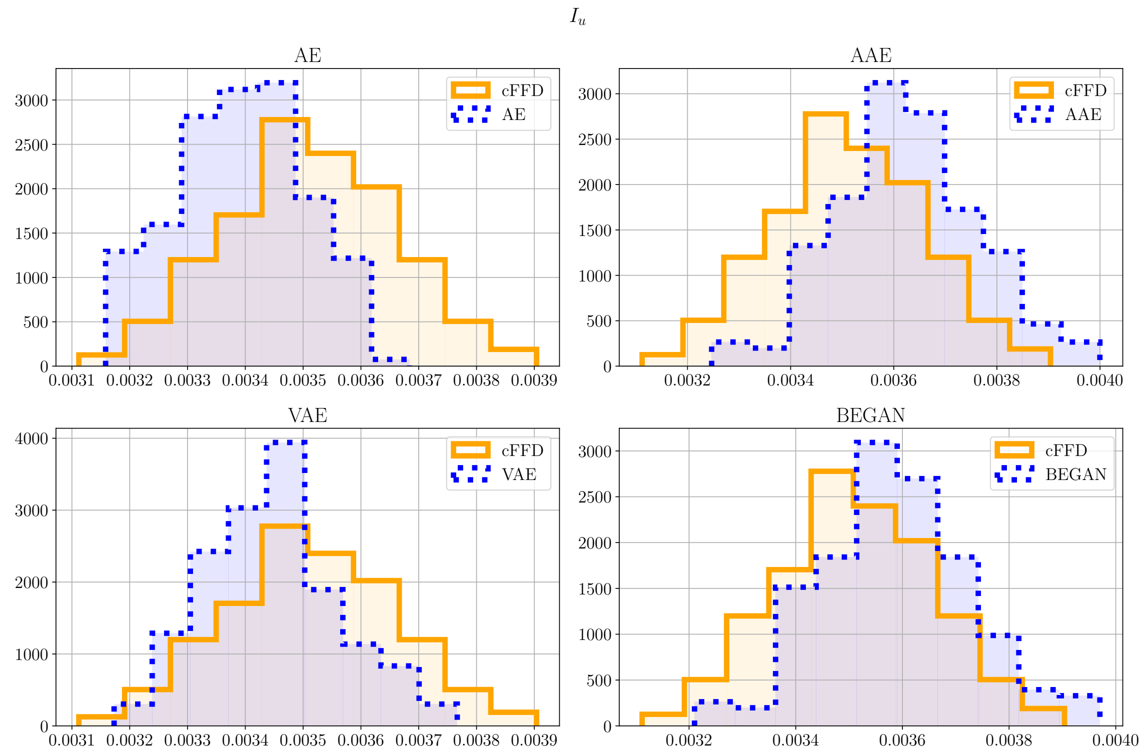


Fig. 4.7 **SB.** In this figure the histograms of the integral of the solution on the Neumann boundary Γ_N of the cFFD and the cGMs are shown. The histogram area intersection is more than 50%. For a quantitative measure see Table 4.1. The histograms are obtained from the $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ test samples.

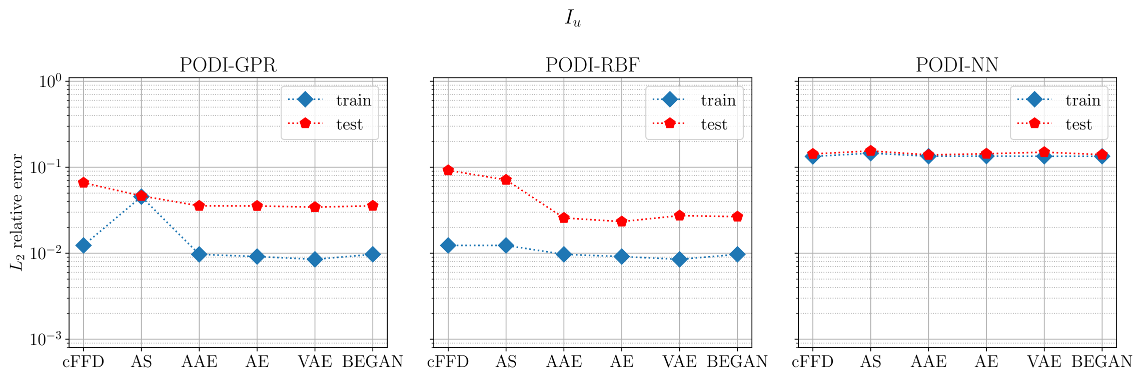


Fig. 4.8 **SB.** Here we show the ROM performance over the training and test datasets, using different interpolation and regression techniques combined with proper orthogonal decomposition with interpolation (PODI): Gaussian process regression (GPR), radial basis functions interpolation (RBF) and feed-forward neural networks (NN). For every method, there is at least a cGM that performs slightly better than the cFFD in terms of accuracy, while reducing the parameters' space dimension from $\mathbf{p} = \mathbf{54}$ to $\mathbf{R} = \mathbf{15}$. The active subspace (AS) dimension chosen is $\mathbf{r}_{\text{AS}} = \mathbf{1}$. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

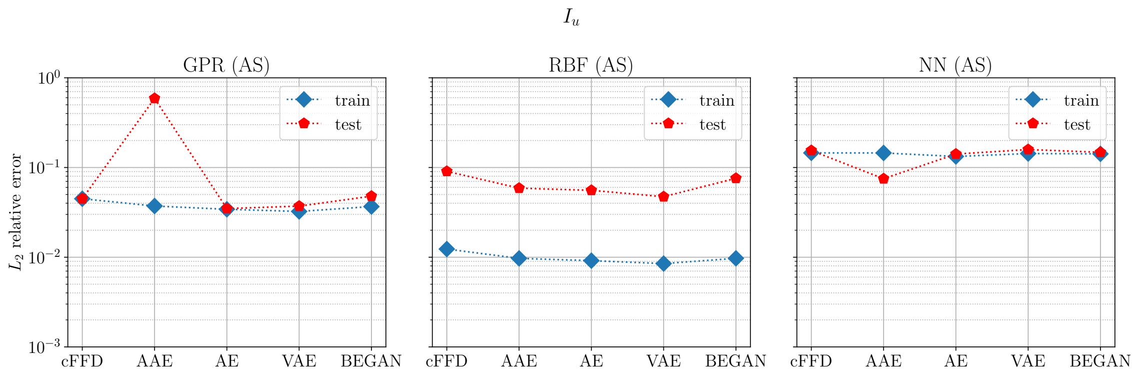


Fig. 4.9 **SB.** In this figure we show the ROM-PODI performance on the training and test datasets coupled with AS dimension reduction in the space of parameters. It can be seen that the latent dimension can be reduced further with AS $\mathbf{r}_{\text{AS}} = \mathbf{1}$ without compromising too much the accuracy. The AS response surface is built over the GPR, RBF, NN interpolations/regressions with an additional GPR from the active one-dimensional variables to the same outputs. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

Water and air are considered as isothermal immiscible fluids. The system of partial equations to be solved is the following

$$\partial_t(\rho\mathbf{u}) + \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u}) + \nabla p - \rho g - \nabla\cdot\nu\nabla\mathbf{u} - \nabla\cdot R = 0, \qquad (\mathbf{x},t)\in\Omega\times[0,T] \tag{4.38a}$$

$$\nabla\cdot\mathbf{u} = 0, \qquad (\mathbf{x},t)\in\Omega\times[0,T] \tag{4.38b}$$

$$\partial_t\alpha + \nabla\cdot(\mathbf{u}\alpha) = 0, \qquad (\mathbf{x},t)\in\Omega\times[0,T] \tag{4.38c}$$

$$\alpha\rho_W + (1-\alpha)\rho_A = \rho, \qquad (\mathbf{x},t)\in\Omega\times[0,T] \tag{4.38d}$$

$$\alpha\nu_W + (1-\alpha)\nu_A = \nu, \qquad (\mathbf{x},t)\in\Omega\times[0,T] \tag{4.38e}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, ρ_W, ρ_A are the densities of water and air, ν_W, ν_A are the dynamic viscosities of water and air, R is the Reynolds' stress tensor, g is the acceleration of gravity and α represents the interphase between the fluids with values from 0 (inside the air phase) to 1 (inside the water phase). The turbulence is modelled with the $\kappa - \omega$ Shear Stress Transport (SST) model [179]. The initial conditions are:

$$\mathbf{u}(\mathbf{x}) = (U_0, 0, 0), \qquad \mathbf{x}\in\Omega \tag{4.38f}$$

$$p(\mathbf{x}) = 0, \qquad \mathbf{x}\in\Omega \tag{4.38g}$$

$$\alpha(\mathbf{x}) = \alpha_0(\mathbf{x}), \qquad \mathbf{x}\in\Omega \tag{4.38h}$$

where $U_0 \in \mathbb{R}$ is the initial velocity, α_0 is the initial water level and the boundary conditions are specified in the `OpenFoam` DTCHull tutorial of the *interFoam* solver. We search for steady-state solutions with a final pseudo time instant $T = 4000$. The average physical quantities we will evaluate such as the drag and the angular momentum along the z-axis are obtained as the mean over the last 500 pseudo-time instants.

Some physical fields of interest at the final pseudo-time instant are shown in Figure 4.10 for the effective pressure and the velocity magnitude, in Figure 4.11 for the effective pressure on the hull surface and the interphase field α and in Figure 4.12 for the interphase field α on the whole domain. Only half of the hull is employed for the numerical simulations.

For the DTCHull test case **HB** we consider the moments of inertia of the hull, the angular momentum along the z-axis of the hull, the surface area of the hull's bulb, and the drag on the hull:

$$I_{xx} = \int_{\Omega_{\text{hull}}} r_X^2(\mathbf{x})d\mathbf{x},\ I_{yy} = \int_{\Omega_{\text{hull}}} r_Y^2(\mathbf{x})d\mathbf{x},\ I_{zz} = \int_{\Omega_{\text{hull}}} r_Z^2(\mathbf{x})d\mathbf{x}, \qquad \text{(principal moments of inertia)} \tag{4.39a}$$

$$I_{xy} = \int_{\Omega_{\text{hull}}} r_X(\mathbf{x})r_Y(\mathbf{x})d\mathbf{x},\ I_{xz} = \int_{\Omega_{\text{hull}}} r_X(\mathbf{x})r_Z(\mathbf{x})d\mathbf{x},\ I_{yz} = \int_{\Omega_{\text{hull}}} r_Y(\mathbf{x})r_Z(\mathbf{x})d\mathbf{x}, \qquad \text{(moments of inertia)} \tag{4.39b}$$

$$M_z = \int_{\Omega_{\text{hull}}} r_z^2(\mathbf{x}) \times \mathbf{u}\, d\mathbf{x}, \qquad \text{(angular momentum along z-axis)} \tag{4.39c}$$

$$A_{\text{bulb}} = \int_{\partial\Omega_{bulb}} d\sigma, \qquad \text{(surface area of the bulb)} \tag{4.39d}$$

$$c_{\text{d}} = \frac{1}{A_{hull}(\mathbf{u}\cdot\mathbf{e}_x)^2}\left(\oint_{\delta\Omega_{\text{hull}}} p\mathbf{n} - [\nu_\alpha(\nabla\mathbf{u}+\nabla\mathbf{u}^T)]\mathbf{n}ds\right)\cdot\mathbf{e}_x, \qquad \text{(drag on the hull)} \tag{4.39e}$$

where the moments of inertia $I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}$ and surface area A_{bulb} are evaluated from the discrete STL file, and the angular momentum along the z-axis M_z and the drag coefficient c_d are evaluated from the computational mesh.

The number of training and test samples are $\mathbf{n}_{\text{train}} = \mathbf{400}$ and $n_{\text{train}} = 200$, respectively. For model order reduction, the number of training and test components are instead $\mathbf{n}_{\text{ROM,train}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$, respectively. We use $\mathbf{r}_{\text{PCA}} = \mathbf{140}$ PCA modes for preprocessing inside the cGMs and $\mathbf{r}_{\text{POD}} = \mathbf{3}$ modes to perform model order reduction. The parameters' space dimension of cFFD is $\mathbf{p} = \mathbf{84}$, while the latent space dimension of the cGMs is $\mathbf{R} = \mathbf{10}$. Some deformations of the cGMs introduced are shown in Figure 4.13: it is shown the overlapping hull's bulbs from the reference STL (in blue) and the STL files generated by cFFD and the other cGMs (in red). The results with respect to the geometrical and physical metrics defined previously are shown in Table 4.2.

Qualitatively the histograms of each architecture are reported in Figure 4.17 for the M_z angular momentum along the z-axis and in Figure 4.16 for the c_d drag coefficient. There is an evident bias of the cGMs distributions with respect to the cFFD's one when considering the drag coefficient. Possible reasons for this bias are the sharp edges introduced with the cFFD deformations on the gluing sites of

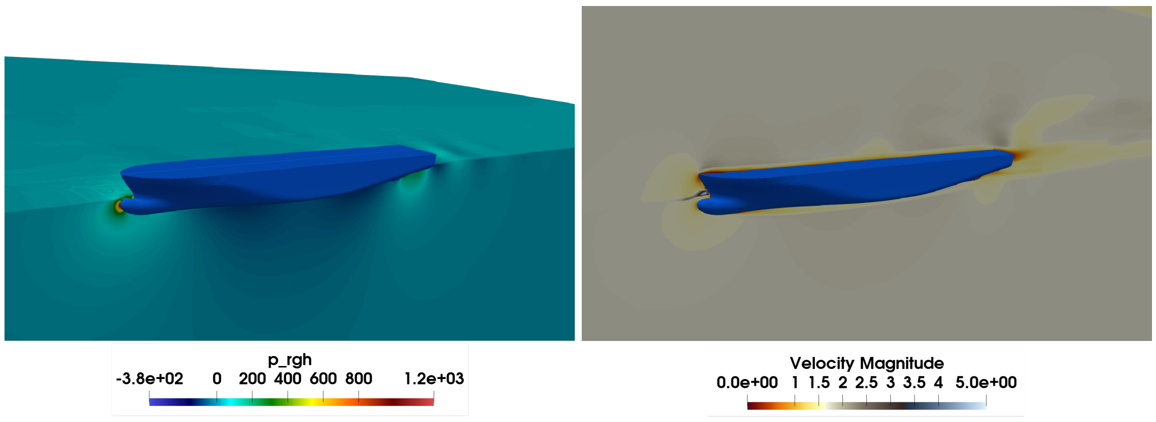

p_rgh
-3.8e+02 0 200 400 600 800 1.2e+03
Velocity Magnitude
0.0e+00 1 1.5 2 2.5 3 3.5 4 5.0e+00

Fig. 4.10 HB. **Left:** pressure field minus the hydrostatic pressure contribution computed on the last pseudo-time instant $T = 4000$. **Right:** velocity magnitude computed on the last pseudo-time instant $T = 4000$.

the bulb on the hull and the coarse computational mesh employed for the numerical simulations with respect to the resolution of the STL files used to train the cGMs.

The non-intrusive reduced order models implemented with GPR-PODI, RBF-PODI and NN-PODI are shown in Figures 4.18 for the effective pressure and 4.20 for the velocity magnitude on the whole computational domain Ω. The original parameters space's dimension of cFFD changes from $\mathbf{p} = \mathbf{84}$ to $\mathbf{R} = \mathbf{10}$ for the cGMs, while keeping more or less the same accuracy on the surrogate models for the pressure and velocity fields. We apply a further level of reduction in the space of parameters with the AS method on the latent space of $\mathbf{R} = \mathbf{10}$ coordinates of the cGMs: the new parameter space is a one-dimensional active subspace for each variable. The accuracy of the AS response surfaces is almost the same for the pressure and the velocity magnitude while reducing the parameter space's dimension from $\mathbf{R} = \mathbf{10}$ to $\mathbf{r}_{\text{AS}} = \mathbf{1}$. For simplicity, we show the AS response surface with dimension $\mathbf{r}_{\text{AS}} = \mathbf{1}$, even if from the plot in Figure 4.22 of the first 20 eigenvalues of the uncentered covariance matrix from equation (4.30), the spectral gap [60] suggests $\mathbf{r}_{\text{AS}} = \mathbf{2}$.

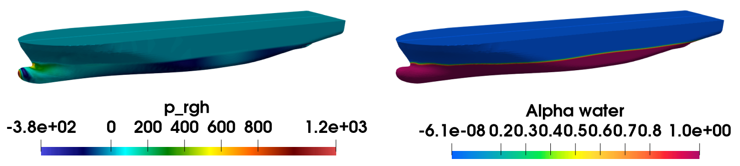


Fig. 4.11 **HB. Left:** pressure field minus the hydrostatic pressure contribution computed on the last pseudo-time instant $T = 4000$ on the hull's surface. **Right:** interphase field α on the hull at the final pseudo-time instant $T = 4000$: the value of 1 correspond to the water phase and the value of 0 to the air phase.

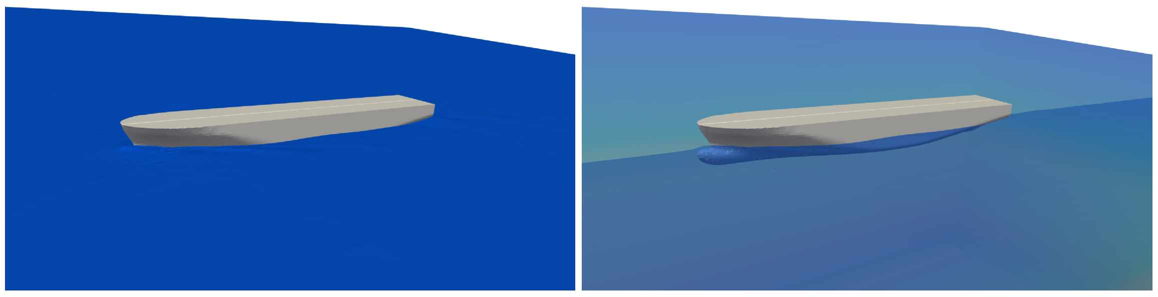


Fig. 4.12 **HB.** Interphase field α on the whole domain at the final pseudo-time instant $T = 4000$: only the values less than 0.5 of the field α are shown, representing the water level.

Table 4.2 **HB.** In this table we show the evaluation metrics of the Duisburg test case defined in equations (4.39):$I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}$ are the components of the inertia tensor, A_{hull} area of the surface of the bulb, c_d is the drag coefficient evaluated on the hull, M_z is the angular momentum along the z-axis. The metrics $I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}, A_{\text{bulb}}$ are obtained from $\mathbf{n}_{\text{test}} = \mathbf{200}$ test samples, while the metrics c_d, M_z are based on $n_{\text{ROM,test}} = 20$ test samples. The model that has overall the best performance is the AE, since the BEGAN produces too many out-of-training distribution samples as can be seen comparing the values of the total variance: the variance of the training data from cFFD is **1.2e-03**.

	AE	AAE	VAE	BEGAN
$JSD(A_{\text{bulb}})$	**1.2e-01**	2.3e-01	3.6e-01	5.2e-01
$JSD(I_{xx})$	**1.6e-01**	3.0e-01	3.8e-01	6.0e-01
$JSD(I_{xy})$	5.5e-01	6.9e-01	**4.7e-01**	6.7e-01
$JSD(I_{xz})$	2.1e-01	2.1e-01	5.0e-01	**2.0e-01**
$JSD(I_{yy})$	**2.4e-01**	4.9e-01	8.8e-01	9.1e-01
$JSD(I_{yz})$	8.7e-01	5.6e-01	6.6e-01	**4.8e-01**
$JSD(I_{zz})$	**1.5e-01**	4.5e-10	5.6e-1	3.2e-01
$JSD(c_d)$	9.1e-01	9.4e-01	8.5e-01	**8.1e-01**
$JSD(M_z)$	3.4e-01	2.9e-01	2.9e-01	**2.6e-01**
Var	1.2e-03	2.4e-03	4e-04	7.0e-03

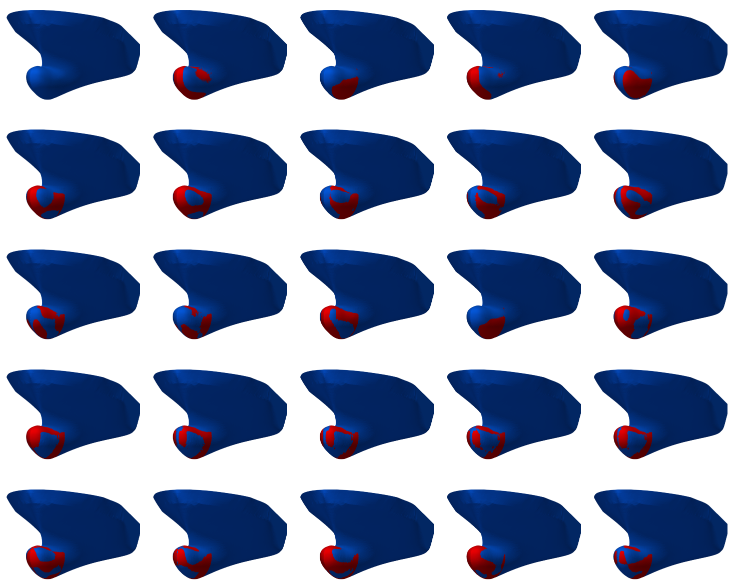


Fig. 4.13 **HB.** In this figure is shown the overlapping of the newly sampled bulbs from cFFD and cGMs (in red) with respect to the reference STL file used for this test case (in blue). The reference geometry is shown on the top left. Each row corresponds to a different architecture: the first row refers to constrained cFFD of section 4.2.2, the second to the adversarial autoencoder of paragraph 4.3.1, the third to the simple autoencoder of paragraph 4.3.1, the fourth to the beta variational autoencoder of paragraph 4.3.1 and the last to the boundary equilibrium generative adversarial networks of paragraph 4.3.1. All the generative models implement the multilinear constraints enforcing layer of section 4.3 to preserve the volume.

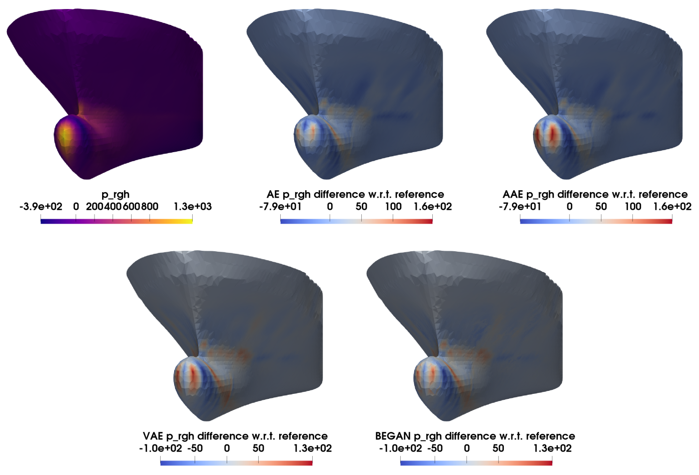

p_rgh
-3.9e+02 0 200 400 600 800 1.3e+03
AE p_rgh difference w.r.t. reference
-7.9e+01 0 50 100 1.6e+02
AAE p_rgh difference w.r.t. reference
-7.9e+01 0 50 100 1.6e+02
VAE p_rgh difference w.r.t. reference
-1.0e+02 -50 0 50 1.3e+02
BEGAN p_rgh difference w.r.t. reference
-1.0e+02 -50 0 50 1.3e+02

Fig. 4.14 **HB.** Difference of the effective pressure field on the reference computational mesh with the effective pressure fields on different geometries sampled from constrained generative models, namely from top left to bottom right from the simple autoencoder, from the adversarial autoencoder, from the beta variational autoencoder and from the boundary equilibrium generative adversarial network. The geometrical constraint preserved is the bulb's and thus the hull's volume.

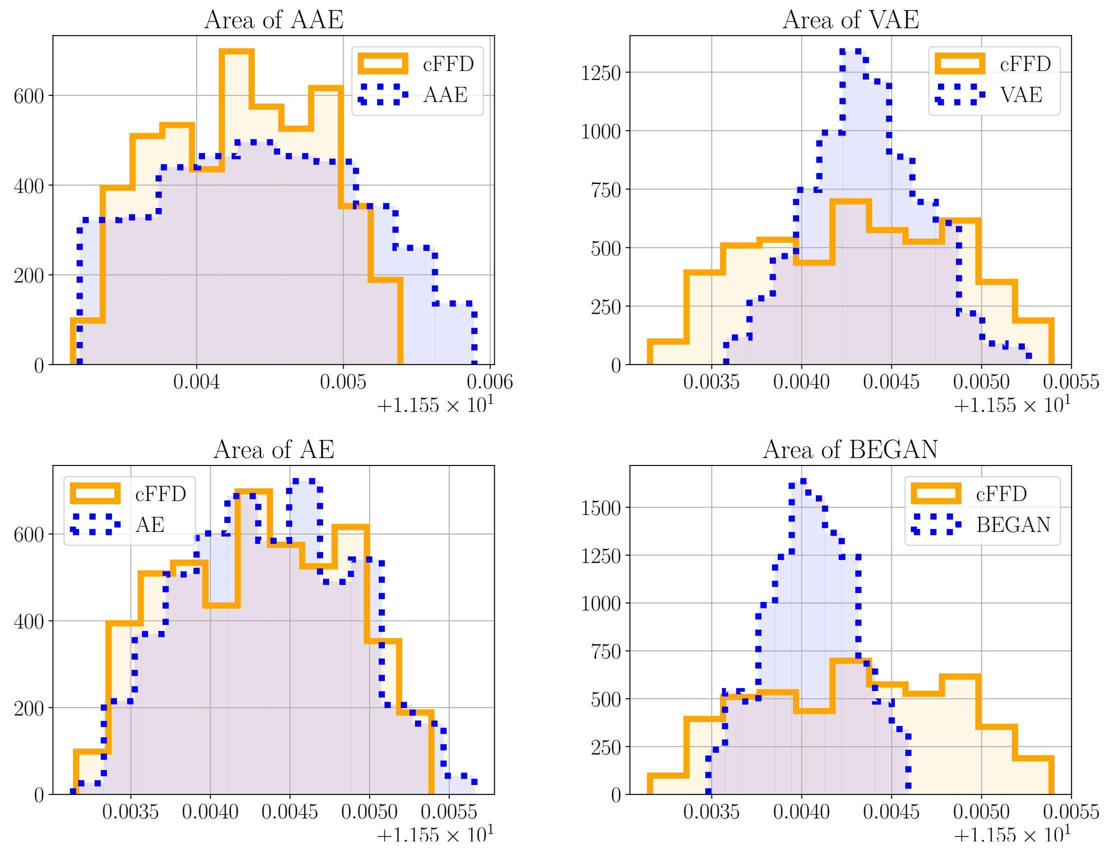


Fig. 4.15 **HB.** In this figure the histograms of the area of the bulb's deformed with cFFD or with cGMs are shown. The histogram area intersection is more than 50%. For a quantitative measure see Table 4.2. The histograms are obtained from the $\mathbf{n}_{\text{test}} = \mathbf{200}$ test samples.

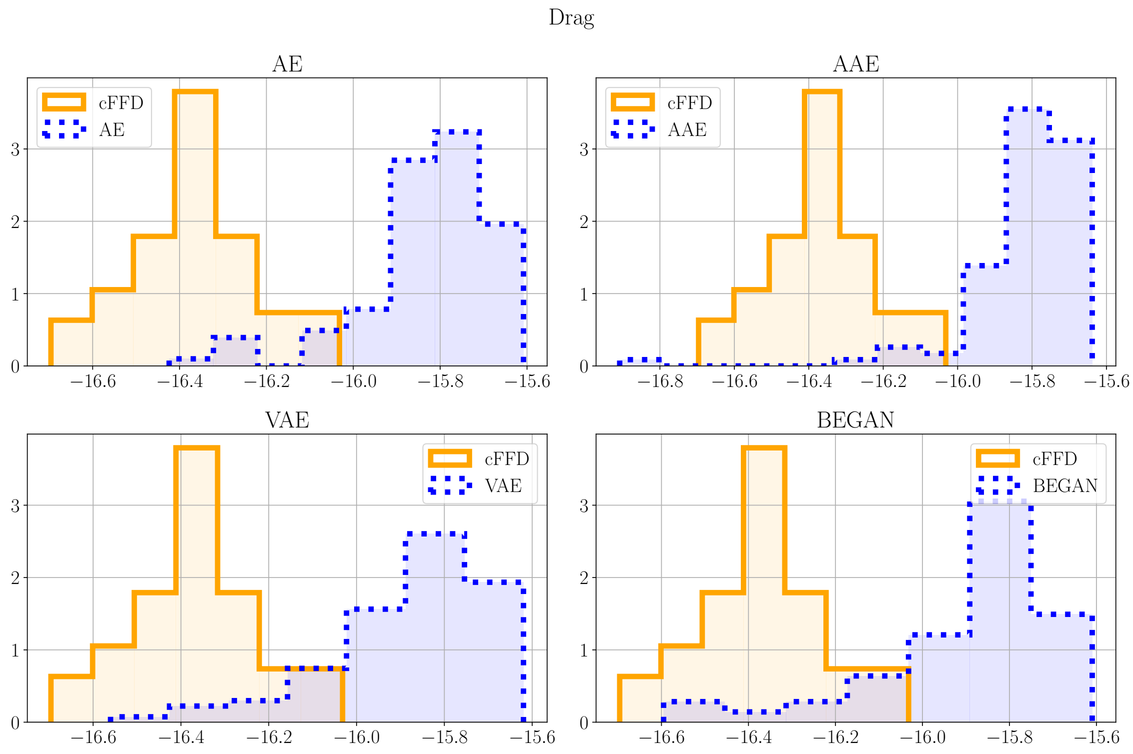


Fig. 4.16 **HB.** In this figure the histograms of the drag coefficient from cFFD and from the cGMs are shown. It can be seen that the drag coefficient is very sensitive to the deformations of the bulb and that our generated geometries are probably biased towards higher values of the drag. For a quantitative measure see Table 4.2. The histograms are obtained from the $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ test samples.

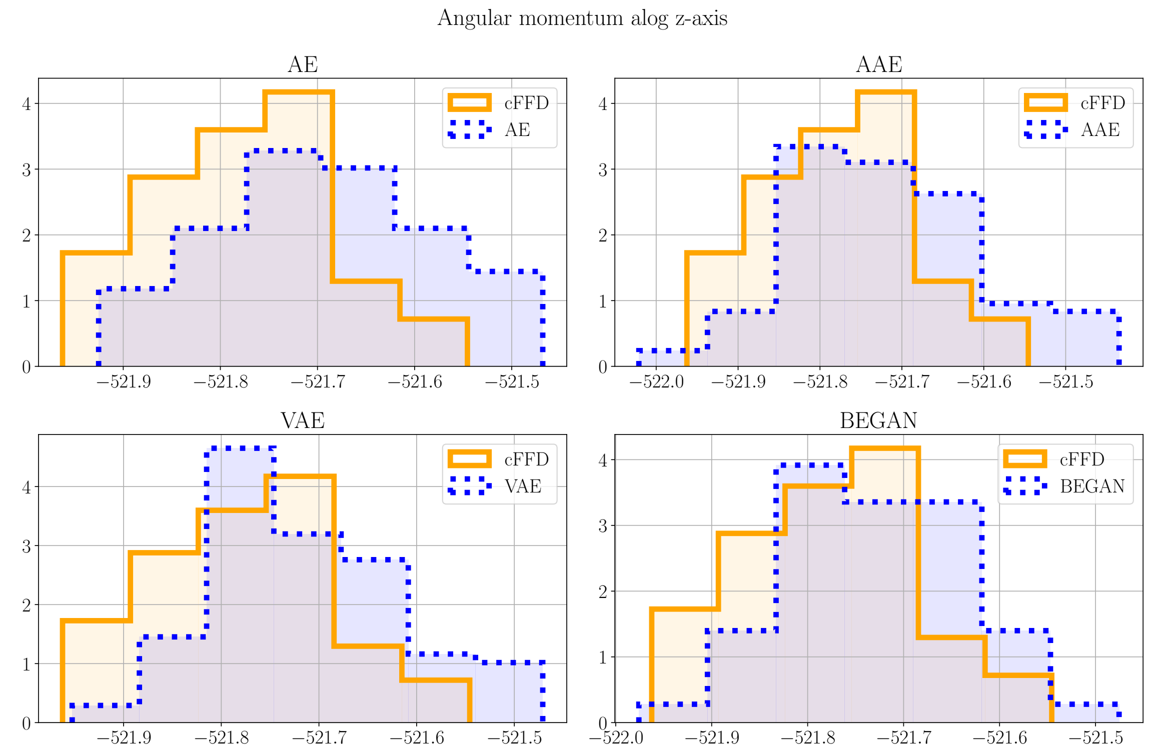


Fig. 4.17 **HB.** In this figure the histograms of the angular momentum along the z-axis M_z from the mesh obtained from cFFD or cGMs are shown. The histogram area intersection is more than 50%. For a quantitative measure see Table 4.2. The histograms are obtained from the $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ test samples.

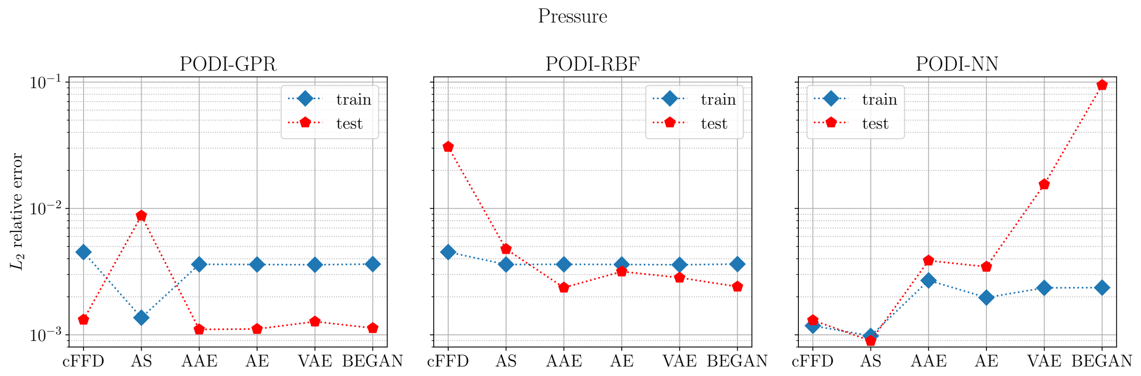


Fig. 4.18 **HB.** Here we show the ROM performance on the pressure on the whole domain over various datasets, using different interpolation and regression techniques combined with proper orthogonal decomposition with interpolation (PODI): Gaussian process regression (GPR), radial basis functions interpolation (RBF) and feed-forward neural networks (NN). For every method, there is at least a cGM that performs slightly better than the cFFD in terms of accuracy, while reducing the parameters' space dimension from $\mathbf{p} = \mathbf{84}$ to $\mathbf{R} = \mathbf{10}$. The active subspace (AS) dimension chosen is $\mathbf{r}_{\text{AS}} = \mathbf{1}$. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

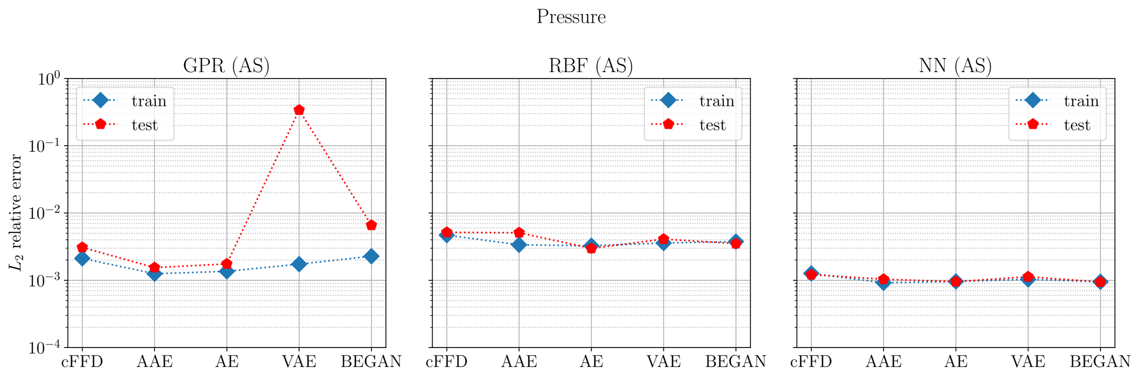


Fig. 4.19 **HB.** In this figure we show the ROM-PODI performance, for the effective pressure on the whole domain, on the training and test datasets coupled with AS dimension reduction in the space of parameters. It can be seen that the latent dimension can be reduced further with AS $\mathbf{r}_{\text{AS}} = \mathbf{1}$ without compromising too much the accuracy. The AS response surface is built over the GPR, RBF, NN interpolations/regressions with an additional GPR from the active one-dimensional variables to the same outputs. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

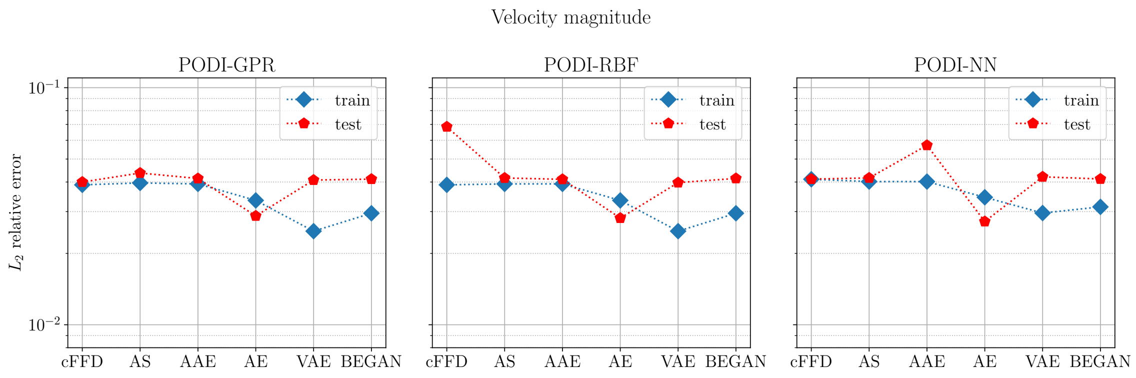


Fig. 4.20 **HB.** Here we show the ROM performance on the velocity magnitude on the whole domain over various datasets, using different interpolation and regression techniques combined with proper orthogonal decomposition with interpolation (PODI): Gaussian process regression (GPR), radial basis functions interpolation (RBF) and feed-forward neural networks (NN). For every method, there is at least a cGM that performs slightly better than the cFFD in terms of accuracy, while reducing the parameters' space dimension from $\mathbf{p} = \mathbf{84}$ to $\mathbf{R} = \mathbf{10}$. The active subspace (AS) dimension chosen is $\mathbf{r}_{\text{AS}} = \mathbf{1}$. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

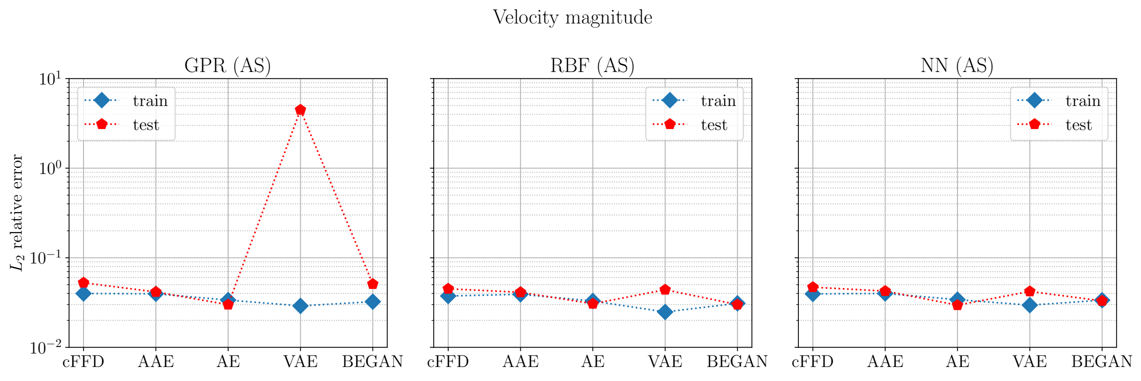


Fig. 4.21 **HB.** In this figure we show the ROM-PODI performance, for the velocity magnitude on the whole domain, on the training and test datasets coupled with AS dimension reduction in the space of parameters. It can be seen that the latent dimension can be reduced further with AS $\mathbf{r}_{\text{AS}} = \mathbf{1}$ without compromising too much the accuracy. The AS response surface is built over the GPR, RBF, NN interpolations/regressions with an additional GPR from the active one-dimensional variables to the same outputs. The training and test errors are evaluated on $\mathbf{n}_{\text{ROM,test}} = \mathbf{80}$ and $\mathbf{n}_{\text{ROM,test}} = \mathbf{20}$ training and test samples.

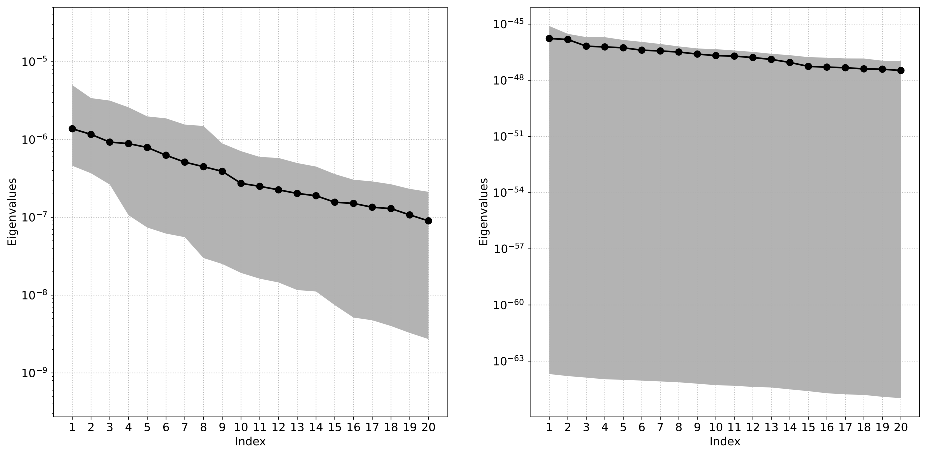


Fig. 4.22 **SB and HB.** In this figure, we show the first 20 eigenvalues of the Active Subspace method for the SB test case on the left and the HB test case on the right. The shaded grey areas are delimited by the minimum and maximum values of the 100 bootstrap replicates. The black line corresponds to the average value. The spectral gaps suggest an AS dimension equal to $r_{\text{AS}} = 9$ and $r_{\text{AS}} = 2$, for the SB and HB test cases, respectively.

The speedup for the generation of the geometries employing the cGMs instead of cFFD is around 360. The speedup of PODI non-intrusive model order reduction with respect to the full-order simulations is around 432000.

4.5.3 Generative models' training specifics and architectures

The following specifics are the same for the test cases **SB** of section 4.5.1 and **HB** of section 4.5.2. Every hidden unit of our models is composed of a linear layer, a normalization layer (typically batch normalization), an Activation Layer (typically ReLU), and a dropout Layer. The AdamW [166] optimizer with 500 epochs is employed with a learning rate of $1e-3$ without any scheduler. The batch size is 200. The entire dataset size is 600. We use $n_{\text{train}} = 400$ samples for the training set and the remaining for the test set. An Nvidia RTX 3050 has been used for the training. In the following Tables 4.3, 4.4, 4.5 and 4.6, *Act.* stands for activation, *Norm.* for normalization and *Drp.* for dropout: they summarize in order, the architectures of the AAE, AE, VAE and BEGAN GMs we employed for both the test cases in section 4.5.

Table 4.3 Adversarial Autoencoder Structure. The number of PCA modes used is $r_{\text{PCA}} = 140$ for the rabbit and $r_{\text{PCA}} = 30$ for the hull, M is the number of mesh points and it is equal to 145821 for the **SB** test case and 5000 for the **HB** test case.

Encoder	Act.	Weights	Norm.	Drp.
None	PCA	[M, r_{PCA}]	None	None
Linear	ReLU	[r_{PCA}, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0,1
Linear	None	[500, 10]	Batch	None

Decoder	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	None	[500, r_{PCA}]	None	None
None	PCA (inverse)	[r_{PCA}, M]	None	None

Discriminator	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.95
Linear	ReLU	[500, 500]	Batch	0.95
Linear	ReLU	[500, 500]	Batch	0.95
Linear	ReLU	[500, 500]	Batch	0.95
Linear	ReLU	[500, 500]	Batch	0.95
Linear	ReLU	[500, 500]	Batch	0.95
Linear	Sigmoid	[500, 1]	None	None

We report the PODI-NN specifics. The training set consists of $\mathbf{n}_{\text{ROM,train}} = \mathbf{80}$ samples and the test set of **HB** $fn_{\text{ROM,test}} = 20$ samples. The AdamW [166] optimizer with 1000 epochs has been used, with a learning rate of $1e-3$ without any scheduler.

Table 4.4 Autoencoder Structure. The number of PCA modes used is $r_{\text{PCA}} = 140$ for the rabbit and $r_{\text{PCA}} = 30$ for the hull, M is the number of mesh points and it is equal to 145821 for the **SB** test case and 5000 for the **HB** test case.

Encoder	Act.	Weights	Norm.	Drp.
None	PCA	[M, r_{PCA}]	None	None
Linear	ReLU	[r_{PCA}, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0,1
Linear	None	[500, 10]	Batch	None

Decoder	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	None	[500, r_{PCA}]	None	None
None	PCA (inverse)	[r_{PCA}, M]	None	None

Table 4.5 Beta Variational Autoencoder Structure. The number of PCA modes used is $r_{\text{PCA}} = 140$ for the rabbit and $r_{\text{PCA}} = 30$ for the hull, M is the number of mesh points and it is equal to 145821 for the **SB** test case and 5000 for the **HB** test case.

Encoder	Act.	Weights	Norm.	Drp.
None	PCA	[M, r_{PCA}]	None	None
Linear	ReLU	[r_{PCA}, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0,1
Linear	None	[500, 10]	Batch	None

Decoder	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	None	[500, r_{PCA}]	None	None
None	PCA (inverse)	[r_{PCA}, M]	None	None

Table 4.6 Boundary Equilibrium Generative Adversarial Network Structure. The number of PCA modes used is $r_{\text{PCA}} = 140$ for the rabbit and $r_{\text{PCA}} = 30$ for the hull, M is the number of mesh points and it is equal to 145821 for the **SB** test case and 5000 for the **HB** test case. The terms *DisEnc* and *DisDec* stand for the discriminator's encoder and decoder.

DisEnc	Act.	Weights	Norm.	Drp.
None	PCA	[M, r_{PCA}]	None	None
Linear	ReLU	[r_{PCA}, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0,1
Linear	None	[500, 10]	Batch	None

DisDec	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	None	[500, r_{PCA}]	None	None
None	PCA (inverse)	[r_{PCA}, M]	None	None

Generator	Act.	Weights	Norm.	Drp.
Linear	ReLU	[10, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	ReLU	[500, 500]	Batch	0.1
Linear	None	[500, r_{PCA}]	None	None
None	PCA (inverse)	[r_{PCA}, M]	None	None

Table 4.7 PODI-NN interpolation: $r_{\text{POD}} = 3$ for both test cases **SB** and **HB**, M is the number of degrees of freedom and P the number of parameters that can vary if cFFD ($P = p = 54$ for **SB** or $P = p = 84$ for **HB**), or cGMs ($P = R = 15$ for **SB** or $P = R = 10$ for **HB**) are considered.

Regressor	Activation	Weights	Norm.	Drp.
Linear	ReLU	[P, 2000]	None	0
Linear	ReLU	[2000, 2000]	None	0
Linear	ReLU	[2000, r_{POD}]	None	0
POD reconstruction	-	[r_{POD}, M]	None	0

4.6 Discussion

We address some questions that may arise, the critical parts of the developed methodology and other observations:

- more complex 3d objects. The respective distributions of the test cases **SB** and **HB** are relatively easy to approximate with GMs, in fact, we need only $n_{\text{train}} = 400$ training data. Provided that enough data and a sufficient computational budget are available, our methodology can be extended to more complex distributions of 3d objects. For example, points clouds with different numbers of points for each geometry or topological changes can still be approximated by GMs. Eventually, Graph Neural networks can be employed [34].

- validate the results with ad hoc metrics. Especially when cGMs are employed for real-world applications, guarantees that the new geometries are not biased or satisfy certain requirements must be verified. For our test cases, we devised several geometrical and physical metrics relevant to validate our cGMs. In particular, from the drag coefficient histograms in Figure 4.16 we could detect a bias that changed the mean value of the distributions of the cGMs. In our case, it probably depends on the application of RBF interpolation to deform the reference computational mesh, on the way the bulb was extracted from the hull and on the coarseness of the mesh. As mentioned in section 4.4, RBF interpolation can be avoided and the cGMs can be directly trained on specific regions of the computational mesh as it was done for the **SB** test case.

- the constraint enforcing layer is independent of the numerical methods used to generate the training datasets. For our test cases, we used cFFD because it is relatively simple to implement, even if the constrained deformations it provides are not particularly large. It is crucial to observe that the cGMs and their multilinear constraints enforcing layers do not depend on how the training data are obtained. For example, as long as linear or multilinear and not nonlinear geometrical constraints are enforced, also other volume-preserving techniques like [262, 83] can be used.

- intrusive model order reduction. It is clear that cGMs with their nonlinear layers introduce nonlinearities also in the numerical models at hand through the mesh deformations. From the point of view of intrusive model order reduction [228], this adds an ulterior complexity especially when the numerical model itself is linear. In this case, hyper-reduction methods must be implemented to completely lose the dependency on the number of degrees of freedom. Non-intusive reduced order models are instead not affected.

- principal component analysis as preprocessing for GMs. We remark that our choice to employ PCA modes to project the training datasets of the GMs onto lower dimensional spaces is valid only when the training distribution can be well approximated with linear subspaces. If this is not the case, an evident result is a high test error even though the training error is low because the GMs cannot generalize well. Moreover, with the projection of the training data through PCA modes, the spatial correlations are lost in favor of the correlations in the space of frequencies of PCA. These lost spatial correlations could instead be exploited with ad hoc architectures like convolutional layers on Cartesian grids or graph neural networks for general meshes.

4.7 Conclusions and perspectives

A novel approach to impose linear or multilinear constraints on classical generative models is tested successfully on two academic benchmarks: the Stanford bunny and the Duisburg test case's naval hull. We proposed to validate the results through specific geometrical and physical metrics of interest for the problem at hand. How to perform non-intrusive model order reduction with proper orthogonal decomposition with interpolation was shown. The benefits of coupling dimension reduction in the space of parameters with cGMs and model order reduction are clear.

Future directions of research include the approximation of more complex distributions with possibly different numbers of degrees of freedom per geometry or topological changes. A relevant task is to enforce nonlinear constraints without weighting too much on the computational cost of the training of the cGMs. We think that our methodology could speed up the generation of geometrically constrained 3d shapes and solve the problem of the lack of experimental data typical of some real-world applications.

Part II

Nonlinear model order reduction

Chapter 5

Friedrichs' systems

Friedrichs' systems (FS) are symmetric positive linear systems of first-order partial differential equations (PDEs), which provide a unified framework for describing various elliptic, parabolic and hyperbolic semi-linear PDEs such as the linearized Euler equations of gas dynamics, the equations of compressible linear elasticity and the Dirac-Klein-Gordon system. FS were studied to approximate PDEs of mixed elliptic and hyperbolic type in the same domain. For this and other reasons, the versatility of the discontinuous Galerkin method (DGM) represents the best approximation space for FS. A distributed memory solver for stationary FS is implemented in `deal.II`. The focus is on model order reduction. Since FS model hyperbolic PDEs, they often suffer from a slow Kolmogorov n-width decay. Two approaches to tackle this problem are developed. The first is domain decomposable reduced-order models (DD-ROMs). It is shown that the DGM offers a natural formulation of DD-ROMs, in particular regarding interface penalties, compared to the continuous finite element method. Also, new repartitioning strategies to obtain more efficient local approximations of the solution manifold are developed. The second approach involves graph neural networks used to infer the limit of a succession of projection-based linear ROMs corresponding to lower viscosity constants: the heuristic behind is to develop a multi-fidelity super-resolution paradigm to mimic the mathematical convergence to vanishing viscosity solutions while exploiting to the most interpretable and certified projection-based ROMs.

Contents

5.1 Literature review

Friedrichs' systems (FS) are a class of symmetric positive linear systems of first-order partial derivative equations (PDEs). They were introduced by Friedrichs [96] as a tool to study hyperbolic and elliptic phenomena in different parts of the domain within a unifying framework. The main ideas that allow recasting many models into the FS frameworks are the introduction of extra variables to lower the order of the higher derivatives and the linearization of nonlinear problems. FS are characterized by linear and positive operators and (non-uniquely defined) boundary operators that allow them to impose classical boundary conditions (BCs). Various works proved the uniqueness, existence and well–posedness of the FS in their strong, weak and ultraweak formulation and the necessary conditions to properly define the boundary operators [96, 215, 216, 85, 87, 8, 76].

In the last decades, different numerical discretizations of the FS have been proposed to approximate the analytical solutions. The strategies vary among finite volume [235] and discontinuous Galerkin (DGM) formulations [130, 140, 85, 84, 87, 86, 40, 49]. Along with the DGM discretization, also error estimation analysis that provides, according to the type of edge penalization, optimal or sub–optimal estimates, have been carried out [85, 84, 76]. We focus on the DGM method since it is more versatile to approximate both elliptic and hyperbolic PDEs and it fits naturally the framework of domain decomposable ROMs (DD-ROMs).

In the context of parametric PDEs, for multi–query tasks or real–time simulations, fast and reliable simulations of the same problem for different parameters are needed. This is especially true when the full-order models (FOMs) are based on expensive and high-order DGM discretizations. Reduced order models (ROMs) decrease the computational costs by looking for the solutions of unseen parametric instances on low-dimensional discretization spaces, called reduced basis spaces. This is possible because the new solutions to be predicted are expected to be highly correlated with the database of training DGM solutions used to build the reduced spaces. ROMs have been proven to be a powerful tool for many applications [207, 172, 126, 228]. In particular for linear problems, classical Galerkin and Petrov-Galerkin projection methods are very easy to set up and extremely convenient in terms of computational costs. FS are perfectly suited for such algorithms due to their linearity. Also, this is a preliminary step needed to reduce parametric nonlinear PDEs whose linearization results in FS. In the most simple formulation, we will apply singular value decomposition (SVD) to compress a database of snapshots and provide a reliable reduced order model (ROM), with standard *a posteriori* error estimators.

In the context of model order reduction, FS are particularly beneficial as theoretical frameworks for many reasons. They represent a new form of structure-preserving ROMs: the positive symmetric properties of FS are in fact easily inherited by the reduced numerical formulations. This advocates for the employment of FS for reduced order modelling whenever a PDE can be reformulated in the FS framework. This is the case for the Euler equations of gas dynamics, when they are written in terms of entropy variables [235, 193]. The same rationale is behind structure-preserving symplectic or Hamiltonian ROMs [127] and port-Hamiltonian ROMs [259, 22]. Moreover, since FS are often studied in their ultraweak formulation, they are good candidates for optimally stable error estimates [39] at the full-order level [40], also in a hybridized DGM implementation in [49], and at the reduced order level, similarly to what has been achieved in the works [28, 116, 124]. Finally, from the point of view of software design, the possibility to implement in a unique maintainable generic manner ROMs for PDEs ascribed to the class of FS is a convenient feature to search for.

Though being linear, FS are hyperbolic systems and often show an advection dominated character, which is not easily approximable through a simple proper orthogonal decomposition (POD). This leads to a slow Kolmogorov n-width (KnW) decay that results in very inefficient approximations of the reduced models. Several approaches have been studied to overcome this difficulty [242, 135, 198, 217, 45, 43, 7, 161, 64, 254, 136].

A strategy that has been developed to reduce PDEs solved numerically with domain decomposition approaches, like fluid-structure interaction systems, are domain decomposable ROMS (DD-ROMS). The initial formulations [172, 173, 134, 82, 134] involved continuous finite elements discretizations for which new ways to couple the solutions restricted to different subdomains needed to be devised, especially to enforce continuity at the interfaces. We show that the DGM imposes naturally flux interface penalties from the full-order discretizations and it is, thus, amenable for straightforward implementations of DD-ROMs. From the point of view of solution manifold approximability and so KnW decay, DD-ROMs are based on local linear approximants that are employed to reach a higher accuracy for unseen solutions. This is useful when the computational domain is divided into subdomains that are independently affected by the parametric instances. The typical case in which this may happen is parametric models for which discontinuous values of the parameters over fixed subdomains cause non-correlated responses on their respective subdomains. Similar cases will be studied in section 5.5.3. Another example is represented by parametric fluid-structure interaction systems in which the parameters cause complex interdependencies between the structure and fluid components in favor of partitioned linear solution manifold approximations (SVD is performed separately for the fluid, for the structure and for the interface) rather than monolithic ones. In our implementation of DD-ROMs, we exploit the partitions obtained from the distributed memory solver in `deal.II`. Since these domain decompositions typically satisfy constraints related to the computational efficiency, we devise some strategies to repartition the domain responding to solution manifold approximability concerns instead. Another work that implements this is [274] where the Reynolds stress tensor is employed, among others, as indicator for partitioning the computational domain. Similarly, we develop new indicators.

Another way to approach the problem of a slow KnW decay is exploiting the mathematical proofs of existence of vanishing viscosity solutions [187, 152, 80, 107]. In fact, solutions to hyperbolic problems can be obtained as a limit process of solutions associated with viscosity terms approaching zero. The crucial point is that ROMs associated with larger viscosity values may not suffer from a slow Kolmogorov n-width decay. Hence, we can set up classical projection based ROMs for the high viscosity solutions, and use graph neural networks (GNNs) [245] only to infer the vanishing viscosity solution in a very efficient manner. This procedure can be applied also to more general hyperbolic problems, not necessarily FS. The key features of this new methodology are the following: the employment of computationally heavy graph neural networks is reduced to a minimum and, at the same time, interpretable certified projection ROMs are exploited as much as possible in their regime of accurate linear approximability. In fact, GNNs, used generally to perform non-intrusive MOR, have high training computational costs and they are employed mainly for small academic benchmarks in terms of numbers of degrees of freedom, up to now. We avoid these high computational efforts with our multi-fidelity formulation: the GNNs are employed only to infer the vanishing viscosity solutions from the previous higher viscosity level, not to approximate and perform dimension reduction of the entire solution manifold. The overhead is the collection of additional full-order snapshots corresponding to high viscosity values, but this can also be performed on coarser meshes as it will be done in section 5.6. Moreover, the support of our GNNs is the DGM discretization space, so, we can enrich the typical machine learning framework of GNNs with data structure and operators from numerical analysis. We validate the use of data augmentation with numerical filters (discretized Laplacian, gradients operators), as proposed in [245].

In brief, we summarize our contributions with the present work:

- structure-preserving model order reduction for Friedrichs' systems. We synthetically describe the realization of ROMs for FS and define standard *a posteriori* error estimators. Hints towards the implementation of optimally stable ROMs are highlighted.

- domain decomposable reduced-order models for full-order models discretized with the discontinuous Galerkin method. We introduce DD-ROMs for DGM discretizations and introduce novel indicators to repartition the computational domain with the aim of obtaining more efficient local solution manifold approximants.

- surrogate modelling of vanishing viscosity solutions with graph neural networks. We propose a new framework for the MOR of parametric hyperbolic PDEs with a slow Kolmogorov n-width decay.

The topics addressed in this work are presented as follows. In Section 5.2, we introduce the definition of FS and well–posedness results and we will provide several examples of models that fall into this framework: the Maxwell equations in stationary regime, the equations of linear compressible elasticity and the advection diffusion reaction equations. Then, we provide a DGM discretization of the FS following [76] with related error estimates in Section 5.3. In Section 5.4, we introduce the projection-based MOR technique and some error bounds that can be effectively used. In Section 5.5, we will discuss a new implementation of domain decomposable ROMs for FOMs discretized with the DGM and we will test the approach on three parametric models. In Section 5.6, we introduce the concept of vanishing viscosity solutions and how graph neural network are exploited to overcome the problem of a slow Kolmogorov n-width decay. We will provide some numerical tests to show the effectiveness of the proposed approach. Finally, in Section 5.7 we summarize our results and we suggest further directions of research.

5.2 Friedrichs' systems

In this section, we will provide a summary of FS theory: their definition, existence, uniqueness and well-posedness results, their weak and ultraweak forms and many PDEs which can be rewritten into FS. The following discussion collects many results from [96, 215, 133, 216, 132, 140, 235, 140, 84, 87, 86, 8, 40, 76], but we will follow the notation in [76]. Let us represent with d the ambient space dimension and with $m \geq 1$ the number of equations of the FS. We consider a connected Lipschitz domain $\Omega \subset \mathbb{R}^d$, with boundary $\partial\Omega$ and outward unit normal $\mathbf{n} : \partial\Omega \to \mathbb{R}^d$.

A FS is defined through $(d+1)$ matrix-valued fields $A^0, A^1 \dots, A^d \in [L^\infty(\Omega)]^{m\times m}$ and the following differential operators $\mathcal{X}, A, \tilde{A} : \Omega \to \mathbb{R}^{m\times m}$. We suppose that $\mathcal{X} \in [L^\infty(\Omega)]^{m\times m}$ and define

$$\mathcal{X} = \sum_{k=1}^{d} \partial_k A^k\,, \qquad A = A^0 + \sum_{i=1}^{d} A^i \partial_i\,, \qquad \tilde{A} = \left(A^0\right)^t - \mathcal{X} - \sum_{i=1}^{d} A^i \partial_i\,, \tag{5.1}$$

assuming that

$$A^k = (A^k)^T \text{ a.e. in } \Omega, \qquad \text{for } k = 1,\dots,d, \qquad \text{(symmetry property)} \tag{5.2a}$$

$$A^0 + (A^0)^T - \mathcal{X} \text{ is u.p.d.} \quad \text{a.e. in } \Omega, \qquad \text{(positivity property)} \tag{5.2b}$$

thus, the name **symmetric positive operators** or **Friedrichs operators**, which is used to refer to (A, $\tilde{A}$). We recall that the operator in (5.2b) is uniformly positive definite (u.p.d) if and only if

$$\exists \mu > 0 : A^0 + (A^0)^T - \mathcal{X} > 2\mu_0 \mathbb{I} \quad \text{a.e. in } \Omega. \tag{5.3}$$

If this property is not satisfied, it can be sometimes recovered as shown in section 5.2.1. A weaker condition can be required for two-field systems [86].

The boundary conditions are expressed through two boundary operators $\mathcal{D} : \partial\Omega \to \mathbb{R}^{m\times m}$ with

$$\mathcal{D} = \sum_{k=1}^{d} n_k A^k, \qquad \text{a.e. in } \partial\Omega \tag{5.4}$$

and $\mathcal{M} : \partial\Omega \to \mathbb{R}^{m\times m}$ satisfying the following **admissible boundary conditions**

$$\mathcal{M} \quad \text{is nonnegative} \quad \text{a.e. on} \quad \partial\Omega, \qquad \text{(monotonicity property)} \tag{5.5a}$$

$$\ker(\mathcal{D}-\mathcal{M}) + \ker(\mathcal{D}+\mathcal{M}) = \mathbb{R}^m \quad \text{a.e. on} \quad \partial\Omega. \qquad \text{(strict adjointness property)} \tag{5.5b}$$

Remark 11 (Strict adjointness). The term strict adjointness property comes from Jensen [140, Theorem 31]. The strict adjointness property is needed for the solution of the ultra-weak formulation of the FS to uniquely satisfy the boundary conditions: in a slightly different framework from the one presented here, see [140, Theorem 29] and [40, proof of Lemma 2.4].

Theorem 2 (Friedrichs' system strong solution [96])**.** *Let $f \in [L^2(\Omega)]^m$, the strong solution $z \in [C^1(\overline{\Omega})]^m$ to Friedrichs' system*

$$\begin{cases} Az = f, & \text{in } \Omega, \\ (\mathcal{D}-\mathcal{M})z = 0, & \text{on } \partial\Omega. \end{cases} \tag{5.6}$$

is unique. Moreover, there exists a solution of the ultra-weak formulation

$$(z, \tilde{A}y)_{L^2} = (f, y)_{L^2}, \qquad \forall y \in [C^1(\overline{\Omega})]^m \ s.t.\ (\mathcal{D}+\mathcal{M}^t)y = 0. \tag{5.7}$$

Let $L = [L^2(\Omega)]^m$. We define the weak formulation on the graph space $V = \{z \in L : Az \in L\}$, which amounts to differentiability in the characteristics directions: $A \in \mathcal{L}(V, L)$ and $\tilde{A} \in \mathcal{L}(V', L)$. The boundary operator $\mathcal{D}$ is translated into the abstract operator $D \in \mathcal{L}(V, V')$:

$$\langle Dz, y\rangle_{V,V'} = (Az, y)_L - (z, \tilde{A}y)_L, \quad \forall z, y \in V. \tag{5.8}$$

When z is smooth, it can be seen as the integration by parts formula [140, 40]:

$$\langle Dz, y\rangle_{V,V'} = \langle \mathcal{D}z, y\rangle_{H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega)}, \quad \forall z \in H^1(\Omega),\ y \in H^1(\Omega). \tag{5.9}$$

A sufficient condition for well-posedness of the weak formulation is provided by the cone formalism [8, 76] that poses the existence of two linear subdomains (V_0, V_0^*) of V:

$$V_0 \text{ maximal in } C^+, \quad V_0^* \text{ maximal in } C^- \tag{5.10a}$$

$$V_0 = D(V_0^*)^\perp, \quad V_0^* = D(V_0)^\perp, \tag{5.10b}$$

such that $A : V_0 \to L$ and $\tilde{A} : V_0^* \to L$ are isomorphism, where $C^\pm = \{w \in V \,|\, \pm\langle Dw, w\rangle_{V,V'} \geq 0\}$.

Provided that $V_0 + V_0^* \subset V$ is closed [8], the conditions in (5.10) are equivalent to the existence of the boundary operator $M \in \mathcal{L}(V, V')$ that satisfies **admissible boundary conditions** analogue to the ones in (5.5):

$$\begin{aligned} &M \quad \text{is monotone}, && \text{(monotonicity property)} && (5.11a)\\ &\ker(D-M)+\ker(D+M)=V, && \text{(strict adjointness property)} && (5.11b)\end{aligned}$$

identifying $V_0 = \ker(D-M)$ and $V_0^* = \ker(D+M^*)$.

Theorem 3 (Friedrichs' System weak form [85, 84, 76, 40]). *Let us assume that the boundary operator $M \in \mathcal{L}(V, V')$ satisfies the monotonicity and strict adjointness properties (5.11). Let us define for $z, z^* \in V$ the bilinear forms*

$$a(z,y) = (Az,y)_L + \tfrac{1}{2}\langle (D-M)z, y\rangle_{V',V}, \quad \forall y \in V, \tag{5.12a}$$
$$a^*(z^*,y) = \left(\tilde{A}z^*,y\right)_L + \tfrac{1}{2}\langle (D+M^*)z^*, y\rangle_{V',V}, \quad \forall y \in V. \tag{5.12b}$$

Then, Friedrichs' operators $A : V_0 \to L$ and $\tilde{A} : V_0^* \to L$ are isomorphisms: for all $f \in L$ and $g \in V$ there exists unique $z, z^* \in V$ s.t.

$$a(z,y) = (f,y)_L + \langle (D-M)g, y\rangle_{V',V} \quad \forall y \in V, \tag{5.13a}$$
$$a^*(z^*,y) = (f,y)_L + \langle (D+M^*)g, y\rangle_{V',V} \quad \forall y \in V, \tag{5.13b}$$

that is

$$\begin{cases} Az = f, & \text{in } L,\\ (M-D)(z-g)=0 & \text{in } V', \end{cases} \qquad \begin{cases} \tilde{A}z^* = f, & \text{in } L,\\ (M^*+D)(z^*-g)=0 & \text{in } V'. \end{cases} \tag{5.14}$$

5.2.1 A unifying framework

The theory of Friedrichs' systems provides a unified framework to study different classes of PDEs [140]: first-order uniformly hyperbolic, second-order uniformly hyperbolic, elliptic and parabolic partial differential equations. Originally, Friedrichs' aim was to study equations of mixed type (hyperbolic, parabolic, elliptic) inside the same domain such as the Tricomi equation [96] (or more generally the Frankl equation [140]) inspired by models from compressible gas dynamics for which the domain is subdivided in a hyperbolic supersonic and an elliptic subsonic part.

Some examples of FS can be found in the literature:

$$(x_2\partial_1^2 \bullet + \partial_1^2 \bullet)u = 0,$$ Tricomi [96]

$$\begin{bmatrix} -\partial_1\bullet & \partial_2\bullet \\ -\partial_2\bullet & \partial_1\bullet \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = 0,$$ Cauchy-Riemann [96]

$$(A(x_2)\partial_1^2 + \partial_1^2)u = 0,$$ Frankl [140]

$$\begin{bmatrix} \mathbb{I}_3 & -\lambda^{-1}\mathbb{I}_3(\nabla\cdot\bullet) - \frac{(\nabla\bullet+(\nabla\bullet)^t)}{2} \\ -\frac{1}{2}\nabla\cdot(\bullet+\bullet^t) & \alpha\mathbb{I}_3 \end{bmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \mathbf{u} \end{pmatrix} = 0,$$ Compressible linear elasticity [85]

$$\begin{bmatrix} \mu\mathbb{I}_3 & \nabla\times\bullet \\ -\nabla\times\bullet & \sigma\mathbb{I}_3 \end{bmatrix} \begin{pmatrix} \mathbf{H} \\ \mathbf{E} \end{pmatrix} = 0,$$ Maxwell eq. in stationary regime [85]

$$(-\nabla\cdot(\kappa\nabla\bullet) + \boldsymbol{\beta}\cdot\nabla\bullet + \mu\bullet)\mathbf{u} = 0,$$ Diffusion advection reaction [85]

$$(A_0\partial_t \bullet + \Sigma_{i=1}^3 \tilde{A}_i\partial_i\bullet)\mathbf{V} = 0,$$ Linearized symmetric Euler [133, 235]

$$(a\gamma^0\partial_t \bullet + \gamma^1\partial_1 \bullet + \gamma^2\partial_2 \bullet + \gamma^3\partial_3 \bullet + B)\boldsymbol{\psi} = 0,$$ Dirac system [9]

$$\begin{bmatrix} -ai\gamma^0\partial_t \bullet - i\gamma^1\partial_1 \bullet - i\gamma^2\partial_2 \bullet - i\gamma^3\partial_3 \bullet + M & \mathbf{1}_4 \\ \mathbf{0}_4 & \partial_t^2 \bullet - \Delta \bullet + m^2\mathbb{I}_1 \end{bmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \phi \end{pmatrix} = \mathbf{f},$$ Dirac-Klein-Gordon system [9]

$$\begin{bmatrix} -\frac{i}{2\pi}(a\gamma^0\partial_t \bullet + \gamma^1\partial_1 \bullet + \gamma^2\partial_2 \bullet + \gamma^3\partial_3 \bullet + B) & \mathbf{1}_4 \\ \mathbf{0}_4 & -\partial_t^2 \bullet + \Delta\bullet \end{bmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \mathcal{A} \end{pmatrix} = \mathbf{f},$$ Maxwell-Dirac system [9]

$$\begin{bmatrix} -i\omega\mu\mathbb{I}_3 & \nabla\times\bullet \\ -\nabla\times\bullet & (-i\omega\varepsilon+\sigma)\mathbb{I}_3 \end{bmatrix} \begin{pmatrix} \mathbf{H} \\ \mathbf{E} \end{pmatrix} = 0,$$ Time-harmonic Maxwell [9]

$$\begin{bmatrix} \nu\mathbb{I}_3 & \nabla\times\bullet \\ \mu\boldsymbol{\beta}\times\bullet - \nabla\times\bullet & \sigma\mathbb{I}_3 \end{bmatrix} \begin{pmatrix} \mathbf{H} \\ \mathbf{E} \end{pmatrix} = \mathbf{f},$$ Magneto-hydrodynamics [84]

$$\begin{bmatrix} \nu^{-1}\mathbb{I}_3 & \mathbf{1}_3 & -\frac{(\nabla\bullet+(\nabla\bullet)^t)}{2} \\ \mathrm{tr}(\bullet) & d\mathbb{I}_1 & 0 \\ -\frac{1}{2}\nabla\cdot(\bullet+\bullet^t) & 0 & \boldsymbol{\beta}\cdot\nabla\bullet \end{bmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ p \\ \mathbf{u} \end{pmatrix} = \mathbf{f},$$ Incompressible linearized Navier-Stokes [86]

for the employed notation we refer to the respective reported references. A non-stationary version of (Maxwell eq. in stationary regime [85]) and (Diffusion advection reaction [85]) from [76] is omitted. The FS framework here presented easily extends to complex-valued systems as in (Dirac system [9]), (Dirac-Klein-Gordon system [9]), (Maxwell-Dirac system [9]) and (Time-harmonic Maxwell [9]) from [9]. We will consider only semi-linear PDEs but FS can be encountered as intermediate steps when solving quasi-linear PDEs: for example solving the compressible Euler equations of gas dynamics in entropy variables with the Newton method brings to the FS (Linearized symmetric Euler [133, 235]), as studied in [235].

One of the critical points of FS is the definition of the boundary conditions. Friedrichs's idea [96] was to impose boundary conditions through a matrix-valued boundary operator. Ern, Guermond and Caplain [87] revised the FS theory, without employing the trace of functions in the graph space as developed in [215, 216, 132, 140], but in terms of operators acting in abstract Hilbert spaces, as presented here.

The most common homogeneous boundary conditions (homogeneous Dirichlet, Neumann, Robin) for (Compressible linear elasticity [85]), (Maxwell eq. in stationary regime [85]) and also for (Diffusion advection reaction [85]) can be found in the literature [85, 84, 87]. For a choice of boundary conditions, and thus for a choice of spaces (V_0, V_0^*), there can be more than one definition of the boundary operator M [87, Remark 5.3]. A constructive methodology for defining the boundary operator $M \in \mathcal{L}(V, V')$ from specific boundary conditions can be found in [87] and it will be employed for the compressible linear elasticity test case in Section 5.2.1. Also, inhomogeneous boundary conditions can be imposed through the definition of traces of functions in graph spaces as in [140] or through a Petrov-Galerkin formulation as in [40]. In the following, we will present in detail three FS on which we will focus in the numerical test section.

Curl–curl problem: Maxwell equations in stationary regime

We will consider the Maxwell equations in stationary regime, also known as the curl-curl problem. Let $\mathbf{E} \in \mathbb{R}^d = \mathbb{R}^3$ be the electric field and $\mathbf{H} \in \mathbb{R}^3$ be the magnetic field. The curl–curl problem is defined as

$$\begin{cases} \mu \mathbf{H} + \nabla \times \mathbf{E} = \mathbf{r}, \\ \sigma \mathbf{E} - \nabla \times \mathbf{H} = \mathbf{g}, \end{cases}$$

with $\mu, \sigma > 0$, the permeability and permittivity constants. The FS is obtained by setting

$$A^0 = \begin{bmatrix} \mu \mathbb{I}_{d,d} & 0_{d,d} \\ 0_{d,d} & \sigma \mathbb{I}_{d,d} \end{bmatrix}, \; A^k = \begin{bmatrix} 0_{d,d} & \mathcal{R}^k \\ (\mathcal{R}^k)^T & 0_{d,d} \end{bmatrix}, \; f = \begin{pmatrix} \mathbf{r} \\ \mathbf{g} \end{pmatrix}$$

with $\mathcal{R}^k_{ij} = \varepsilon_{ikj}$ being the Levi-Civita tensor. The graph space is $V = H(\text{curl}, \Omega) \times H(\text{curl}, \Omega)$. The boundary operator is

$$\mathcal{D} = \sum_{k=1}^{d} n_k A^k = \begin{bmatrix} 0_{d,d} & \mathcal{T} \\ \mathcal{T}^T & 0_{d,d} \end{bmatrix}, \qquad \text{with } \mathcal{T}\xi := \mathbf{n} \times \boldsymbol{\xi}, \tag{5.16}$$

$$\langle D(\mathbf{H}, \mathbf{E}), (\mathbf{h}, \mathbf{e}) \rangle_{V',V} = (\mathbf{n} \times \mathbf{E}, \mathbf{e})_{L^2(\partial\Omega)} - (\mathbf{n} \times \mathbf{H}, \mathbf{h})_{L^2(\partial\Omega)}. \tag{5.17}$$

We impose homogeneous Dirichlet boundary conditions tangential to the electric field $(\mathbf{n} \times \mathbf{E})_{|\partial\Omega} = 0$ through

$$\mathcal{M} = \begin{bmatrix} 0_{d,d} & -\mathcal{T} \\ \mathcal{T}^T & 0_{d,d} \end{bmatrix}, \qquad \langle M(\mathbf{H}, \mathbf{E}), (\mathbf{h}, \mathbf{e}) \rangle_{V',V} = -(\mathbf{n} \times \mathbf{E}, \mathbf{e})_{L^2(\partial\Omega)} - (\mathbf{n} \times \mathbf{H}, \mathbf{h})_{L^2(\partial\Omega)}. \tag{5.18}$$

Compressible linear elasticity

We consider the parametric compressible linear elasticity system in $\mathbb{R}^d = \mathbb{R}^3$, where $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ is the stress tensor and $\mathbf{u} \in \mathbb{R}^d$ is the displacement vector. The system can be written as

$$\begin{pmatrix} \boldsymbol{\sigma} - \mu_1(\nabla\cdot\mathbf{u})\mathbb{I}_{3,3} - 2\mu_2\frac{(\nabla\mathbf{u}+(\nabla\mathbf{u})^t)}{2} \\ -\frac{1}{2}\nabla\cdot(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)+\mu_3\mathbf{u} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{r} \end{pmatrix}, \quad \forall x\in\Omega, \tag{5.19}$$

where $\mathbf{r}\in\mathbb{R}^3$, and $\mu_1,\ \mu_2 > 0$ are the Lamé constants. Rescaling the displacement $\mathbf{u}$ by $2\mu_2$, we obtain

$$\begin{pmatrix} \boldsymbol{\sigma} - \frac{\mu_1}{2\mu_2+3\mu_1}\mathrm{tr}(\boldsymbol{\sigma})\mathbb{I}_{3,3} - \frac{\left(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^T\right)}{2} \\ -\frac{1}{2}\nabla\cdot\left(\boldsymbol{\sigma}+\boldsymbol{\sigma}^T\right)+\frac{\mu_3}{2\mu_2}\boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{r} \end{pmatrix}, \quad \forall x\in\Omega. \tag{5.20}$$

In this case, we consider the graph space

$$V = H_{\boldsymbol{\sigma}}\times[H^1(\Omega)]^d, \quad H_{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma}\in[L^2(\Omega)]^{d\times d} \mid \nabla\cdot(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\in[L^2(\Omega)]^d\}. \tag{5.21}$$

If we reorder the coefficients of $\boldsymbol{\sigma}$ into a vector, we can define $\mathbf{z} = \begin{pmatrix}\boldsymbol{\sigma}\\ \mathbf{u}\end{pmatrix}$ and have

$$A^0 = \begin{bmatrix} \mathbb{I}_{d^2,d^2} - \frac{\mu_1}{2\mu_2+3\mu_1}\mathcal{Z} & 0_{d^2,d} \\ 0_{d,d^2} & \frac{\mu_3}{2\mu_2}\mathbb{I}_{d,d} \end{bmatrix}, \qquad A^k = \begin{bmatrix} 0_{d^2,d^2} & \mathcal{E}^k \\ (\mathcal{E}^k)^T & 0_{d,d} \end{bmatrix}, \qquad f = \begin{bmatrix} \mathbf{0}_{d^2} \\ \mathbf{r} \end{bmatrix}, \tag{5.22}$$

with $\mathcal{Z}_{[ij],[kl]} = \delta_{ij}\delta_{kl}$ and $\mathcal{E}^k_{[ij],l} = -\frac{1}{2}\left(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk}\right)$. This leads to the definition of the boundary operator

$$\mathcal{D} = \sum_{k=1}^{d} n_k A^k = \begin{bmatrix} 0_{d^2,d^2} & \mathcal{N} \\ \mathcal{N}^T & 0_{d,d} \end{bmatrix} \text{ with } \mathcal{N}\boldsymbol{\xi} := -\frac{1}{2}(\mathbf{n}\otimes\boldsymbol{\xi}+\boldsymbol{\xi}\otimes\mathbf{n}), \tag{5.23a}$$

$$\langle D(\boldsymbol{\sigma},\mathbf{u}),(\boldsymbol{\tau},\mathbf{v})\rangle_{V',V} = -\langle\tfrac{1}{2}(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\cdot\mathbf{n},\mathbf{v}\rangle_{-\frac{1}{2},\frac{1}{2}} - \langle\tfrac{1}{2}(\boldsymbol{\tau}+\boldsymbol{\tau}^t)\cdot\mathbf{n},\mathbf{u}\rangle_{-\frac{1}{2},\frac{1}{2}}. \tag{5.23b}$$

Mixed boundary conditions $\mathbf{u}_{|\Gamma_D} = 0$ and $(\boldsymbol{\sigma}\cdot\mathbf{n})_{|\Gamma_N} = 0$ can be applied through the following boundary operator on the Dirichlet boundary Γ_D and on the Neumann boundary Γ_N:

$$\begin{aligned} \langle M(\boldsymbol{\sigma},\mathbf{u}),(\boldsymbol{\tau},\mathbf{v})\rangle_{V',V} = &-\langle\tfrac{1}{2}(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\cdot\mathbf{n},\mathbf{v}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_D} + \langle\tfrac{1}{2}(\boldsymbol{\tau}+\boldsymbol{\tau}^t)\cdot\mathbf{n},\mathbf{u}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_D} \\ &+\langle\tfrac{1}{2}(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\cdot\mathbf{n},\mathbf{v}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_N} - \langle\tfrac{1}{2}(\boldsymbol{\tau}+\boldsymbol{\tau}^t)\cdot\mathbf{n},\mathbf{u}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_N}, \end{aligned} \tag{5.24}$$

the constructive procedure employed to define the boundary operator $M\in\mathcal{L}(V,V')$ is reported in the section 5.2.1.

Grad–div problem: advection–diffusion–reaction equations

Another example is the advection–diffusion–reaction equation

$$-\nabla\cdot(\kappa\nabla u)+\boldsymbol{\beta}\cdot\nabla u+\mu u=\mathbf{r}, \tag{5.25}$$

with $\kappa\in[L^\infty(\Omega)]^{d\times d}$ and $\boldsymbol{\beta}\in[W^{1,\infty}(\Omega)]^d$, under the hypothesis that $\kappa\in[L^\infty(\Omega)]^{d\times d}$ and $\mu-\nabla\cdot\boldsymbol{\beta}\in L^\infty(\Omega)$ are uniformly bounded from below to satisfy the positivity property (5.2b). Let us write the equation in the mixed form with $\boldsymbol{\sigma}=-\kappa\nabla u$ and $\mathbf{z}=\begin{pmatrix}\boldsymbol{\sigma}\\ u\end{pmatrix}$. Then, (5.25) can be rewritten as (5.6) with

$$A^0=\begin{bmatrix}\kappa^{-1} & 0_{d,1}\\ 0_{1,d} & \mu\end{bmatrix},\qquad A^k=\begin{bmatrix}0_{d,d} & \mathbf{e}_k\\ (\mathbf{e}_k)^T & \beta_k\end{bmatrix},\qquad \mathbf{f}=\begin{pmatrix}0\\ \mathbf{r}\end{pmatrix}. \tag{5.26}$$

Here, $0_{m,\ell}\in\mathbb{R}^{m\times\ell}$ is a matrix of zeros and $\mathbf{e}_k$ is the unitary vector with the k-th entry equal to 1. The graph space is $V=H(\mathrm{div},\Omega)\times H^2(\Omega)$. The boundary operator D becomes

$$D=\sum_{k=1}^{d}n_kA^k=\begin{bmatrix}0_{d,d} & \mathbf{n}\\ \mathbf{n}^t & \beta\cdot\mathbf{n}\end{bmatrix},\qquad \langle D(\boldsymbol{\sigma},u),(\boldsymbol{\tau},v)\rangle_{V',V}=\langle\boldsymbol{\sigma}\cdot\mathbf{n},v\rangle_{-\frac12,\frac12}-\langle\boldsymbol{\tau}\cdot\mathbf{n},u\rangle_{-\frac12,\frac12}. \tag{5.27}$$

Homogeneous Dirichlet boundary conditions $u_{|\partial\Omega}=0$ can be imposed with

$$\mathcal{M}=\begin{bmatrix}0_{d,d} & -\mathbf{n}\\ \mathbf{n}^t & 0\end{bmatrix}, \tag{5.28}$$

while Robin/Neumann boundary conditions of the type $\boldsymbol{\sigma}\cdot\mathbf{n}=\gamma u$ are imposed with

$$\mathcal{M}=\begin{bmatrix}0_{d,d} & \mathbf{n}\\ -\mathbf{n}^t & 2\gamma+\boldsymbol{\beta}\cdot\mathbf{n}\end{bmatrix}. \tag{5.29}$$

For our test case in Section 5.6, we will consider as advection field $\boldsymbol{\beta}:\Omega\to\mathbb{R}^d$ an incompressible velocity field from the solution of the 2d incompressible Navier-Stokes equations as described later. Similarly to the linear compressible elasticity mixed boundary conditions in Section 5.2.1, we want to impose $u_{|\Gamma_D}=\mathbf{g}\in[L^{\frac12}(\Gamma_D)]^d$ and $(\boldsymbol{\sigma}\cdot\mathbf{n})_{|\Gamma_N}=0$. This is possible with

$$\langle M(\boldsymbol{\sigma},u),(\boldsymbol{\tau},v)\rangle_{V',V}=\langle\boldsymbol{\sigma}\cdot\mathbf{n},v\rangle_{-\frac12,\frac12,\Gamma_D}+\langle\boldsymbol{\tau}\cdot\mathbf{n},u\rangle_{-\frac12,\frac12,\Gamma_D}-\langle\boldsymbol{\sigma}\cdot\mathbf{n},v\rangle_{-\frac12,\frac12,\Gamma_N}-\langle\boldsymbol{\tau}\cdot\mathbf{n},u\rangle_{-\frac12,\frac12,\Gamma_N}, \tag{5.30}$$

the proof is similar to the one reported in section 5.2.1.

Transformation into dissipative system

In some cases, the term $\mathcal{A}^0 = 0$ or property (5.2b) is not satisfied, but there is a way to recover the previous framework. We want to recover a dissipative [171] or accretive system [140]. For example, the linearized Euler equations in entropy variables [235] have $\mathcal{A}^0 = 0$.

The condition of uniform positive definiteness

$$\exists \mu_0, \quad \mathcal{A}^0 + (\mathcal{A}^0)^t - \mathcal{X} \geq 2\mu_0 \mathbb{I}_m \text{ a.e. in } \Omega, \tag{5.31}$$

is still valid if there exist $\boldsymbol{\xi} \in \mathbb{R}^d$, $\|\boldsymbol{\xi}\| = 1$ and $\beta \in \mathbb{R}$, $\beta > 0$ such that after the transformation

$$v(x) = e^{-\beta \xi \cdot x} z(x), \tag{5.32}$$

the resulting system

$$\sum_i A^i \partial_i v(x) + \beta \sum_{i=1}^d \xi_i A^i v(x) = e^{-\beta(\xi \cdot x)} f, \tag{5.33}$$

satisfies, with the newly found $A^0 = \beta \sum_{i=1}^d \xi_i A^i$,

$$\exists \mu_0, \quad A^0 + (A^0)^t - \mathcal{X} = 2\beta \sum_{i=1}^d \xi_i A^i - \mathcal{X} \geq 2\mu_0 \mathbb{I}_m \text{ a.e. in } \Omega. \tag{5.34}$$

In some cases, such ξ and β exist, for example if the symmetric matrix $\sum_{i=1}^d \xi_i A^i$ has at least one positive eigenvalue for some ξ for almost every $x \in \Omega$, then taking β sufficiently large is enough to satisfy the condition. It is also sufficient that $\sum_{i=1}^d \xi_i(x) A^i(x)$ has at least a positive eigenvalue for almost every $x \in \Omega$ where $\xi = \xi(x)$, see [140, Example 28].

Remark 12. A more general transformation is

$$v(x) = w(x) z(x), \tag{5.35}$$

so that the positive definiteness condition becomes

$$\exists \mu_0, \quad \mathcal{A}^0 + (\mathcal{A}^0)^t - \mathcal{X} = 2\sum_{i=1}^d \partial_i(-\log w)\mathcal{A}^i - \mathcal{X} \geq 2\mu_0 \mathbb{I}_m \text{ a.e. in } \Omega. \tag{5.36}$$

Constructive method to define boundary operators

We report a procedure to define a boundary operator $M \in \mathcal{L}(V, V')$ starting from some specified boundary conditions. We exploit Theorem 4.3, Lemma 4.4 and Corallary 4.1 from [87]. It can be seen that the most common Dirichlet, Neumann and Robin boundary conditions can be found for some FS [85, 84, 76], following this procedure.

Lemma 1 (Theorem 4.3, Lemma 4.4 and Corollary 4.1 from [87]). *Let us assume that* (V_0, V_0^*) *satisfy* (5.10) *and that* $V_0 + V_0^* \subset V$ *is closed. We denote with* $P : V \to V_0$ *and* $Q : V :\to V_0^*$ *the* *projectors onto the subspaces $V_0 \subset V$ and $V_0^* \subset V$ of the Hilbert space V, respectively. Then, the boundary operator $\mathcal{M} \in \mathcal{L}(V,V')$ defined as*

$$\begin{aligned}\langle \mathcal{M}u, v\rangle_{V',V} =& \langle \mathcal{D}Pu, Pv\rangle_{V',V} - \langle \mathcal{D}Qu, Qv\rangle_{V',V} + \\ & \langle \mathcal{D}(P+Q-PQ)u, v\rangle_{V',V} - \langle \mathcal{D}u, (P+Q-PQ)v\rangle_{V',V}\end{aligned} \tag{5.37}$$

is admissible and satisfies $V_0 = \ker(\mathcal{D}-\mathcal{M})$ and $V_0^* = \ker(\mathcal{D}+\mathcal{M}^*)$. In particular,

1. *If $V = V_0 + V_0^*$, then $\mathcal{M}$ is self-adjoint and*

$$\langle \mathcal{M}u, v\rangle_{V',V} = \langle \mathcal{D}Pu, Pv\rangle_{V',V} - \langle \mathcal{D}Qu, Qv\rangle_{V',V}\,. \tag{5.38}$$

2. *If $V_0 = V_0^*$, then $\mathcal{M}$ is skew-symmetric and*

$$\langle \mathcal{M}u, v\rangle_{V',V} = \langle \mathcal{D}Pu, v\rangle_{V',V} - \langle \mathcal{D}Pv, u\rangle_{V',V}\,. \tag{5.39}$$

We remark that, for fixed (V_0, V_0^*), admissible boundary operators $\mathcal{M} \in \mathcal{L}(V,V')$ that satisfy $V_0 = \ker(\mathcal{D}-\mathcal{M})$ and $V_0^* = \ker(\mathcal{D}+\mathcal{M}^*)$ are not unique. The boundary operator defined in Lemma 1 is just a possible explicit definition, in general.

As an exercise, we show how to find the definition of the operator $\mathcal{M}$ for our linear compressible elasticity FS, from Section 5.2.1. We want to impose the boundary conditions $\mathbf{u}_{|\Gamma_D} = 0$ and $(\boldsymbol{\sigma}\cdot\mathbf{n})_{|\Gamma_N} = 0$, so,

$$V_0 = V_0^* = \{(\mathbf{u},\boldsymbol{\sigma}) \in V \mid \mathbf{u}_{|\Gamma_D} = 0, \quad (\boldsymbol{\sigma}\cdot\mathbf{n})_{|\Gamma_N} = 0\} = H_{\boldsymbol{\sigma},\Gamma_N} \times [H^1_{\Gamma_D}(\Omega)]^d, \tag{5.40}$$

since we defined $V = H_{\boldsymbol{\sigma}} \times [H^1(\Omega)]^d$, with $H_{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma} \in [L^2(\Omega)]^{d\times d} \mid \nabla\cdot(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t) \in [L^2(\Omega)]^d\}$, the traces $\gamma_D : [H^1(\Omega)]^d \to [H^{\frac{1}{2}}(\Gamma_D)]^d$ and $\gamma_N : H^d_{\boldsymbol{\sigma}} \to [H^{-\frac{1}{2}}(\Gamma_N)]^d$ on Γ_D and Γ_N are well-defined. In particular,

$$H_{\boldsymbol{\sigma},\Gamma_N} = \{\boldsymbol{\sigma} \in H_{\boldsymbol{\sigma}} \mid \gamma_N(\boldsymbol{\sigma}) = (\boldsymbol{\sigma}\cdot\mathbf{n})_{|\Gamma_N} = 0\}, \qquad [H^1_{\Gamma_D}(\Omega)]^d = \{\mathbf{u} \in [H^1(\Omega)]^d \mid \gamma_D(\mathbf{u}) = \mathbf{u}_{|\Gamma_D} = 0\}. \tag{5.41}$$

Moreover, (V_0, V_0^*) satisfy the properties of cone formalism (5.10). Thus, we can use the definition (5.39) of Lemma 1:

$$\begin{aligned}\langle \mathcal{M}(\boldsymbol{\sigma},\mathbf{u}),(\boldsymbol{\tau},\mathbf{v})\rangle_{V',V} =& \langle \mathcal{D}P(\boldsymbol{\sigma},\mathbf{u}),(\boldsymbol{\tau},\mathbf{v})\rangle_{V',V} - \langle \mathcal{D}P(\boldsymbol{\tau},\mathbf{v}),(\boldsymbol{\sigma},\mathbf{u})\rangle_{V',V} \\ =& -\langle \tfrac{1}{2}(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\cdot\mathbf{n},\mathbf{v}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_D} + \langle \tfrac{1}{2}(\boldsymbol{\tau}+\boldsymbol{\tau}^t)\cdot\mathbf{n},\mathbf{u}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_D} + \\ & \langle \tfrac{1}{2}(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)\cdot\mathbf{n},\mathbf{v}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_N} - \langle \tfrac{1}{2}(\boldsymbol{\tau}+\boldsymbol{\tau}^t)\cdot\mathbf{n},\mathbf{u}\rangle_{-\frac{1}{2},\frac{1}{2},\Gamma_N},\end{aligned} \tag{5.42}$$

where $P : V = H_{\boldsymbol{\sigma}} \times [H^1(\Omega)]^d \to V_0 = H_{\boldsymbol{\sigma},\Gamma_N} \times [H^1_{\Gamma_D}(\Omega)]^d$ is the projector into the subspace V_0 of the Hilbert graph space V with scalar product:

$$((\boldsymbol{\sigma},\mathbf{u}),(\boldsymbol{\tau},\mathbf{v}))_V = (\mathbf{u},\mathbf{v})_{[L^d(\Omega)]^d} + (\boldsymbol{\sigma},\boldsymbol{\tau})_{[L^2(\Omega)]^{d\times d}} + (A(\boldsymbol{\sigma},\mathbf{u}), A(\boldsymbol{\tau},\mathbf{v})). \tag{5.43}$$

5.3 Discontinuous Galerkin discretization

In the literature, a few discretization approaches for FS are presented, e.g. finite volume method [235] or discontinuous Galerkin (DGM) method [85, 76, 40]. More recently, a hp-adaptive hybridizable DGM formulation was introduced in [49]. In this work, we perform a DGM discretization following the notation reported in [85, 76]. Consider a shape-regular tessellation $\mathcal{T}_h$ of the domain Ω and take a piecewise polynomial space V_h over $\mathcal{T}_h$, defined by $V_h = \{z \in V : z|_T \in \mathbb{P}^k_d(T), \forall T \in \mathcal{T}_h\}$, where k is the polynomial degree. We assume that there is a partition $P_\Omega = \{\Omega_i\}_{1\le i\le N_\Omega}$ of Ω into disjoint polyhedra such that the exact solution z belongs to $V^* = V \cap [H^1(P_\Omega)]^m$. We define the discrete bilinear form $\forall y_h \in V_h,\ z \in V^*$

$$a^{cf}_h(z,y_h) = \sum_{T\in\mathcal{T}_h} (Az,y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}^b_h} ((\mathcal{M}-\mathcal{D})z,y_h)_{L^2(F)} - \sum_{F\in\mathcal{F}^i_h} (\mathcal{D}_F[\![z]\!],\{\{y_h\}\})_{L^2(F)} \tag{5.44a}$$

$$= \sum_{T\in\mathcal{T}_h} (z,\tilde{A}y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}^b_h} ((\mathcal{M}+\mathcal{D})z,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}^i_h} (\mathcal{D}_F\{\{z\}\},[\![y_h]\!])_{L^2(F)}, \tag{5.44b}$$

where the first two terms are the piece-wise discontinuous discretization of the bilinear form (5.12a) and the last term penalizes the jump across neighboring cells and stabilizes the method. Here, $\mathcal{F}^b_h$ is the collection of the faces of the triangulation $\mathcal{T}_h$ belonging to the boundary of Ω_h, while $\mathcal{F}^i_h$ is the collection of internal faces. The jump and the average of a function on a face F shared by two elements T_1 and T_2 are defined as $[\![u]\!] = u|_{T_1} - u|_{T_2}$ and $\{\{u\}\} = \frac{1}{2}(u|_{T_1} + u|_{T_2})$, respectively. The boundary operator $\mathcal{D} : \partial\Omega \to \mathbb{R}^{m\times m}$ can be extended also on the internal faces $F \in \mathcal{F}^i_h$ as $\mathcal{D}_F = \sum_{k=1}^d n^F_k A^k$, where n^F is a normal to the face F and it is well-defined.

In order to obtain quasi-optimal error estimates, extra stabilization terms are needed. We additionally impose that $A^i \in [C^{0,\frac{1}{2}}(\overline{\Omega}_j)]^{m\times m}$, $\forall T \in \mathcal{T}_h$, $i = 1,\dots,d$, $\forall \Omega_j \in P_\Omega$. A possibility is given by the following stabilization term

$$s_h(z,y_h) = \sum_{F\in\mathcal{F}^b_h} (S^b_F z,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}^i_h} (S^i_h[\![z]\!],[\![y_h]\!])_{L^2(F)}, \tag{5.45}$$

where the operators S_h^i and S_F^b have to satisfy the following constraints for some $\alpha_j > 0$ for $j = 1, \dots, 5$:

$$S_F^b z = 0 \quad \forall F \in \mathcal{F}_h^b, \qquad S_F^i [\![z]\!] = 0 \quad \forall F \in \mathcal{F}_h^i, \qquad \text{with } z \text{ the exact solution,} \tag{5.46a}$$

$$S_F^b \text{ and } S_F^i \text{ are symmetric and nonnegative,} \tag{5.46b}$$

$$S_F^b \leq \alpha_1 \mathbb{I}_{m,m}, \qquad \alpha_2 |D_F| \leq S_F^i \leq \alpha_3 \mathbb{I}_{m,m}, \tag{5.46c}$$

$$|((M-D)y, z)_{L^2(F)}| \leq \alpha_4 ((S_F^b + M)y, y)_{L^2(F)}^{1/2} \|z\|_{L^2(F)}, \tag{5.46d}$$

$$|((M+D)y, z)_{L^2(F)}| \leq \alpha_5 ((S_F^b + M)z, z)_{L^2(F)}^{1/2} \|y\|_{L^2(F)}. \tag{5.46e}$$

Specific definitions of these operators for our test cases are presented in [85, 76], properly declined for our mixed boundary conditions in the compressible linear elasticity and advection–diffusion–reaction test cases, see Sections 5.2.1 and 5.2.1, respectively. Finally, we can define the bilinear form and the right-hand side

$$a_h(z, y_h) = a_h^{cf}(z, y_h) + s_h(z, y_h), \quad l_h(y_h) = \sum_{T \in \mathcal{T}_h} (f, y_h)_{L^2(T)} + \frac{1}{2} \sum_{F \in \mathcal{F}_h^b} ((M-D)g, y_h)_{L^2(F)} \tag{5.47}$$

that lead to the definition of the discrete problem.

Definition 5 (DG Friedrichs' System). *Given $f \in L$ and $g \in V_h$, the DGM approximation of the FS constitute in finding a $z_h \in V_h$ such that*

$$a_h(z_h, y_h) = l_h(y_h), \qquad \forall y_h \in V_h. \tag{5.48}$$

To prove the accuracy of the discrete problem, it is necessary to have the following conditions:

- Consistency, i.e., $a_h(z, y_h) = a(z, y_h)$ for $z \in V^*$;
- L^2-coercivity, i.e., $a_h(y_h, y_h) \geq \mu_0 \|y_h\|_L^2 + \frac{1}{2}|y_h|_M^2$, with $|y_h|_M^2 = \int_{\partial\Omega} y^t M y$;
- Inf-sup stability
$$|||z_h||| \lesssim \sup_{y_h \neq 0} \frac{a_h(z_h, y_h)}{|||y_h|||} \tag{5.49}$$
with $|||y|||^2 = ||y||_{L^2}^2 + |y|_M^2 + |y|_S^2 + \sum_{T \in \mathcal{T}_h} h_T \left\|A^k \partial_k y\right\|_{L^2(T)}^2$ and $|y|_S^2 = s_h(y, y)$;
- Boundedness $a_h(w, y_h) \lesssim |||w|||_* |||y_h|||$ with
$$|||y|||_*^2 = |||y|||^2 + \sum_{T \in \mathcal{T}_h} \left(h_T^{-1} ||y||_{L^2(T)}^2 + ||y||_{L^2(\partial T)}^2 \right). \tag{5.50}$$

Theorem 4 (Error estimate from [85, 76]). *Let $z \in V^*$ be the solution of the weak problem (5.13a) and $z \in V_h$ be the solution of the discrete DGM problem (5.48). Then, the consistency and inf-sup* *stability of the discrete system* (5.48) *imply*

$$|||z - z_h||| \lesssim \inf_{y_h \in V_h} |||z - y_h|||_*, \tag{5.51}$$

in particular, if $z \in [H^{k+1}(\Omega)]^m$ *the following convergence rate holds*

$$|||z - z_h||| \lesssim h^{k+\frac{1}{2}} \|z\|_{[H^{k+1}(\Omega)]^m}. \tag{5.52}$$

5.4 Projection-based model order reduction

The computation of discrete solutions of parametrized PDEs can require a not negligible computational time. In particular, in multi-query context, when many evaluations for different parameters are required, the computations may become unbearable. In this section, we introduce a reduced order model (ROM) for the FS in case of parameter dependent problems [126, 228], in order to drastically reduce the computational costs. To do so, we exploit two aspects of the above presented FS: the linearity of the problems and the affine dependence of the operators on the physical parameters.

As we have seen in Section 5.2.1, all the problems are depending on some parameters $\boldsymbol{\rho} \in \mathcal{P} \subset \mathbb{R}^{N_{\text{par}}}$ and the dependence is affine. This means that it is possible to find N_{aff} terms independent of the parameters for each form, such that they can be affinely combined with some parameter dependent functions to obtain the original operator, i.e.,

$$a_h(z, y_h; \boldsymbol{\rho}) = \sum_{\ell=1}^{N_{\text{aff}}} \theta_\ell^a(\boldsymbol{\rho}) a_{\ell,h}(z, y_h), \qquad l_h(y_h) = \sum_{\ell=1}^{N_{\text{aff}}} \theta_\ell^f(\boldsymbol{\rho}) l_{h,l}(y_h). \tag{5.53}$$

Then, we select a reduced space $V_n \subset V_h$ provided by a compression algorithm, e.g. SVD/POD/PCA [141, 153, 255] or Greedy algorithm [206, 207, 126, 64]. We suppose that the reduced dimension n is much smaller than the dimension N_h of the full-order model space V_h. We look as ansatz for a reduced solution $z_{\text{RB}} \in V_n$ a linear combination of the bases $\{\psi_j^{\text{RB}}\}_{j=1}^n$ of V_n, i.e.,

$$z_{\text{RB}} = \sum_{j=1}^{n} z_{\text{RB}}^j \psi_j^{\text{RB}}, \tag{5.54}$$

then, performing a standard Galerkin projection, we obtain the following RB problem.

Definition 6 (Reduced Basis Problem)**.** *Find $z_{RB} \in V_n$, given by the coefficients z_{RB}^j, such that*

$$\sum_{i=1}^{n} z_{RB}^j(\boldsymbol{\rho}) \sum_{\ell=1}^{N_{aff}} \theta_\ell^a(\boldsymbol{\rho}) a_{\ell,h}(\psi_j^{RB}, \psi_i^{RB}) = \sum_{\ell=1}^{N_{aff}} \theta_\ell^f(\boldsymbol{\rho})(f_\ell, \psi_i^{RB}), \qquad \textit{for all } i = 1, \dots, n. \tag{5.55}$$

The obtained problem scales depend on the dimension n and N_{aff} in its assembly and only on n in its solution, and it is completely independent of N_h. To obtain computational advantages for the parametric problem, we split the tasks into an expensive *offline phase* and a cheap *online phase*. In the *offline phase*, we find the reduced space V_n and we assemble the reduced matrices and right-hand sides

$$A_\ell := \{a_{\ell,h}(\psi_j^{\text{RB}}, \psi_i^{\text{RB}})\}_{i,j}, \qquad b_\ell := \{(f_\ell, \psi_i^{\text{RB}})\}_i. \tag{5.56}$$

In the *online phase*, we can simply evaluate the coefficients $\theta_\ell^a(\boldsymbol{\rho})$ and $\theta_\ell^f(\boldsymbol{\rho})$ and obtain the reduced linear system

$$A(\boldsymbol{\rho})z_{\text{RB}} = b(\boldsymbol{\rho}), \qquad \text{with } A(\boldsymbol{\rho}) := \sum_\ell \theta_\ell^a(\boldsymbol{\rho})A_\ell \text{ and } b(\boldsymbol{\rho}) := \sum_\ell \theta_\ell^f(\boldsymbol{\rho})b_\ell. \tag{5.57}$$

This gives a great speedup in computational times.

5.4.1 Reduced basis a posteriori error estimate

We derive two error estimators for the energy norm and the L^2 norm of the reduced basis error $e_h = z_h - z_{RB} \in V_h$ following the procedure in [126]. Exploiting the equality in (5.44), we obtain the following lower bound

$$a_h(e_h,e_h) = a_h^{cf}(e_h,e_h) + s_h(e_h,e_h) \tag{5.58a}$$

$$= \sum_{T\in\mathcal{T}_h} (Ae_h,e_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}^b} ((\mathcal{M}-\mathcal{D})e_h,e_h)_{L^2(F)} - \sum_{F\in\mathcal{F}_h^i} (\mathcal{D}_F[\![e_h]\!],\{\{e_h\}\})_{L^2(F)} + \tag{5.58b}$$

$$\sum_{F\in\mathcal{F}_h^b} (S_F^b e_h,e_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_h^i} (S_h^i[\![e_h]\!],[\![e_h]\!])_{L^2(F)} \tag{5.58c}$$

$$= \sum_{T\in\mathcal{T}_h} ((A^0 - \tfrac{1}{2}\mathcal{X})e_h,e_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}^b} (\mathcal{M}e_h,e_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_h^b} (S_F^b e_h,e_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_h^i} (S_h^i[\![e_h]\!],[\![e_h]\!])_{L^2(F)} \tag{5.58d}$$

$$\geq \mu_0\|e_h\|_L^2 + \tfrac{1}{2}|e_h|_M^2 + |e_h|_S^2, \tag{5.58e}$$

where we have defined $|\cdot|_M^2 = \frac{1}{2}\sum_{F\in\mathcal{F}^b}(\mathcal{M}\cdot,\cdot)_{L^2(F)}$ and $|\cdot|_S^2 = s_h(\cdot,\cdot)$. We define the R-norm

$$||y_h||_R^2 = \mu_0||y_h||_L^2 + \tfrac{1}{2}|y_h|_M^2 + |y_h|_S^2, \quad \forall y_h \in V_h, \tag{5.59}$$

that may depend on ρ only through μ_0 and is generated by the scalar product

$$\langle u_h,v_h\rangle_R = \mu_0\sum_{T\in\mathcal{T}_h}(u_h,v_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}^b}(\mathcal{M}^{sym}u_h,v_h)_{L^2(F)} + \sum_{F^b}(S^b u_h,v_h) + \sum_{F^i}(S^i u_h,v_h). \tag{5.60}$$

The boundary operators we will employ in our benchmarks are all skew-symmetric so $\mathcal{M}^{sym} = \frac{\mathcal{M}+\mathcal{M}^t}{2}$ is the null matrix and $|e_h|_M = 0$. Now, we can proceed to provide an *a posteriori* error estimate for the R-norm and energy norm. Hence, let us define $r_{RB}(y_h) = l_h(y_h;\rho) - a_h(z_{RB},y_h;\rho)$ and its R and

L-Riesz representations as $\hat{r}_R$ and $\hat{r}_L$ such that

$$r_{RB}(u_h) = \langle \hat{r}_R, u_h \rangle_R, \quad X_R \hat{\mathbf{r}}_R = \mathbf{L}_h - A_h \mathbf{z}_{RB}, \quad r_{RB}(u_h) = \langle \hat{r}_L, u_h \rangle_L, \quad X_L \hat{\mathbf{r}}_L = \mathbf{L}_h - A_h \mathbf{z}_{RB}, \tag{5.61}$$

where X_R and X_L are the R-norm and L-norm mass matrices, and L_h, A_h and $\mathbf{z}_{RB}$ are the representations of $l_h(\cdot;\rho)$, $a_h(\cdot,\cdot;\rho)$ and z_{RB} in the DG basis of V_h. The $\hat{r}_L$ representation can be computed cheaply when the parametric model is affinely decomposable with respect to the parameters, while $\hat{r}_R$ requires the inversion of a possibly parametric dependent matrix X_R.

Now, consider the energy norm of the error $||e_h||^2_{nrg} = a_h(e_h, e_h)$ and the coercivity constant $||e_h||^2_{nrg} \geq \mu_0 ||e_h||^2_L$ derived in (5.58e), we have the following *a posteriori* error estimates

$$\frac{||e_h||_{nrg}}{||z_h||_{nrg}} \leq \frac{||\hat{r}_R||_R}{||z_h||_{nrg}}, \qquad \frac{||e_h||_R}{||z_h||_R} \leq \frac{||\hat{r}_R||_R}{||z_h||_R}, \qquad \frac{||e_h||_{nrg}}{||z_h||_{nrg}} \leq \frac{||\hat{r}_L||_L}{\sqrt{\mu_0}||z_h||_{nrg}}, \qquad \frac{||e_h||_L}{||z_h||_L} \leq \frac{||\hat{r}_L||_L}{\mu_0||z_h||_L}, \tag{5.62}$$

namely, the relative energy error with the corresponding *a posteriori* R-norm energy estimate, the relative R-norm error with the corresponding *a posteriori* R-norm estimate, the relative energy error with the corresponding *a posteriori* L-norm energy estimate, and the relative L-norm error with the corresponding *a posteriori* L-norm estimate.

5.4.2 Optimally stable error estimates for the ultraweak Petrov-Galerkin formulation

In this section, we show that Friedrichs' systems are a desirable unifying formulation to consider when performing model order reduction also due to the possibility to achieve an optimally stable formulation. This can further simplify the error estimator analysis reaching the equality between the error and the residual norm. This is not the first case in which optimally stable formulations are introduced also at the reduced level, see [28, 116, 124]. In the following, we describe how to achieve this ultraweak formulation and we delineate the path one should follow to use such formulation. Nevertheless, we will not use this formulation in our numerical tests and we leave the implementation to future studies.

We introduce the following Discontinuous Petrov-Galerkin (DPG) formulation from [40]. To do so, we first define $V(\mathcal{T}_h)$ the broken graph space with norm $||\bullet||^2_{V(\mathcal{T}_h)} = ||\bullet||^2_L + \sum_{T\in\mathcal{T}_h} ||A\bullet||^2_{L^2(T)}$ and $\tilde{V} = V/Q(\Omega)$ the quotient of the graph space V with

$$\begin{aligned} Q(\Omega) &= \left\{ z \in V \,\middle|\, \sum_{T\in\mathcal{T}_h} \langle Dz, y \rangle_{\tilde{V}(T),V(T)} + \tfrac{1}{2}\langle (M-D)z, y\rangle_{\tilde{V}(\Omega),V(\Omega)} = 0, \quad \forall y \in V(\mathcal{T}_h) \right\} \\ &= \left\{ z \in V \,\middle|\, a(z,y) = 0, \quad \forall y \in V(\mathcal{T}_h) \right\}. \end{aligned} \tag{5.63}$$

The DPG formulation reads: find $(z,q) \in L \times \tilde{V}$ such that, for all $y \in V(\mathcal{T}_h)$,

$$\sum_{T\in\mathcal{T}_h}(z,\tilde{A}y)_{L^2(T)} + \sum_{T\in\mathcal{T}_h}\langle Dq,y\rangle_{\tilde{V}(T),V(T)} + \tfrac{1}{2}\langle (M-D)q,y\rangle_{\tilde{V},V} = \sum_{T\in\mathcal{T}_h}(f,v)_{\tilde{V}(T),V(T)} + \tfrac{1}{2}\langle (M-D)g,y\rangle_{\tilde{V},V}. \tag{5.64}$$

The introduction of the hybrid face variables $q \in \tilde{V}$ is necessary since $z \in L$ does not satify (5.8). In practice, assuming that the traces of $y \in V(\mathcal{T}_h)$ are well-defined and belong to a space $X(\mathcal{F}_{i,b})$, we can formulate (5.64) as follows: find $(z,q) \in L \times X(\mathcal{F}_{i,b})$ such that, for all $y \in V(\mathcal{T}_h)$,

$$\sum_{T\in\mathcal{T}_h}(z,\tilde{A}y)_{L^2(T)} + \sum_{F\in\mathcal{F}_i}(\mathcal{D}q,[\![y]\!])_{X(F)} + \tfrac{1}{2}\sum_{F\in\mathcal{F}_b}((\mathcal{M}-\mathcal{D})q,y)_{X(F)} = \sum_{T\in\mathcal{T}_h}(f,v)_{\tilde{V}(T),V(T)} + \tfrac{1}{2}\sum_{F\in\mathcal{F}_b}((\mathcal{M}-\mathcal{D})g,y)_{X(F)}, \tag{5.65}$$

where $X(\mathcal{F}_{i,b})$ is, for example, $[H^{-\frac{1}{2}}(\mathcal{F}_{i,b})]^d \times [H^{\frac{1}{2}}(\mathcal{F}_{i,b})]^d$ for compressible linear elasticity, $H^{-\frac{1}{2}}(\mathcal{F}_{i,b}) \times H^{\frac{1}{2}}(\mathcal{F}_{i,b})$ for the scalar advection–diffusion–reaction and $L^2_T(\mathcal{F}_{i,b}) \times L^2_T(\mathcal{F}_{i,b})$ for the Maxwell equations in stationary regime, with $L^2_T(\mathcal{F}_{i,b})$ being the space of fields in $H(\mathrm{curl},\mathcal{T}_h)$ whose tangential component belongs to $[L^2(\mathcal{F}_{i,b})]^3$.

The problem (5.64) above is well-posed and consistent [40, Lemma 2.4] with the previous formulation in (5.13a). We consider the optimal norms,

$$\|(z,q)\|^2_{\mathcal{U}} = \sum_{T\in\mathcal{T}_h}\|z\|^2_{L(T)} + \|q\|^2_{\tilde{V}}, \qquad \|y\|^2_{\mathcal{Y}} = \sum_{T\in\mathcal{T}_h}\|\tilde{A}y\|^2_{L(T)} + \|[\![y]\!]\|^2_{\partial\Omega_h}, \quad \text{with} \quad \|[\![y]\!]\|_{\partial\Omega_h} = \sup_{q\in\tilde{V}}\frac{a(q,y)}{\|q\|_{\tilde{V}}}, \tag{5.66}$$

or formally, considering (5.65)

$$\|(z,q)\|^2_{\mathcal{U}} = \sum_{T\in\mathcal{T}_h}\|z\|^2_{L(T)} + \sum_{F\in\mathcal{F}_{i,b}}\|q\|^2_{X(F)}, \qquad \|y\|^2_{\mathcal{Y}} = \sum_{T\in\mathcal{T}_h}\|\tilde{A}y\|^2_{L(T)} + \sum_{F\in\mathcal{F}_i}\|\mathcal{D}_F[\![y]\!]\|^2_{X(F)} + \sum_{F\in\mathcal{F}_b}\|(\mathcal{M}^t-\mathcal{D})y\|^2_{X(F)}. \tag{5.67}$$

With these optimal norms for the trial and test spaces, we have the following result [39, Theorem 2.6].

Theorem 5 (Optimally stable formulation)**.** *The bilinear form* $b : (L\times\tilde{V}, \|\bullet\|_{\mathcal{U}}) \to (V(\mathcal{T}_h), \|\bullet\|_{\mathcal{Y}})$ *defined as*

$$b(u,y) = \sum_{T\in\mathcal{T}_h}(z,\tilde{A}y)_{L^2(T)} + \sum_{T\in\mathcal{T}_h}\langle Dq,y\rangle_{\tilde{V}(T),V(T)} + \tfrac{1}{2}\langle (M-D)q,y\rangle_{\tilde{V},V} \tag{5.68}$$

with $u=(z,q)$*, is an isometry between* $L\times\tilde{V}$ *and* $V'(\mathcal{T}_h)$*: we have that* $\gamma=\beta=\beta^*=1$*, where*

$$\gamma := \sup_{u\in\mathcal{U}}\sup_{y\in\mathcal{Y}}\frac{b(u,y)}{\|u\|_{\mathcal{U}}\|y\|_{\mathcal{Y}}}, \quad \beta := \inf_{u\in\mathcal{U}}\sup_{y\in\mathcal{Y}}\frac{b(u,y)}{\|u\|_{\mathcal{U}}\|y\|_{\mathcal{Y}}}, \quad \beta^* := \inf_{y\in\mathcal{Y}}\sup_{u\in\mathcal{U}}\frac{b(u,y)}{\|u\|_{\mathcal{U}}\|y\|_{\mathcal{Y}}} \tag{5.69}$$

with $\mathcal{U} = (L\times\tilde{V}, \|\bullet\|_{\mathcal{U}})$.

This property is inherited at the discrete level as long as fixed $U_h^{N_h} \subset \mathcal{U}$ a discretization of the trial space with $\dim Z_h = N_h$, the discrete test space $Y_h^{N_h} \subset V(\mathcal{T}_h)$ is the set of supremizers

$$Y_h^{N_h} = \text{span}\left\{ y_{u_h} \in V(\mathcal{T}_h) \,\middle|\, y = \underset{y\in V(\mathcal{T}_h)}{\text{argmax}} \frac{b(u_h, y)}{\|y\|_{\mathcal{Y}}}, \quad u_h \in Z_h^{N_h} \right\} \tag{5.70}$$

and $\dim U_h = \dim Y_h$, see [39, Lemma 2.8]. In particular, for every $u_h = (z_h, q_h) \in U_h^{N_h}$, we have the optimal *a posteriori* error estimate

$$\|u - u_h\|_{\mathcal{U}} = \sup_{y\in V(\mathcal{T}_h)} \frac{b(u - u_h, y)}{\|y\|_{\mathcal{Y}}} = \|r_h(u_h)\|_{(V(\mathcal{T}_h))'}, \quad \langle r_h(u_h), v\rangle_{(V(\mathcal{T}_h))', V(\mathcal{T}_h)} = \langle f, v\rangle_{(V(\mathcal{T}_h))', V(\mathcal{T}_h)} - b(u_h, v). \tag{5.71}$$

The same reasoning can be iterated another time to perform model order reduction with the choice $V_n = \{\psi_j^{\text{RB}}\}_{j=1}^n \subset U_h^{N_h} \subset \mathcal{U}$, and

$$Y^{RB} = \text{span}\left\{ y_{u_{RB}} \in V(\mathcal{T}_h) \,\middle|\, y = \underset{y\in V(\mathcal{T}_h)}{\text{argmax}} \frac{b(u_{RB}, y)}{\|y\|_{\mathcal{Y}}}, \quad u_{RB} \in V_n \right\}, \tag{5.72}$$

such that for $u_{RB} \in V_n$,

$$\|u - u_{RB}\|_{\mathcal{U}} = \|r_{RB}(u_{RB})\|_{(V(\mathcal{T}_h))'}, \quad \langle r_{RB}(u_{RB}), v\rangle_{(V(\mathcal{T}_h))', V(\mathcal{T}_h)} = \langle f, v\rangle_{(V(\mathcal{T}_h))', V(\mathcal{T}_h)} - b(u_{RB}, v). \tag{5.73}$$

The main difficulty is the evaluation of the trial spaces $Y_h^{N_h}$ and Y^{RB} since the bilinear form b may depend on the parameters $\boldsymbol{\rho}$. If the parameters affect only the source terms, the boundary conditions or the initial conditions for time-dependent FS, this problem is avoided. The evaluation of $Y_h^{N_h}$ can be performed locally for each element $T \in \mathcal{T}_h$, differently from Y^{RB}. An example of the evaluation of the basis of $Y_h^{N_h}$ is presented in [39, Equations 24, 25] for linear scalar hyperbolic equations that can be interpreted as FS.

5.5 Domain decomposable Discontinuous Galerkin ROMs

Extreme-scale parametric models are unfeasible to reduce with standard approaches due to the high computational costs of the offline stage. Parametric multi-physics simulations, such as fluid-structure interaction problems, are reduced inefficiently with a global reduced basis, depending on the complexity of the interactions between the physical models considered and the parametric dependency. In some cases, only a part of a decomposable system is reducible with a ROM, thus a possible solution is to implement a ROM-FOM coupling through an interface. In the presence of moving shocks [14] affected by the parametrization, one may want to isolate these difficult features to approximate and apply different dimension reduction methodologies depending on the subdomain. These are the main reasons to develop domain decomposable or partitioned ROMs (DD-ROMs).

Some approaches from the literature are the reduced basis element methods [172, 173], the static condensation reduced basis element method [134, 82, 134], non-intrusive methods based on local regressions in each subdomain [275, 274] and hyper-reduced ROMs [168]. In this last case, local approximations are useful because the local reduced dimensions are smaller and therefore more accurate local regressions can be designed to perform non-intrusive surrogate modelling. Little has been developed for the DGM method, even though its formulation imposes naturally flux and solution interface penalties at the internal boundaries of the subdomains, in perspective of performing model order reduction. In our case, the linear systems associated with the parametric models are algebraically partitioned in disjoint subdomains coupled with the standard penalties from the weak DGM formulations, without the need to devise additional operators to perform the coupling as long as the interfaces' cuts fall on the cell boundaries.

Another less explored feature of DD-ROMs is the possibility to repartition the computational domain while keeping the data structures relative to each subdomain local in memory, with the aim of obtaining more efficient or accurate ROMs. In fact, one additional reason to subdivide the computational domain is to partition the solution manifold into local solution manifolds that have a faster decay of the Kolmogorov n-width. The repartition of the computational domain can be performed with *ad hoc* domain decomposition strategies. To our knowledge, the only case found in the literature is introduced in [274], where the degrees of freedom are split in each subdomain minimizing the communication and activity between them and balancing the computational load across them. Relevant is the choice of weights to assign to each degree of freedom: uniform weights, nodal values of Reynolds stresses for the turbulent Navier-Stokes equations or the largest singular value of the discarded local POD modes. In particular, the last option results in a balance of energy in L^2 norm retained in each subdomain. We explore a different approach.

It must be remarked that in any case, fixed the value of the reconstruction error of the training dataset in Frobenious norm $\|\cdot\|_F$, there must be at least a local reduced basis dimension greater or equal to the global reduced basis space dimension. In fact, for the Eckhart-Young theorem, if $X \in \mathbb{R}^{d\times n}$ is the snapshots matrix ordered by columns, we have that the projection into the first k modes $\{v_k\}_{i=1}^k$ achieves the best approximation error in the Frobenious norm in the space of matrices of rank k:

$$P_k = \underset{P\in\mathbb{R}^{m\times n} s.t.\ r(P)=k}{\operatorname{argmin}} \|X - PX\|_F, \quad P_k = \sum_{i=1}^{k} v_i \otimes v_i, \tag{5.74}$$

where $r(\cdot)$ is the matrix rank. So, in general, it is not possible to achieve a better training approximation error in the Frobenious norm $\|\cdot\|_F$ employing a number of local reduced basis smaller than what would be needed to achieve the same accuracy with a global reduced basis. So, differently from [274], instead of balancing the local reduced basis dimension among subdomains, we repartition the computational domain in regions whose restricted solution manifold is approximable by linear subspaces of higher or lower local dimension. Anyway, for truly decomposable systems we expect that the reconstruction error on the test set is lower when considering local reduced basis instead of global ones, as will be shown for the Maxwell equation in stationary regime test case with discontinuous piecewise constant parameters, see Figure 5.7.

5.5.1 Implementation of Domain Decomposable ROMs

Let us assume that the full-order model is implemented in parallel with memory distributed parallelism with $K > 1$ cores, i.e., each i-th core owns locally the data structures relevant only to its assigned subdomain Ω_i of the whole computational domain $\cup_{i=1}^{K}\Omega_i = \Omega_h \subset \mathbb{R}^d$, for $i = 1,\dots,K$. We will employ the `deal.II` library [12] to discretize the FS with the DGM method, assemble the associated linear systems and solve them in parallel [18]. In particular, we employ `p4est` [42] to decompose the computational domain, `PETSc` [17] to assemble the linear system and solve it at the full-order level and `petsc4py` [66] to assemble and solve the reduced order system. At the offline and online stages, the computations are performed in a distributed memory setting in which each core assembles its own affine decomposition so that the evaluation of the reduced basis and of the projected local operators is always performed in parallel.

The weak formulation (5.48) is easily decomposable thanks to the additive properties of the integrals. We recall the definition of the weak formulation $\forall y_h \in V_h,\ z_h \in V^*$

$$a_h^{cf}(z_h,y_h)+s_h(z_h,y_h) = \sum_{T\in\mathcal{T}_h}(z_h,\tilde{A}y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}_h^b}((\mathcal{M}+\mathcal{D})z_h,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_h^i}(\mathcal{D}_F\{\{z_h\}\},[\![y_h]\!])_{L^2(F)} + \\ \sum_{F\in\mathcal{F}_h^b}(S_F^b z_h,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_h^i}(S_h^i[\![z_h]\!],[\![y_h]\!])_{L^2(F)}, \tag{5.75}$$

and we decompose it into the K subdomains as

$$a_h^{cf}(z_h,y_h)+s_h(z_h,y_h) = \sum_{i=1}^{K}\left(\sum_{T\in\mathcal{T}_{h,i}}(z_h,\tilde{A}y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}_{h,i}^b}((\mathcal{M}+\mathcal{D})z_h,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_{h,i}^i}(\mathcal{D}_F\{\{z_h\}\},[\![y_h]\!])_{L^2(F)} + \right. \\ \left. \sum_{F\in\mathcal{F}_{h,i}^b}(S_F^b z_h,y_h)_{L^2(F)} + \sum_{F\in\mathcal{F}_{h,i}^i}(S_h^i[\![z_h]\!],[\![y_h]\!])_{L^2(F)}\right) + \sum_{\substack{i=1\\ j=i}}^{K}\left(\sum_{F\in\mathcal{F}_{h,i,j}^i}(\mathcal{D}_F\{\{z_h\}\},[\![y_h]\!])_{L^2(F)} + \right. \\ \left. \sum_{F\in\mathcal{F}_{h,i,j}^i}(S_h^i[\![z_h]\!],[\![y_h]\!])_{L^2(F)}\right), \tag{5.76}$$

$$l_h(y_h) = \sum_{T\in\mathcal{T}_h}(f,y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}_h^b}((M-D)g,y_h)_{L^2(F)} = \sum_{i=1}^{K}\left(\sum_{T\in\mathcal{T}_{h,i}}(f,y_h)_{L^2(T)} + \frac{1}{2}\sum_{F\in\mathcal{F}_{h,i}^b}((M-D)g,y_h)_{L^2(F)}\right), \tag{5.77}$$

where we have defined the internal subsets $\mathcal{T}_{h,i} = \mathcal{T}_h \cap \Omega_i$, $\mathcal{F}^i_{h,i} = \mathcal{F}^i_h \cap \mathring{\Omega}_i$ and $\mathcal{F}^b_{h,i} = \mathcal{F}^b_h \cap \mathring{\Omega}_i$, $\forall i = 1,\dots,K$ and the interfaces subsets $\mathcal{F}^i_{h,i,j} = \mathcal{F}^i_h \cap \overline{\Omega}_i \cap \overline{\Omega}_j$ and $\mathcal{F}^b_{h,i,j} = \mathcal{F}^b_h \cap \overline{\Omega}_i \cap \overline{\Omega}_j$, $\forall i = 1,\dots,K$. We remark that the computational domain is always decomposed such that the cuts of the subdomains $\{\partial\Omega_i\}_{i=1}^K$ fall on the interfaces of the triangulation $\mathcal{F}^i_h \cup \mathcal{F}^b_h$.

We define the bilinear and linear operators in V_h^*,

$$\begin{aligned} \mathcal{A}_{ii} = &\sum_{T\in\mathcal{T}_{h,i}} (\bullet, \tilde{A}\bullet)_{L^2(T)} + \tfrac{1}{2}\sum_{F\in\mathcal{F}^b_{h,i}} ((\mathcal{M}+\mathcal{D})\bullet,\bullet)_{L^2(F)} + \sum_{F\in\mathcal{F}^i_{h,i}} (\mathcal{D}_F\{\{\bullet\}\}, [\![\bullet]\!])_{L^2(F)} + \\ &\sum_{F\in\mathcal{F}^b_{h,i}} (S^b_F\bullet,\bullet)_{L^2(F)} + \sum_{F\in\mathcal{F}^i_{h,i}} (S^i_h[\![\bullet]\!],[\![\bullet]\!])_{L^2(F)}, \quad \forall i = 1,\dots,K, \end{aligned} \tag{5.78}$$

$$\mathcal{A}_{ij} = \mathcal{A}_{ji} = \sum_{F\in\mathcal{F}^i_{h,i,j}} (\mathcal{D}_F\{\{\bullet\}\}, [\![\bullet]\!])_{L^2(F)} + \sum_{F\in\mathcal{F}^i_{h,i,j}} (S^i_h[\![\bullet]\!],[\![\bullet]\!])_{L^2(F)}, \quad \forall j,i = 1,\dots,K,\ i\neq j, \tag{5.79}$$

$$\mathcal{F}_i = \sum_{T\in\mathcal{T}_{h,i}} (f,\bullet)_{L^2(T)} + \tfrac{1}{2}\sum_{F\in\mathcal{F}^b_{h,i}} ((M-D)g,\bullet)_{L^2(F)}, \quad \forall i = 1,\dots,K, \tag{5.80}$$

and their matrix representation in the discontinuous Galerkin basis of V_h,

$$(\mathcal{A}_{ii})|_{V_h} = A_{ii}, \qquad \mathcal{F}_i|_{V_h} = F_i, \quad \forall i = 1,\dots,K, \qquad (\mathcal{A}_{ij})\big|_{V_h} = (\mathcal{A}_{ji})\big|_{V_h} = A_{ij} = A_{ji}, \quad \forall j,i = 1,\dots,K,\ i\neq j, \tag{5.81}$$

and in the local reduced basis $V_i = \{\psi^{\text{RB}}_{j,i}\}_{j=1}^n \subset V_h(\Omega_i)$, $i = 1,\dots,K$,

$$(\mathcal{A}_{ii})|_{V_{RB}} = B_{ii}, \qquad \mathcal{F}_i|_{V_{RB}} = L_i, \quad \forall i = 1,\dots,K, \qquad (\mathcal{A}_{ij})\big|_{V_{RB}} = (\mathcal{A}_{ji})\big|_{V_{RB}} = B_{ij} = B_{ji}, \quad \forall j,i = 1,\dots,K,\ i\neq j. \tag{5.82}$$

As anticipated, in our test cases the subdomains interface penalties are naturally included inside $\{\mathcal{A}_{ij}\}_{i,j=1,\dots,K}$. In practice, additional penalty terms could be implemented:

$$\mathcal{S}_{ij} = \sum_{F\in\mathcal{F}^i_{h,i,j}} (S[\![\bullet]\!],[\![\bullet]\!])_{L^2(F)}, \qquad \mathcal{S}_{ij}\big|_{V_h} = S_{ij} \qquad (\mathcal{A}_{ij}+\mathcal{S}_{ij})\big|_{V_{RB}} = B_{ij}, \quad \forall j,i = 1,\dots,K,\ i\neq j. \tag{5.83}$$

A matrix representation of the projection of the full-order block matrix $(A_{ij})_{i,j=1}^K \in \mathbb{R}^{d\times d}$ into the reduced order block matrix $(B_{ij})_{i,j=1}^K \in \mathbb{R}^{Kn\times Kn}$ is shown in Figure 5.1 for $K = 4$. We remark that, differently from continuous Galerkin formulations, the DGM penalization on jumps across the interfaces is already enough to couple the subdomains and there is no need for further stabilization, as shown in Figure 5.1. Nonetheless, additional interface penalties terms can be easily introduced, taking also into account DGM numerical fluxes. The reduced dimension is the number of subdomains K times the local reduced basis dimensions $\{n_i\}_{i=1}^K$, here supposed equal $n = n_i$, $i = 1,\dots,K$, but in general can be different.

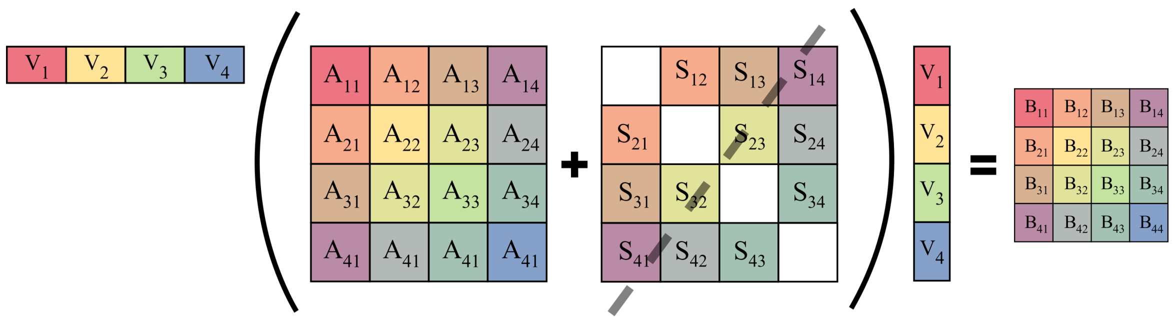


Fig. 5.1 Assembly of the reduced block matrix $\{B_{i,j}\}_{i,j=1}^4$ through the projection onto the local reduced basis $\{V_i\}_{i=1}^4$ of the full-order partitioned matrix $A = \{A_{i,j}\}_{i,j=1}^4$ when considering 4 subdomains. The natural DGM penalty terms are included in the matrix A without the need for additional penalty terms $\{S_{i,j}\}_{i\neq j,\ ,i,j=1}^4$ to impose stability at the reduced level.

5.5.2 Repartitioning strategy

A great number of subdomains can pollute the efficiency of the developed DD-ROMs at the online stage since the reduced dimension would be $\sum_{i=1}^K n_i$ that scales linearly with the number of cores if the local dimensions are equal. In order to keep the computational savings in the assembly of the affine decomposition at the offline stage, we may want to preserve the distributed property of our ROM. One possible solution is to fix a reduced number of subdomains $k \ll K$ such that $\sum_{i=1}^k n_i$ is small enough to achieve a significant speedup with respect to the FOM. The additional cost with respect to a monodomain ROM is associated with the evaluation of the k local reduced basis with SVD and the assembly of the affine decomposition operators. The new k reduced subdomains do not need to be agglomerations of the FOM subdomains, hence, different strategies to assemble the new k reduced subdomains can be investigated.

The number of subdomains K was kept the same as the FOM since it is necessary to collect the snapshots efficiently at the full-order level through ***p4est***. However, if we decide to repartition our computational domain, we can develop decomposition strategies that reduce $\sum_{i=1}^K n_i$. Ideally, having in mind the Eckhart-Young theorem, a possible strategy is to lump together all the dofs of the cells that have a fast decaying Kolmogorov n-width, and focus on the remaining ones. We test this procedure in the practical case $k = 2,\ K = 4$ to perform numerical experiments in section 5.5.3.

To solve the classification problem of partitioning the elements of the mesh into k subdomains, we describe here two scalar indicators that will be used as metrics. For $k = 2$ subdomains, it will be sufficient to choose the percentage of cells P_l corresponding to the lowest values of the chosen scalar indicator. Other strategies for $k > 2$ may also involve clustering algorithms and techniques to impose connectedness of the clusters, as done for local dimension reduction in parameter spaces in [220]. A first crude and cheap indicator to repartition the computational domain is the cellwise variance of the training snapshots, as it measures how well, in mean squared error, the training snapshots are approximated by their mean, $\forall T \in \mathcal{T}_h$.

Definition 7 (Cellwise variance indicator)**.** *We define the cellwise variace indicator $I_{var}:\mathcal{T}_h\to\mathbb{R}^+$,*

$$I_{var}(T)=\int_T \|Var(\{\mathbf{z}(\boldsymbol{\rho}_i)\}_{i=1}^n)\|_{L^2(\mathbb{R}^m)}\,d\mathbf{x},\qquad (Var(\{\mathbf{z}(\boldsymbol{\rho}_i)\}_{i=1}^n))_l=\frac{1}{n}\sum_{i=1}^n\left|\mathbf{z}_l(\boldsymbol{\rho}_i)-\frac{1}{n}\sum_{j=1}^n\mathbf{z}_l(\boldsymbol{\rho}_j)\right|^2,\quad l=1,\dots,m, \tag{5.84}$$

where $n>0$ is the number of training DGM solutions $\{\mathbf{z}(\boldsymbol{\rho}_i)\}_{i=1}^n$ with $\mathbf{z}(\boldsymbol{\rho}_i):\Omega\subset\mathbb{R}^d\to\mathbb{R}^m$, $\forall i\in\{1,\dots,n\}$.

Note that the indicator is a scalar function on the set of elements of the triangulation $\mathcal{T}_h$. This is possible thanks to the assumption that boundaries of the subdomains belong to the interfaces of the elements of $\mathcal{T}_h$. Moreover, when this hypothesis is not fulfilled, we would need to evaluate additional operators to impose penalties at the algebraical interfaces between subdomains that are not included in the set $\mathcal{F}_h^i\cup\mathcal{F}_h^b$, not to degrade the accuracy.

The cellwise variance indicator is effective for all the test cases for which there is a relatively large region that is not sensitive to the parametric instances, as in our advection diffusion reaction test case in Section 5.5.3. Common examples are all the CFD numerical simulations that have a far-field with fixed boundary conditions. However, the variance indicator may be blind to regions in which the snapshots can be spanned by a one or higher dimensional linear subspace and are not well approximated by a constant field, as in the compressible linear elasticity test case in Section 5.5.3.

In these cases, a valid choice is represented by a cellwise Grassmannian dimension indicator. We denote with D_{T} the number of degrees of freedom associated with each element T, assumed constant in our test cases.

Definition 8 (Cellwise Grassmannian dimension indicator)**.** *Fixed $1\leq n_T\in\mathbb{N}$, and $1\leq n_{neig}\in\mathbb{N}$, we define the cellwise Grassmannian dimension indicator $I_G:\mathcal{T}_h\to\mathbb{R}^+$,*

$$I_G(T)=\|X_T-U_TU_T^TX_T\|_F, \tag{5.85}$$

where $X_T\in\mathbb{R}^{n_{neig}D_T\times n}$ is the snapshots matrix restricted to the cell T and its n_{neig} nearest neighbours, and $U_T\in\mathbb{R}^{n_{neig}D_T\times r_T}$ are the modes of the truncated SVD of X_T with dimension r_T.

The cellwise Grassmannian dimension indicator I_G is a measure of how well the training snapshots restricted to a neighbour of each cell are approximated by a n_T dimensional linear subspace. Employing this indicator, we recover an effective repartitioning of the computational subdomain of the compressible linear elasticity test case, see Section 5.5.3. The Grassmannian indicator has two hyperparameters that we fix for each test case in section 5.5.3: the number of nearest-neighbour cells is $n_{\mathrm{neigh}}=3$ and the number of reduced local dimension used to evaluate the L^2 reconstruction error is $n_T=1$.

We remark that both indicators do not guarantee that the obtained subdomains belong to the same connected components and, though this might be a problem in terms of connectivity and computational costs for the FOM, at the reduced level this does not affect the online computational costs. Nevertheless, in the tests we perform, the obtained subdomains are connected.

Now, the assembly of the affine decomposition proceeds as explained in Section 5.5.1 with the difference that at least one local reduced basis and reduced operator is split between at least 2 subdomains/cores. A schematic block matrix representation of the procedure is shown in Figure 5.2.

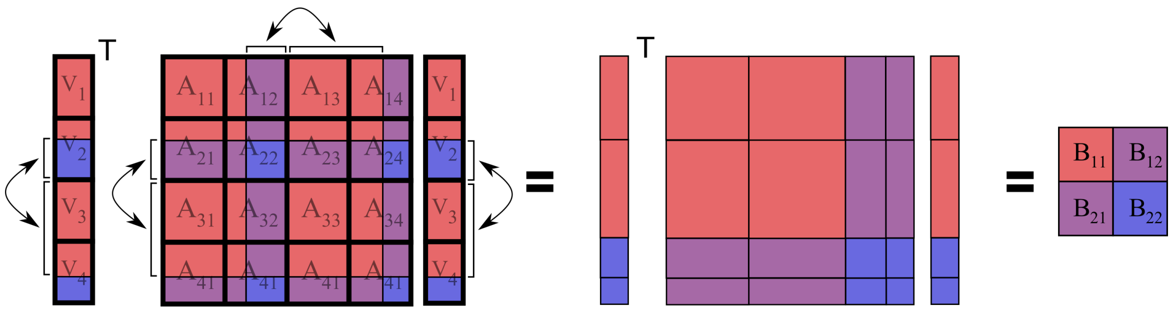


Fig. 5.2 Repartitioning of the reduced block matrix shown in Figure 5.1 from $K = 4$ subdomains to $k = 2$ repartitioned subdomains. The projection from the full-order matrix $A = \{A_{i,j}\}_{i,j=1}^{4}$ to the reduced matrix $B = \{B_{i,j}\}_{i,j=1}^{2}$ is sketched. It is performed locally in a distributed memory setting, the re-ordering shown by the arrows is reported only to visually see which block structure of the full-order matrix A would correspond to the blocks of the reduced matrix B.

5.5.3 Numerical experiments

In this section, we test the presented methodology for different linear parametric partial differential equations: the Maxwell equations in stationary regime in section 5.5.3 (**MS**), the compressible linear elasticity equations in section 5.5.3 (**CLE**) and the advection diffusion reaction equations in section 5.5.3 (**ADR**). We study two different parametrizations for the test cases **MS** and **CLE**: one with parameters that affect the whole domain **MS1** and **CLE1**, and one with parameters that affect independently different subdomains **MS2** and **CLE2**. We show a case in which DD-ROMs work effectively **MS2** and a case **CL2** in which the performance is analogous to single domain ROMs, even if the parameters have a local influence.

We test the effectiveness of the *a posteriori* error estimates introduced in section 5.4.1, the accuracy of DD-ROMs for $K = 4$ and the results of repartitioning strategies with $k = 2$ subdomains. When performing a repartition of the computational domain Ω in subdomains $\{\Omega_i\}_{i=1}^{k}$ with reduced dimensions $\{r_{\Omega_i}\}_{i=1}^{k}$, we call the subdomains with lower values of the variance indicator I_{var}, see definition 7, *low variance* regions and with lower values of the Grassmannian indicator I_G, see definition 8, *low Grassmannian reconstruction error*. The complementary subdomains are the *high variance* and *high Grassmannian reconstruction error* regions, respectively. We show a case (**CLE1**) in which the Grassmannian indicator detects a better partition in terms of local reconstruction error with respect to the variance indicator.

We will observe that the relative errors in R-norm and energy norm and the L^2 relative error estimator and L^2 relative energy norm estimator are the most affected by the domain partitions.

The open–source software library employed for the implementation of the full-order Friedrichs' systems discontinuous Galerkin solvers is `deal.II` [12]. The partition of the computational domain is performed in `deal.II` through the open–source `p4est` package [42]. The distributed affine decomposition data structures are collected in the offline stage and exported in the sparse `NumPy` format [121]. The reduced order models and the repartition of the computational domains are implemented in Python with MPI-based parallel distributed computing `mpi4py` [65] and `petsc4py` [17] for solving the linear full-order systems through `MUMPS` [6], a sparse direct solver.

Maxwell equations in stationary regime (MS)

We consider the parametric Maxwell equations in the stationary regime in $d = 3$ spatial dimensions, with $m = 6$ equations, on a torus $\Omega \subset \mathbb{R}^3$ with inner radius $r = 0.5$ and outer radius $R = 2$ centered in $\mathbf{0}$ and lying along the (x,z) plane:

$$\begin{pmatrix} \mu\mathbf{H} + \nabla\times\mathbf{E} \\ \sigma\mathbf{E} - \nabla\times\mathbf{H} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{f} \end{pmatrix}, \qquad \forall \mathbf{x}\in\Omega, \tag{5.86}$$

the tangential homogeneous boundary conditions $\mathbf{n}\times\mathbf{E} = \mathbf{0}$ are applied with the boundary operator (5.18). We vary the parameters in the interval $\boldsymbol{\rho} = (\mu,\sigma) \in [0.5,2]\times[0.5,3] \subset \mathbb{R}^2$, leading to $\mu_0 = \min(\mu,\sigma)$.

We consider the exact solutions

$$\begin{aligned}
\mathbf{H}_{\text{exact}}(\mathbf{x}) &= -\frac{1}{\mu}\left(\frac{2xy}{\sqrt{x^2+z^2}}, \frac{-4y^2\sqrt{x^2+z^2}+\sqrt{x^2+z^2}(-12(x^2+z^2)-15)+32(x^2+z^2)}{4(x^2+z^2)}, \frac{2xy}{\sqrt{x^2+z^2}}\right),\\
\mathbf{E}_{\text{exact}}(\mathbf{x}) &= \left(\frac{z}{\sqrt{x^2+z^2}}, 0, -\frac{x}{\sqrt{x^2+z^2}}\right)\cdot\left(r^2-y^2-\left(R-\sqrt{x^2+z^2}\right)^2\right).
\end{aligned}$$

We remark that the exact solutions can be approximated with a linear reduced subspace of dimension 1, if we obtained the reduced basis with a partitioned SVD on the fields $(\mathbf{H},\mathbf{E})$ separately. We do not choose this approach and perform a monolithic SVD to test the convergence of the approximation with a DD-ROMs with respect to the local reduced dimensions. The source terms are defined consequently as

$$\mathbf{g}(\mathbf{x}) = 0, \qquad \mathbf{f}(\mathbf{x}) = \sigma\mathbf{E}_{\text{exact}} - \nabla\times\mathbf{H}_{\text{exact}}. \tag{5.87}$$

We consider two parametric spaces:

$$\boldsymbol{\rho} = (\mu,\sigma) \in [0.5,2]\times[0.5,3] = \mathcal{P}_1 \subset \mathbb{R}^2, \qquad (\textbf{MS1}) \tag{5.88a}$$

$$\boldsymbol{\rho} = (\mu_1,\sigma_1,\mu_2,\sigma_2) \in [0.5,2]\times[0.5,3]\times[0.5,2]\times[0.5,3] = \mathcal{P}_2 \subset \mathbb{R}^4, \qquad (\textbf{MS2}) \tag{5.88b}$$

where in the second case, the parameters μ and σ are now piecewise constant:

$$\mu = \begin{cases} \mu_1, & x < 0, \\ \mu_2, & x \geq 0, \end{cases} \qquad \sigma = \begin{cases} \sigma_1, & x < 0, \\ \sigma_2, & x \geq 0, \end{cases} \tag{5.89}$$

where $\mathbf{x} = (x,y,z) \in \Omega \subset \mathbb{R}^3$. In Figure 5.3, we show solutions for $\mu = \sigma = 1$ and for discontinuous values of the parameters: $\mu_1 = \sigma_1 = 1$ in $\{x < 0\} \cap \Omega$ and $\mu_2 = \sigma_2 = 2$ in $\{x \geq 0\} \cap \Omega$. The FOM partitioned and DD-ROM repartitioned subdomains are shown in Figure 5.4. For **MS1**, we choose the variance indicator to repartition the computational subdomain in two subsets: $P_l = 20\%$ of the cells for the *low variance* part and 80% for the *high variance* part. For **MS2**, we split the computational domain into two parts with the Grassmannian indicator and $P_l = 50\%$.

At the end of this subsection a comparison of the effectiveness of DD-ROMs with and without discontinuous parameters will be performed, the associated error plots are reported in Figure 5.6 and Figure 5.7. We will see that, for this simple test case **MS2**, there is an appreciable improvement of the accuracy when the computational domain subdivisions match the regions $\{x < 0\} \cap \Omega$ and $\{x \geq 0\} \cap \Omega$ in which μ and σ are constant. Such subdivision is detected by the Grassmannian indicator with $P_l = 50\%$, as shown in Figure 5.4 on the right. This is the archetypal case in which DD-ROMs are employed successfully, in comparison with **MS1** for which there is no significant improvement with respect to classical global linear reduced basis.

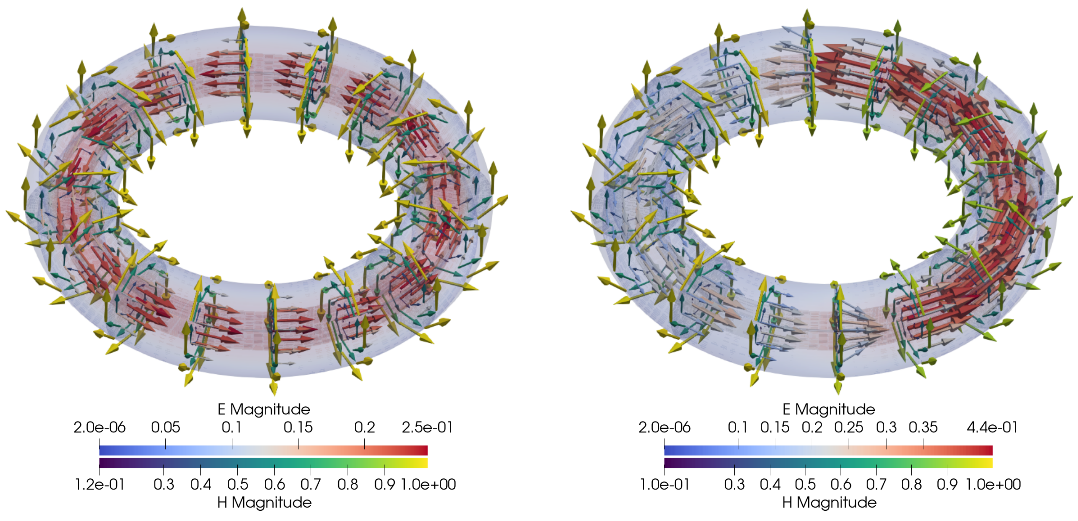

E Magnitude
2.0e-06 0.05 0.1 0.15 0.2 2.5e-01
1.2e-01 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0e+00
H Magnitude
E Magnitude
2.0e-06 0.1 0.15 0.2 0.25 0.3 0.35 4.4e-01
1.0e-01 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0e+00
H Magnitude

Fig. 5.3 **MS**. Electric and magnetic fields of the Maxwell equations in stationary regime with Dirichlet homogeneous boundary conditions $\mathbf{n} \times \mathbf{E} = \mathbf{0}$. The vectors of the magnetic and electric fields are scaled by 0.5 and 2 of their magnitude respectively. **Left**: **MS1**, $\mu = \sigma = 1$, test case errors shown in Figure 5.6. **Right**: **MS2**, $\mu = \sigma = 1$ in $\{x < 0\} \cap \Omega$ and $\mu = \sigma = 2$ in $\{x \geq 0\} \cap \Omega$, test case errors shown in Figure 5.7.

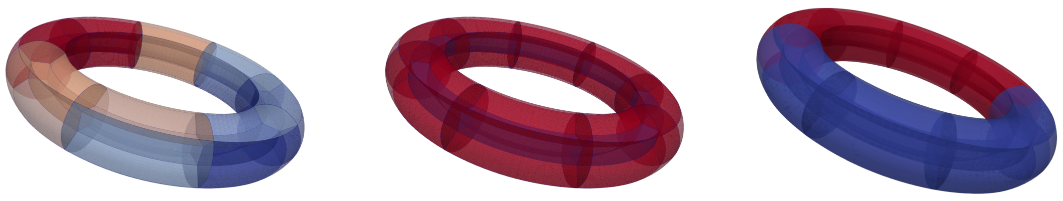


Fig. 5.4 **MS**. **Left**: FOM computational domain partitioned in $K = 4$ subdomains inside `deal.II`. **Center**: **MS1**, DD-ROM repartition of the computational subdomain $k = 2$ with the cellwise variance indicator I_{var}, definition 7: 20% of the cells belong to the *low variance* part, represented in blue inside the torus, and the other 80% belong to the *high variance* part, represented in red. **Right**: **MS2**, DD-ROM repartition with variance indicator $P_l = 50\%$. The computational domain is exactly split at the interfaces that separate the subdomains $\{x < 0\} \cap \Omega$ and $\mu = \sigma = 2$ in which the parameters μ and σ are constant.

In Figure 5.5, we show how the different thresholds applied to the two indicators can affect the reconstruction error on a reduced space with $n_{\Omega_i} = 3$. All the lines plot the local relative error computed on different subdomains (either one of the k DD-ROM subdomains or on the whole domain). On the x-axis it is shown the percentage of cells that are grouped into the low variance or low Grassmannian DD-ROM subdomain. We observe that the cellwise variance indicator is a good choice for the purpose of repartitioning the subdomain from $K = 4$ to $k = 2$. Indeed, it is possible to build a low variance subdomain (value of the abscissa 20% in Figure 5.5) with a low local relative reconstruction error ($5 \cdot 10^{-4}$) with respect to the global one ($8.6 \cdot 10^{-4}$). This means that by choosing the threshold $P_l = 20\%$ for the low variance subdomain, we should be able to use less reduced basis functions for that subdomain without affecting too much the global error.

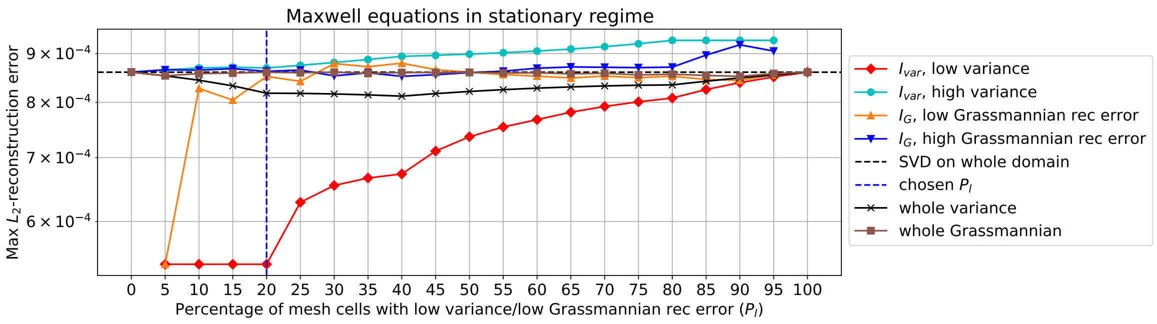


Fig. 5.5 **MS1**. Local relative L^2-reconstruction errors of the snapshots matrix restricted to the two subdomains of the repartitioning performed with the indicator I_{var} (in red and light-blue), Definition 7, and I_{G} (in orange and blue), Definition 8. The relative L^2-reconstruction error attained on the whole domain is shown in black for the indicator I_{var} and in brown for the indicator I_{G}. The local reduced dimensions used to evaluate the local reconstruction errors is $n_{\Omega_i} = 3,\ i = 1,2$.

Test case **MS1**. We evaluate $n_{\text{train}} = 20$ training full-order solutions and $n_{\text{test}} = 80$ test full-order solutions, corresponding to a uniform independent sampling from the parametric domain

$\mathcal{P}_1 \subset \mathbb{R}^2$. Figure 5.6 shows the result relative to the relative L^2-error and relative errors in energy norm, with associated *a posteriori* estimators. The numberd abscissae $0, 5, 10, \dots, 95$ represents the train parameters $n_{\text{train}} = 20$ while the others $n_{\text{test}} = 80$ parameters are the test set. For these studies, we have fixed the local reduced dimensions to $n_{\Omega_i} = 3,\ i = 1, \dots, K$ for $K = 4$, $n_\Omega = 3$ for the whole computational domain and $n_{\Omega_1} = 2,\ n_{\Omega_2} = 3$ for the DD-ROM repartitioned case with $k = 2$. This choice of repartitioning with the 20% of *low variance* cells and local reduced dimension $n_{\Omega_1} = 2$ does not deteriorate significantly the accuracy and the errors almost coincide for all approaches. However, unless the parameters σ, μ assume different discontinuous values in the computational domain Ω, DD-ROMs are not advisable for this test case if the objective is improving the predictions' accuracy.

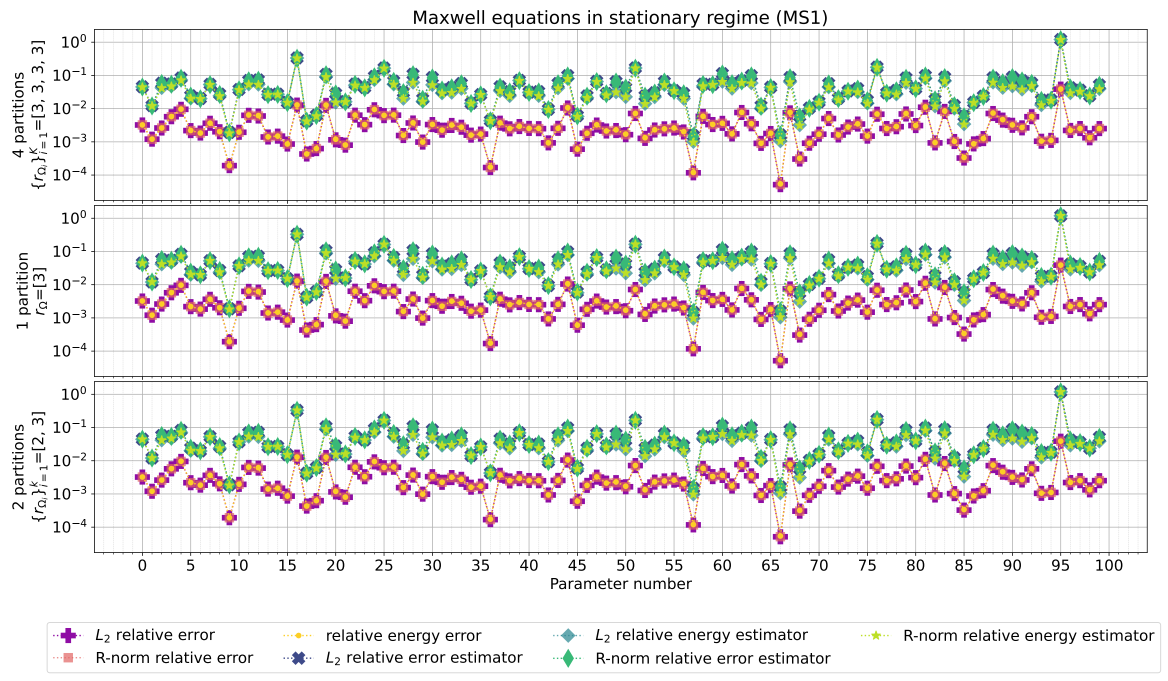


Fig. 5.6 **MS1**. Errors and estimators for Maxwell equations corresponding to the $n_{\text{train}} = 20$ uniformly sampled training snapshots corresponding to the abscissae $0, 5, 10, \dots, 95$, and $n_{\text{test}} = 80$ uniformly sampled test snapshots, corresponding to the other abscissae. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [3,3,3,3]$ for $K = 4$ partitions, $n_\Omega = 3$ for $k = 1$ partition, and $\{n_{\Omega_i}\}_{i=1}^{k} = [2,3]$ for $k = 2$ partitions. For the case $k = 2$ we employed the cellwise variance indicator I_G, Definition 7, with $P_l = 20\%$. It can be seen that even by reducing the local dimension from 3 to 2 of one of the $k = 2$ repartitioned subdomains, the accuracy of the predictions does not decrease sensibly.

Test case **MS2**. Similarly to the previous case, we evaluate $n_{\text{train}} = 20$ training full-order solutions and $n_{\text{test}} = 80$ test full-order solutions, corresponding to a uniform independent sampling from the parametric domain $\mathcal{P}_2 \subset \mathbb{R}^4$. As mentioned above, if we vary the parameters $\boldsymbol{\rho} = (\mu, \sigma)$ discontinuously on the subdomains $\{x \geq 0\} \cap \Omega$ and $\mathbf{x} \in \{x < 0\} \cap \Omega$, we obtain the results shown in Figure 5.7. It can be seen that repartitioning Ω in $k = 2$ DD-ROM subdomains with the local Grassmannian indicator I_G and $P_l = 50\%$ produces effective DD-ROMs compared to the case of a single reduced solution manifold for the whole computational domain and for the DD-ROM with $k = 4$ for which the subdomains do not match $\{x < 0\} \cap \Omega$ and $\{x \geq 0\} \cap \Omega$. In this case, we kept the local dimension of DD-ROM repartitioned case with $k = 2$ equal $n_{\Omega_1} = n_{\Omega_2} = 3$. For this simple test case, there is an appreciable improvement for some test parameters in the accuracy for $k = 2$ instead of $K = 4$ or a classical global linear basis ROM.

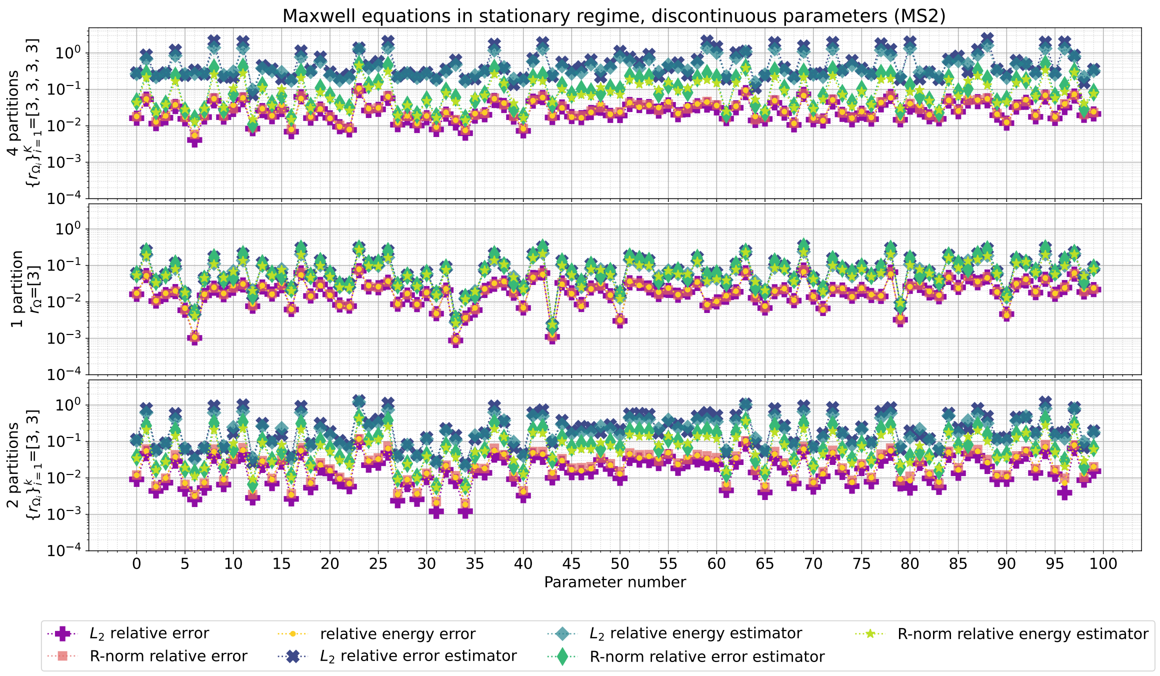


Fig. 5.7 **MS2**. Errors and estimators for Maxwell equations with discontinuous μ and σ corresponding to the $n_{\text{train}} = 20$ uniformly sampled training snapshots corresponding to the abscissae $0, 5, 10, \dots, 95$, and $n_{\text{test}} = 80$ uniformly sampled test snapshots, corresponding to the other abscissae. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [3,3,3,3]$ for $K = 4$ partitions, $n_\Omega = 3$ for $k = 1$ partition, and $\{n_{\Omega_i}\}_{i=1}^{k} = [3,3]$ for $k = 2$ partitions. For the case $k = 2$ we employed the cellwise local Grassmannian dimension indicator I_G, Definition 7, with $P_l = 50\%$. The subdivisions detected exactly match the subdomains $\{x < 0\} \cap \Omega$ and $\{x \geq 0\} \cap \Omega$ in which the parameters are constant. An improvement of the predictions can be appreciated for some test parameters when employing $k = 2$ repartitions.

In Table 5.1, we list the computational times and speedups for a simulation with the different methods. For an error convergence analysis with respect to the size of the reduced space, we refer to section 5.5.3.

Compressibile linear elasticity (CLE)

Next, we consider the parametric compressible linear elasticity system in $d = 3$ physical dimensions with a cylindrical shell along the z-axis as domain: the inner radius is 1, outer radius 3 and height 10,

Table 5.1 **MS1**. Average computational times and speedups for ROM and DD-ROM approaches for Maxwell equations. The speedup is computed as the FOM computational time over the ROM one. The FOM runs in parallel with 4 cores, so "FOM time" refers to wallclock time.

FOM		ROM			DD-ROM		
N_h	time	n	time	speedup	n_i	time	speedup
6480	254.851 [ms]	3	51.436 [μs]	~ 495	[3, 3, 3, 3]	62.680 [μs]	~ 406

and the base centered in $\mathbf{0}$. The $m = 12$ equations of the FS are

$$\begin{pmatrix} \boldsymbol{\sigma} - \mu_1(\nabla\cdot\mathbf{u})\mathbb{I}_3 - 2\mu_2\frac{(\nabla\mathbf{u}+(\nabla\mathbf{u})^t)}{2} \\ -\frac{1}{2}\nabla\cdot(\boldsymbol{\sigma}+\boldsymbol{\sigma}^t)+\mu_3\mathbf{u} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{f} \end{pmatrix}, \qquad \forall \mathbf{x}\in\Omega\subset\mathbb{R}^3, \tag{5.90}$$

where $\boldsymbol{\rho} = (\mu_1,\mu_2,\mu_3) \in [100,1000]^2\times[1,1] = \mathcal{P}\subset\mathbb{R}^3$ and $\mathbf{f} = (0,-1,0)$. The system can be rewritten as FS as in (5.20). We define the boundaries

$$\Gamma_D = \partial\Omega\cap\{z=0\}, \qquad \Gamma_N = \partial\Omega\setminus\Gamma_D. \tag{5.91}$$

Mixed boundary conditions are applied with the boundary operator (5.24): homogeneous Dirichlet boundary conditions are imposed on Γ_D and homogeneous Neumann boundary conditions on Γ_N.

We consider two parametric spaces:

$$\boldsymbol{\rho} = (\mu_1,\mu_2) \in [100,1000]^2 = \mathcal{P}_1\subset\mathbb{R}^2, \qquad (\textbf{CLE1}) \tag{5.92a}$$

$$\boldsymbol{\rho} = (\mu_1,\mu_2,f_1,f_2) \in [100,1000]^2\times[-2,2]^2 = \mathcal{P}_2\subset\mathbb{R}^4, \qquad (\textbf{CLE2}) \tag{5.92b}$$

where in the second case, the source term $\mathbf{f}$ is now piecewise constant:

$$\mathbf{f} = \begin{cases} f_1\cdot(0,-1,0), & z<5, \\ f_2\cdot(0,-1,0), & z\geq 5. \end{cases} \tag{5.93}$$

We show two sample solutions for $\mu_1 = \mu_2 = 1000$ in Figure 5.8 for **CLE1** and $\mu_1 = \mu_2 = 1000$, $f_1 = 1$ and $f_2 = -1$ for **CLE2**, on the left and on the right, respectively. The partitioned and repartitioned subdomains are shown in Figure 5.9. For the first case **CLE1** we employ a mesh of 24 cells and 7776 dofs, for the second **CLE2** a mesh of 60 cells and 19440 dofs.

Test case **CLE1**. This test case presents no region for which the restricted solutions are more or less approximable with a constant field, as would be detected by the variance indicator: as shown in Figure 5.10, the local relative L^2-reconstruction error in the region with *low variance*, assigned by I_{var}, deteriorates from the value $2\cdot10^{-3}$ of the abscissae 0% and 100% to $1\cdot10^{-2}$ of the abscissae 4%. Nonetheless, despite the parametric solutions are not approximabile efficiently with a constant field, they are well represented by a one-dimensional linear subspace in the region located by the

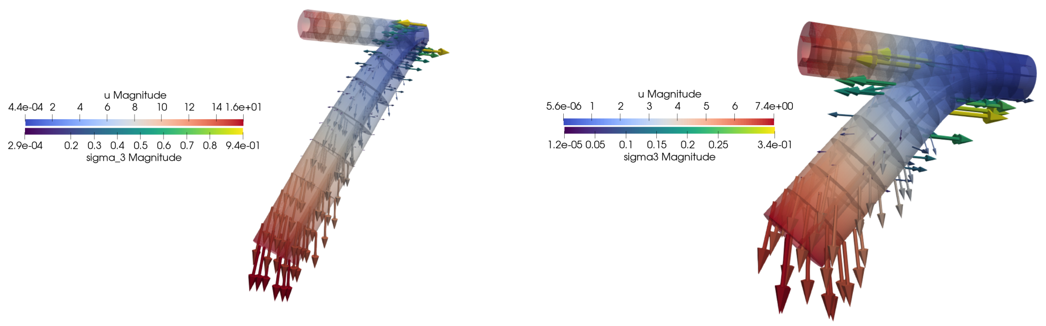

u Magnitude
4.4e-04 2 4 6 8 10 12 14 1.6e+01
2.9e-04 0.2 0.3 0.4 0.5 0.6 0.7 0.8 9.4e-01
sigma_3 Magnitude
u Magnitude
5.6e-06 1 2 3 4 5 6 7.4e+00
1.2e-05 0.05 0.1 0.15 0.2 0.25 3.4e-01
sigma3 Magnitude

Fig. 5.8 **CLE**. **Left:** solution of the compressible linear elasticity FS **CLE1** with parameter values $\mu_1 = \mu_2 = 1000$. The cylindrical shell displacement $\mathbf{u}$, and with a different colorbar also the field $\sigma\mathbf{e_z}$, named `sigma_3`, are shown. At the extremity close to $z = 0$ homogeneous Dirichlet boundary conditions are imposed. **Right:** solution of the test case **CLE2** with discontinuous values of the source terms along the computational domain $\{z < 5\} \cap \Omega$ and $\{z \geq 5\} \cap \Omega$: $\mu_1 = \mu_2 = 1000$, $f_1 = 1$ and $f_2 = -1$.

cellwise Grassmannian dimension indicator I_G, for $P_l = 12\%$. The associated *low local Grassmannian dimension* region for $P_l = 12\%$ is shown in Figure 5.9 in blue.

Also in this test case, the employment of DD-ROMs is not advisable, since there are little gains in the local relative L^2-reconstruction error for the *low local Grassmannian dimensional* region (values around $3 \cdot 10^{-3}$, in orange for the abscissa $P_l = 12\%$, in Figure 5.10). The choice of local reduced dimensions $n_{\Omega_1} = 2$ and $n_{\Omega_2} = 3$ does not affect greatly the errors shown in Figure 5.11. Also in this case, we evaluate $n_{\text{train}} = 20$ training full-order solutions and $n_{\text{test}} = 80$ test full-order solutions, corresponding to a uniform independent sampling from the parametric domain $\mathcal{P} \subset \mathbb{R}^3$. Also for these studies, we have fixed the local dimensions to $n_{\Omega_i} = 3,\ i = 1, \dots, K$ for $K = 4$, $n_\Omega = 3$ for the whole computational domain and $n_{\Omega_1} = 2$, $n_{\Omega_2} = 3$ for the repartitioned case with $k = 2$.

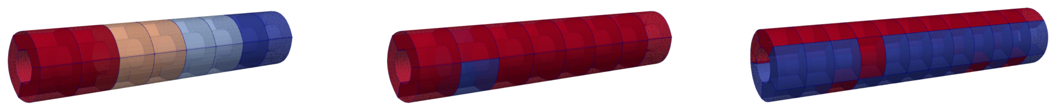


Fig. 5.9 **CLE. Left:** computational subdomains partitioned in $K = 4$ subdomains by `petsc4py` inside `deal.II`. **Center:** test case **CLE1** repartition of the computational subdomain $k = 2$ with the cellwise Grassmannian dimension indicator I_G, Definition 8: 12% of the cells belong to the *low local Grassmannian dimension* part, represented in blue inside the torus, and the other 88% belong to the *high local Grassmannian dimension* part, represented in red. **Right:** test case **CLE2** repartition of the computational subdomain $k = 2$ with the cellwise Grassmannian dimension indicator I_G and $P_l = 50\%$.

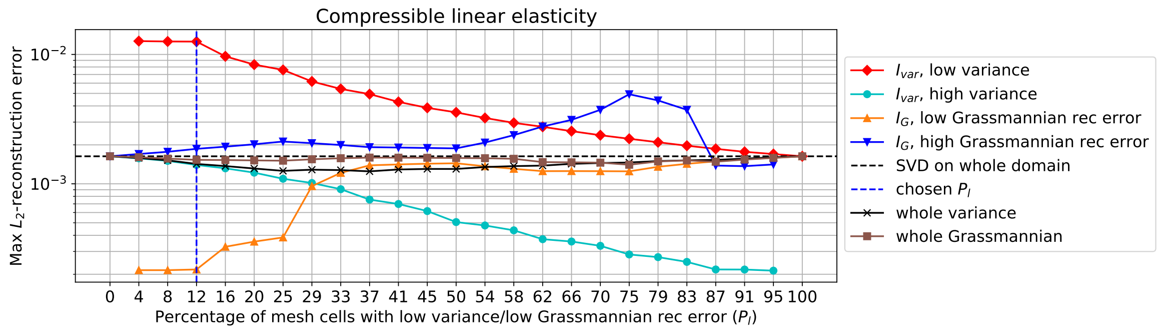


Fig. 5.10 **CLE1.** Local relative L^2-reconstruction errors of the snapshots matrix for elasticity equations restricted to the two subdomains of the repartitioning performed with the indicator I_{var} (in red and light-blue), Definition 7, and I_{G} (in orange and blue), Definition 8. The relative L^2-reconstruction error attained on the whole domain is shown in black for the indicator I_{var} and in brown for the indicator I_{G}. The local reduced dimensions used to evaluate the local reconstruction errors is $n_{\Omega_i} = 3,\ i = 1,2$.

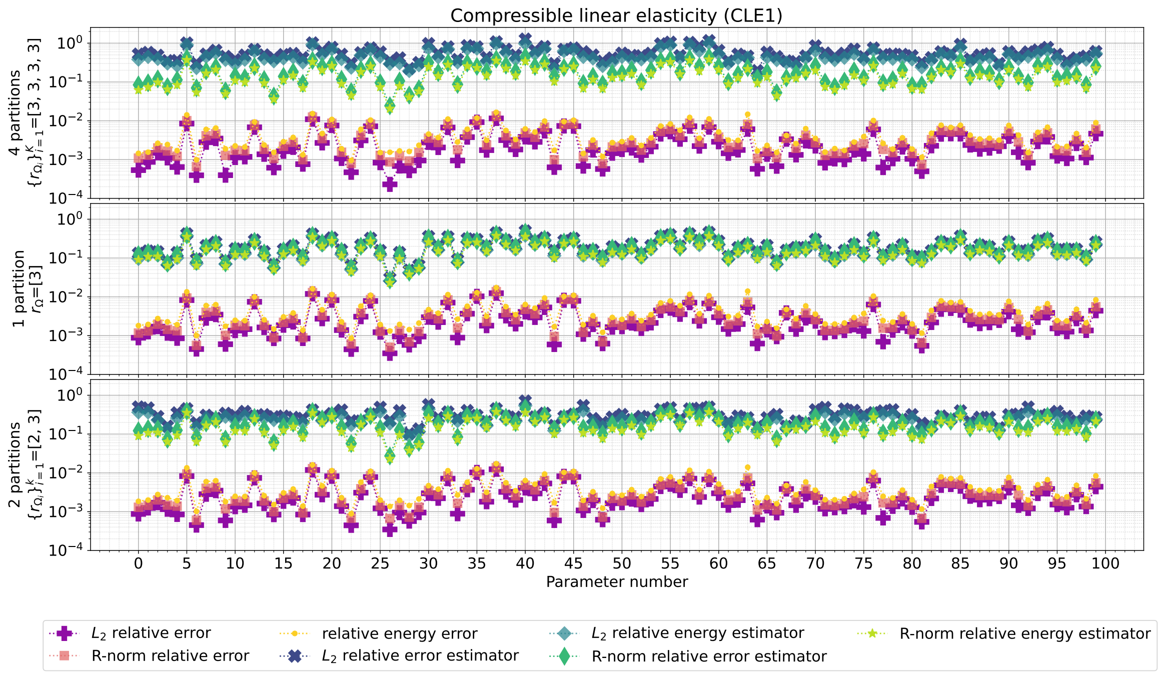


Fig. 5.11 **CLE1.** Errors and estimators for elasticity equations corresponding to the $n_{\text{train}} = 20$ uniformly sampled training snapshots corresponding to the abscissae $0, 5, 10, \dots, 95$, and $n_{\text{test}} = 80$ uniformly sampled test snapshots, corresponding to the other abscissae. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [3, 3, 3, 3]$ for $K = 4$ partitions, $r_\Omega = 3$ for $k = 1$ partition, and $\{n_{\Omega_i}\}_{i=1}^{k} = [2, 3]$ for $k = 2$ partitions. For the case $k = 2$ we employed the cellwise local Grassmannian dimension indicator I_G, Definition 8, with $P_l = 12\%$.

Test case **CLE2**. Similarly to the previous case, we evaluate $n_{\text{train}} = 20$ training full-order solutions and $n_{\text{test}} = 80$ test full-order solutions, corresponding to a uniform independent sampling from the parametric domain $\mathcal{P}_2 \subset \mathbb{R}^4$. This time, if we vary the parameters f_1 and f_2 inside different subdomains $\{z \geq 5\} \cap \Omega$ and $\{z < 5\} \cap \Omega$, we obtain the results shown in Figure 5.12. It can be seen that repartitioning Ω in $k = 2$ DD-ROM subdomains with the local Grassmannian indicator I_G and $P_l = 50\%$ does not produce more accurate DD-ROMs compared to the case of a single reduced solution manifold for the whole computational domain and for the DD-ROM with $k = 4$. In this case, we kept the local dimension of DD-ROM repartitioned case with $k = 2$ equal $n_{\Omega_1} = n_{\Omega_2} = 3$. For this simple test case, there is not an appreciable improvement for some test parameters in the accuracy for $k = 2$ instead of $K = 4$ or a classical global linear basis ROM. The reason is that even if the parameters f_1 and f_2 affect different subdomains of Ω, the solutions on the whole domain are still well correlated. Differently from the previous test case **MS2** from section 5.5.3, this is a typical case for which DD-ROMs are not effective, even if the parametrization affects independently two regions of the whole domain Ω.

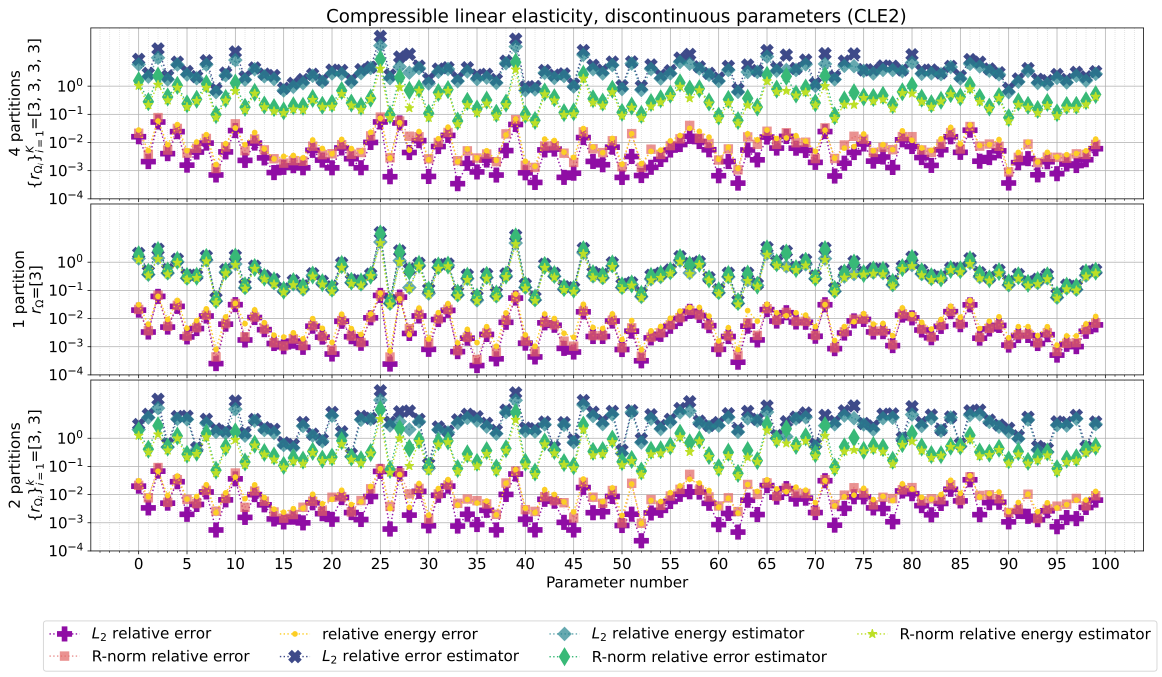


Fig. 5.12 **CLE2.** Errors and estimators corresponding to the $n_{\text{test}=80}$ uniformly sampled test snapshots corresponding to the abscissae $0, 5, 10, \dots, 95$, and $n_{\text{train}=20}$ uniformly sampled training snapshots, corresponding to the other abscissae. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [3, 3, 3, 3]$ for $K = 4$ partitions, $n_\Omega = 3$ for $k = 1$ partition, and $\{n_{\Omega_i}\}_{i=1}^{k} = [3, 3]$ for $k = 2$ partitions. For the case $k = 2$ we employed the cellwise variance dimension indicator I_{var}, Definition 7, with $P_l = 50\%$.

In Table 5.2, we list the computational times and speedups for a simulation with the different methods. For an error analysis with respect to the size of the reduced space, we refer to section 5.5.3.

Table 5.2 **CLE.** Average computational times and speedups for ROM and DD-ROM approaches for Maxwell equations. The speedup is computed as the FOM computational time over the ROM one. The FOM runs in parallel with 4 cores, so "FOM time" refers to wallclock time. The first row correspond to test case **CLE1**, the second to test case **CLE2**.

FOM		ROM			DD-ROM		
N_h	time	n	time	speedup	n_i	time	speedup
7776	411.510 [ms]	3	80.444 [μs]	~ 5115	[3, 3, 3 ,3]	85.108 [μs]	~ 4835
19440	2.080 [s]	3	69.992 [μs]	~ 29718	[3, 3, 3 ,3]	94.258 [μs]	~ 22067

ROM convergence studies

In this section, we validate the DD-ROM implementation, checking the convergence towards the FOM solutions with respect to the dimension of the reduced space. Uniform local reduced dimensions are employed $\{r_{\Omega_i}\}_{i=1}^{K}$ and $\{r_{\Omega_i}\}_{i=1}^{k}$. For each convergence study, 20 uniformly independent samples are used as training dataset and 50 uniformly independent samples as test dataset.

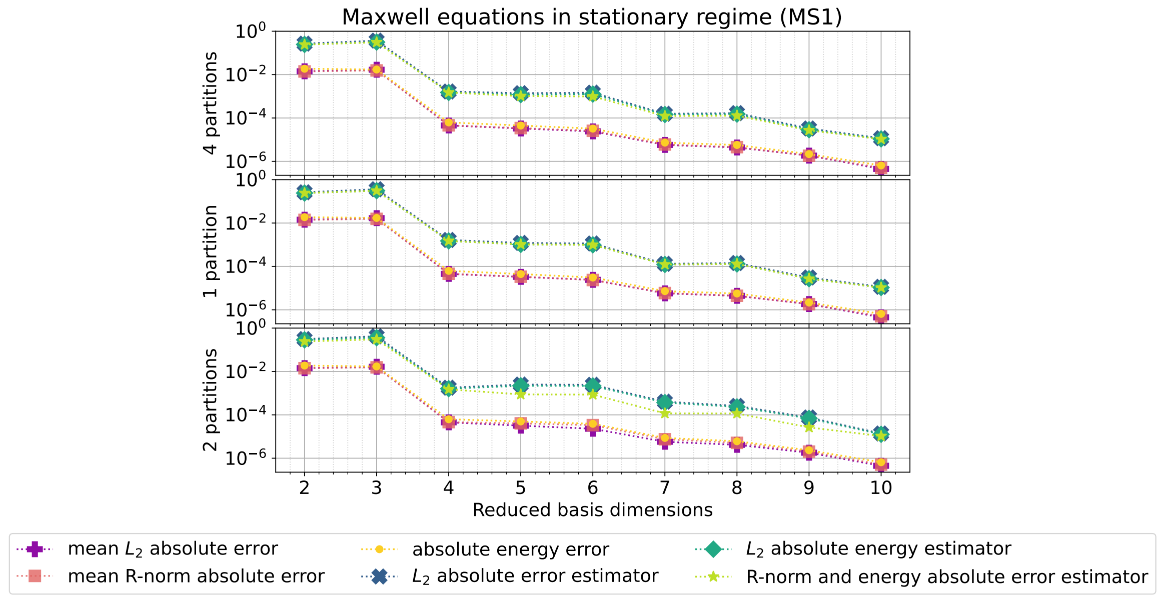


Fig. 5.13 **MS1**. The convergence of DD-ROMS with uniform local reduced dimensions $\{r_{\Omega_i}\}_{i=1}^{K}$ and $\{r_{\Omega_i}\}_{i=1}^{k}$ is assessed. The uniform value of the local reduced dimensions is reported in the abscissae. For this test case, an improvement of the accuracy with respect to the single domain reduced basis is not observed.

In Figure 5.13, we show the L^2-error, the R-error and the energy error decay and their respective error estimators for the Maxwell equations test case **MS1**, section 5.5.3, with constant parameters μ and σ on the whole domain. We clearly see an exponential behavior in the error as we add basis functions. On the other hand, we do not observe strong differences between the ROM, DD-ROM with

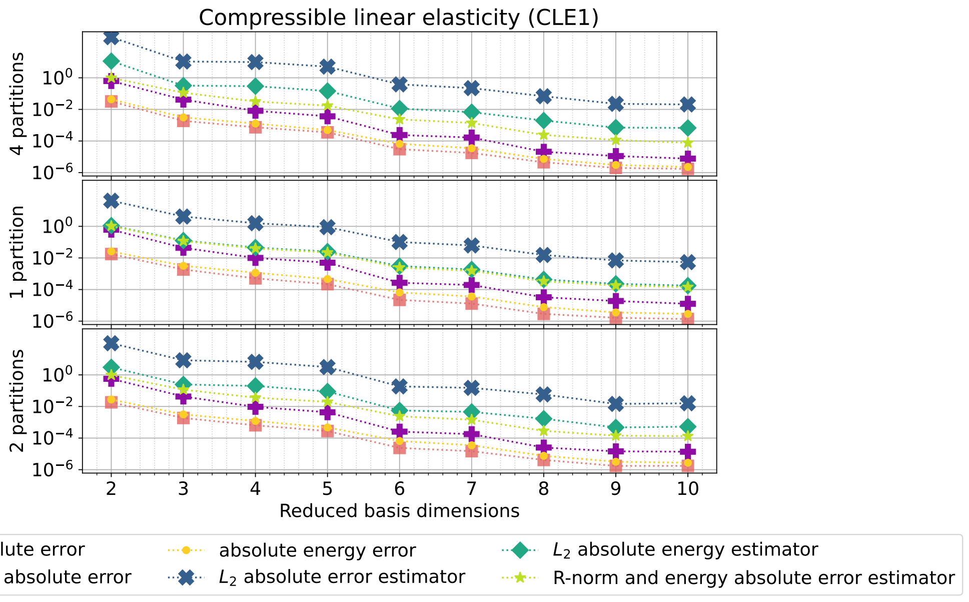


Fig. 5.14 **CLE1**. The convergence of DD-ROMS with uniform local reduced dimensions $\{r_{\Omega_i}\}_{i=1}^{K}$ and $\{r_{\Omega_i}\}_{i=1}^{k}$ is assessed. The uniform value of the local reduced dimensions is reported in the abscissae. For this test case, an improvement of the accuracy with respect to the single domain reduced basis is not observed.

repartitioning and DD-ROM with `deal.II` subdomains, for this simple test case. Similar results can be observed in Figure 5.14, where the same analysis is applied for the compressible linear elasticity test **CLE1** from section 5.5.3.

From these results, it should be clear that the employment of local reduced basis is not always useful to increase the accuracy of the predictions. Nonetheless, it may be used to locally reduce the dimension of the linear approximants. Possible benefits include the adaptation of the computational resources (higher dimensional reduced bases are chosen only where it is necessary) and the possibility to speed up parametric studies and non-intrusive surrogate modelling thanks to the further reduced local dimensions [275, 274].

Typical cases where DD-ROMs are effective to increase the accuracy of the predictions are truly decomposable systems where the parameters affect independently different regions of the computational domain, as in test case **MS2** in section 5.5.3.

Scalar concentration advected by an incompressible flow (ADR)

We consider the parametric semi-linear advection diffusion reaction equation in $d=2$ dimensions, with $m=3$ equations, rewritten in mixed form:

$$\begin{cases} \kappa^{-1}\sigma + \nabla u = 0, & \text{in } \Omega, \\ \nabla\cdot\sigma + \mathbf{v}\cdot\nabla u + u = f, & \text{in } \Omega, \\ \sigma\cdot\mathbf{n} = 0, & \text{on } \Gamma_N\cup\Gamma_{D,0}, \\ u = \sum_{i=1}^{P}\mu_i\chi_{I_i}, & \text{on } \Gamma_D, \end{cases} \tag{5.94}$$

where $\kappa = 0.05$ is fixed for this study,

$$\boldsymbol{\rho} = (\mu_1,\dots,\mu_P)\in\mathcal{P}\subset\mathbb{R}^P, \qquad \mathcal{P} = \{\boldsymbol{\rho}\in\{0,1\}^P|\mu_i=1,\ \mu_j=0,\ \forall j\in\{0,\dots,99\}\backslash\{i\}\}, \tag{5.95}$$

and $\{\chi_{I_i}\}_{i=0}^{N_{\text{par}}}$ are the characteristic functions of the symmetric intervals $I_i = 0\times[-i0.01+1.5, i0.01+2.5]$, with $N_{\text{par}}=99$. The domain is shown in Figure 5.15. The advection velocity $\mathbf{v}$ is obtained from the following incompressible Navier-Stokes equation at $t=2$s:

$$\begin{cases} \partial_t\mathbf{v} + \mathbf{v}\cdot\nabla\mathbf{v} - \nu\Delta\mathbf{v} + \nabla p = \mathbf{0}, & \text{in } \Omega \\ \nabla\cdot\mathbf{v} = 0, & \text{in } \Omega \\ \mathbf{v}\times\mathbf{n} = 0,\ p = 0, & \text{on } \Gamma_N \\ \mathbf{v} = 0, & \text{on } \Gamma_{D,0} \\ \mathbf{v}(t=0) = \mathbf{v}_b, & \text{on } \Gamma_D \end{cases} \tag{5.96}$$

with initial conditions on the boundary Γ_D, $\mathbf{v}_b = \mathbf{v}(x,y,t=0) = (6y(4.1-y)/4.1^2, 0)\in\mathbb{R}^2$ and $\nu\in\mathbb{R}$ such that the Reynolds number is $Re = 100$. The implementation is the one of `step-35` of the tutorials of the `deal.II` library [12]. Homogeneous Neumann boundary on $\Gamma_N\cup\Gamma_{D,0}$ and Dirichlet

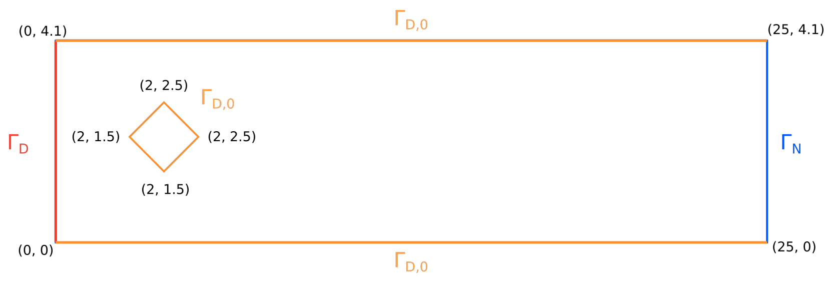


Fig. 5.15 **ADR.** Computational domain of the advection diffusion reaction equation FS (5.94) and the incompressible Navier-Stokes equations (5.96). The boundary conditions specified for each system are reported in the text.

non-homogeneous boundary conditions on Γ_D are applied with the boundary operator (5.30). A sample solution is shown in Figure 5.16 for $\mu_i = 0,\ i = 0,\dots,98$ and $\mu_{99} = 1$, $\kappa = 0.05$. We remark that, for the moment, we consider only fixed values of $\kappa = 0.05$. For the convergence of ROMs to vanishing viscosity solutions with graph neural networks, see Section 5.6.

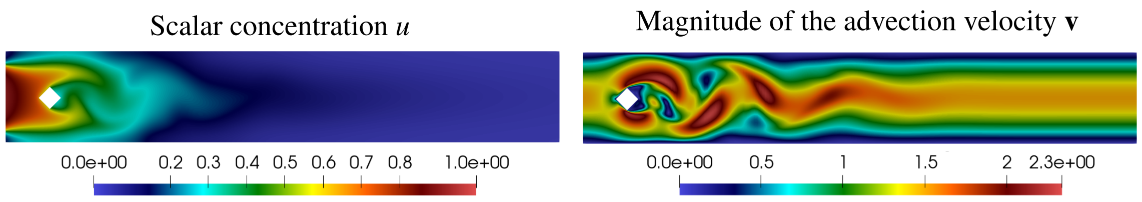


Fig. 5.16 **ADR. Left**: scalar concentration u of the advection diffusion reaction equations (5.94), with $\mu_i = 0,\ i = 0,98$ and $\mu_{99} = 1$, $\kappa = 0.05$. **Right**: advection velocity employed for the FS (5.94), obtained as the velocity $\mathbf{v}$ from the INS (5.96) at $t = 2$s.

The FOM partitioned and DD-ROM repartitioned subdomains are shown in Figure 5.17. We choose the variance indicator to repartition the computational subdomain in two subsets: 21% of the cells for the *low variance* part and 79% for the *high variance* part. With respect to the previous test cases, now it is evident the change in the order of magnitude of the local relative L^2-reconstruction error in Figure 5.18, especially for the cellwise variance indicator I_{var}. We expect that lowering the local reduced dimension of the *low variance* repartitioned region will not affect sensibly the accuracy.

We use for the monodomain approach $n_\Omega = 5$ reduced basis as well as $n_{\Omega_i} = 5$ for $i = 1,\dots,K$ for the FOM partitioned subdomains. In the DD-ROM approach, we can use even $n_{\Omega_1} = 2$ and $n_{\Omega_2} = 5$ for the lower and higher variance subdomains, respectively, without affecting the error of the ROM solution, as we see in Figure 5.19. Indeed, the accuracy in terms of L^2 and energy norms is essentially identical for all approaches, even with so little number of basis functions for the DD-ROM one.

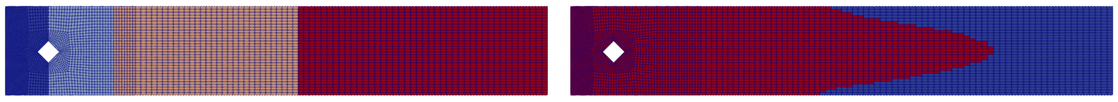


Fig. 5.17 **ADR.** Domain of advection diffusion reaction equation. **Left**: computational subdomains partitioned in $K = 4$ subdomains by `petsc4py` inside `deal.II`. **Right**: DD-ROM repartition of the computational subdomain $k = 2$ with the cellwise variance indicator I_{var}, Definition 7: 21% of the cells belong to the *low variance* part, represented in blue, and the other 79% belong to the *high variance* part in red.

Again, we evaluate $n_{\text{train}} = 20$ training full-order solutions and $n_{\text{test}} = 80$ test full-order solutions, corresponding to the parameter choices $\mu_i = 1$ and $\mu_{\bar{i}} = 0$, for $i = 0,\dots,99$, with fixed viscosity $\kappa = 0.05$, where $\bar{i}$ represents all the indices in $\{0,\dots,99\}$ except from i. So, the training snapshots correspond to $i = 0,5,10,\dots,95$. For these studies we have fixed the local dimensions to $n_{\Omega_i} =$

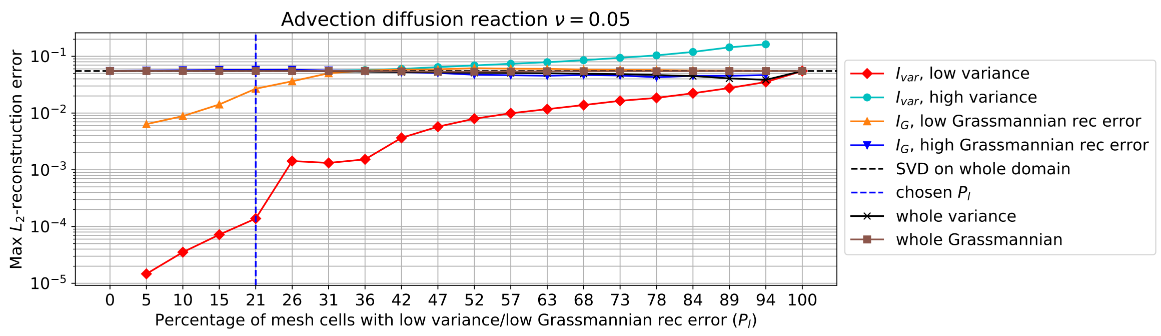


Fig. 5.18 **ADR.** Local relative L^2-reconstruction errors of the snapshots matrix of the advection diffusion reaction equation restricted to the two subdomains of the repartitioning performed with the indicator I_{var} (in red and light-blue), Definition 7, and I_{G} (in orange and blue), Definition 8. The relative L^2-reconstruction error attained on the whole domain is shown in black for the indicator I_{var} and in brown for the indicator I_{G}. The local reduced dimensions used to evaluate the local reconstruction errors is $n_{\Omega_i} = 3,\ i = 1,2$.

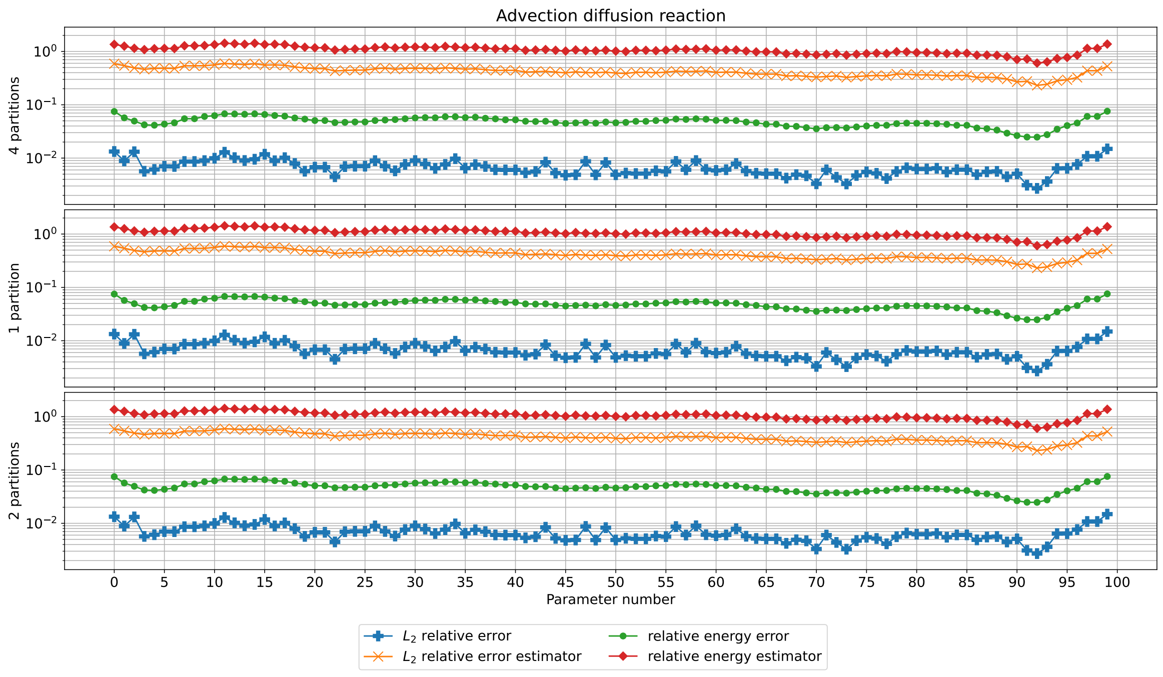


Fig. 5.19 **ADR.** Errors and estimators for advection diffusion reaction equation corresponding to the $n_{\mathrm{train}=20}$ uniformly sampled train snapshots corresponding to the abscissae $0,5,10,\dots,95$, and $n_{\mathrm{test}=80}$ uniformly sampled test snapshots, corresponding to the other abscissae. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [5,5,5,5]$ for $K = 4$ partitions, $r_\Omega = 5$ for $k = 1$ partition, and $\{n_{\Omega_i}\}_{i=1}^{k} = [2,5]$ for $k = 2$ partitions. For the case $k = 2$ we employed the cellwise variance indicator I_G, Definition 7, with $P_l = 21\%$.

Table 5.3 **ADR.** Average computational times and speedups for ROM and DD-ROM approaches for Maxwell equations. The speedup is computed as the FOM computational time over the ROM one. The FOM runs in parallel with 4 cores, so "FOM time" refers to wallclock time.

FOM		ROM			DD-ROM		
N_h	time	n	time	speedup	n_i	time	speedup
131328	3.243 [s]	5	79.112 [μs]	$\sim$ 40992	[5, 5, 5, 5]	59.912 [μs]	$\sim$ 54129

5, $i=1,\dots,K$ for $K=4$, $n_\Omega=5$ for the whole computational domain and $n_{\Omega_1}=2,\ r_{\Omega_2}=5$ for the repartitioned case with $k=2$, as mentioned.

In Table 5.3, we list the computational times and speedups for a simulation with the different methods.

5.6 Graph Neural Networks approximating Vanishing Viscosity solutions

In this section, we want to highlight how the well-known concept of vanishing viscosity solutions can be related to FS. In hyperbolic problems, the uniqueness of the weak solution is not guaranteed, already for very simple problems, e.g. inviscid Burgers' equations. In order to filter out the physically relevant solution, the concept of vanishing viscosity solution has been introduced, *inter alia* [104], and, consequently, vanishing viscosity methods have been developed, e.g. [81, 174].

We will consider the topic of vanishing viscosity solutions from the different perspectives of model order reduction. It is known that slow decaying Kolmogorov n-width solution manifolds result in ineffective linear reduced order models. The origin of this problem rests theoretically on the regularity of the parameter to solution map [58, 59], and with less generality on the nature of some PDEs (e.g. advection dominated PDEs, nonlinearities, complex dynamics), on the size of the parameter space, and on the smoothness of the parametric initial data or parametric boundary conditions [10], mainly. A possible way to obtain more approximable solution manifolds is through regularization or filtering [276, 265], e.g. adding artificial viscosity. Heuristically, the objective is to smoothen out the parametric solutions of the PDEs, for example removing sharp edges, local features, complex patterns, with the aim of designing more efficient ROMs for the filtered solution manifolds. Then, the linear ROMs will be applied to different levels of *regularization*, still remaining in the regime where they have good approximation properties. Finally, the original (vanishing viscosity) solutions will be recovered with a regression method from the succession of filtered linear ROMs. This is realized without the need to directly reduce with a linear reduced manifold the original solution manifold, thus avoiding the problem of its approximability with a linear subspace and the slow Kolmogorov n-width decay.

In our case, we consider regularization by viscosity levels: the vanishing viscosity solutions u_ν with viscosity $0 \le \nu \ll 1$, will be recovered as the limit $\lim_{i\to\infty} u_{\nu_i} = u_\nu$ of a potentially infinite succession of viscosity levels $\{\nu_i\}_{i=0}^{\infty}$, $\nu_0 > \nu_1 > \cdots > 0$, each associated to its efficient reduced order model. In practice, $\{\nu_i\}_{i=0}^{\infty} \approx \{\nu_i\}_{i=0}^{q}$, where q is the number of additional viscosity ROMs. It is clear the connection with multi-fidelity and super-resolution methods [97, 150]. The rationale of the approach is supported by the proofs of convergence to vanishing viscosity solutions of hyperbolic PDEs under various hypotheses [187, 152, 80, 107].

The framework is general and can be applied in particular to FS. We will achieve this for the advection–diffusion–reaction problem changing the viscosity constant $\mathbb{R} \ni \kappa > 0$ in (5.94). While this choice is specific to the model we are considering, a more general approach could consist in adding a viscous dissipative term to the generic FS obtaining another FS:

$$\begin{cases} Au = f + \boldsymbol{\kappa\Delta u}, & \text{in } \Omega \\ (\mathcal{D}-\mathcal{M})(u-g) = 0, & \text{on } \partial\Omega \end{cases} \to \begin{cases} \begin{cases} \kappa^{-1}\sigma + \nabla u = 0 \\ \nabla\cdot\sigma + Au = f \end{cases}, & \text{in } \Omega \\ (\mathcal{D}-\mathcal{M})(u-g) = 0, & \text{on } \partial\Omega, \end{cases} \tag{5.97}$$

recalling that the additional degrees of freedom are needed only for the high viscosity ROMs and FOMs (to collect the snapshots) and not the full-order vanishing viscosity solutions. This is only an example of how the procedure could be applied to any FS. In fact, the methodology is not designed specifically for FS.

The overhead of the methodology is related to the evaluation of the snapshots, the assembling of each level of viscosity $\{\nu_i\}_{i=0}^{q}$, and the computational costs of the regression method. We remark that the full matrices of the affine decomposition of each $\{ROM_{\nu_i}\}_{i=0}^{q}$ are the same. This is the price necessary to tackle the realization of reduced order models of parametric PDEs affected by a slow Kolmogorov n-width decay with our approach.

With respect to standard techniques for nonlinear manifold approximation, the proposed one is more interpretable as a mathematical limit of a succession of solutions to the vanishing viscosity one. Moreover, it has a faster training stage relying on the efficiency of the $\{ROM_{\nu_i}\}_{i=0}^{q}$. To the authors' knowledge, cheap analytical ways to obtain the vanishing viscosity solution from a finite succession of high viscosity ones are not available, so we will rely on data-driven regression methods.

5.6.1 Graph neural networks augmented with differential operators

Generally, machine learning (ML) architectures are employed in surrogate modelling to approximate nonlinear solution manifolds, otherwise linear subspaces are always preferred. The literature is vast on the subject and there are many frameworks that develop surrogate models with ML architectures. They promise to define data-driven reduced order models that infer solutions for new unseen parameters provided that there are enough data to train such architectures. This depends crucially on the choice of the encoding and inductive biases employed to represent the involved datasets: the training computational time and the amount of training data can change drastically.

On this matter, convolutional autoencoders (CNN) are one of the most efficient architectures to approximate nonlinear solution manifolds [161] for data structured on Cartesian grids, mainly thanks to their shift-equivariance property. For fields on unstructured meshes the natural choice are Graph neural networks (GNNs). Since their employment, GNNs architectures from the ML community have been enriched with physical inductive biases and other tools from numerical analysis. We want to test one of the first implementations and modifications of GNNs [245]. We also want to remark that in the literature, there are still very few test cases of ROMs that employ GNNs with more than ≥ 50000 degrees of freedom. The difficulty arises when the training is performed on large meshes, thus the need for tailored approaches.

The majority of GNNs employed for surrogate modelling are included in autoencoders [93, 203] or are directly parametrized to infer the unseen solution with a forward evaluation. These architectures may become heavy, especially for non-academic test cases. One way to tackle the problem of parametric model order reduction of slow Kolmogorov n-width solution manifolds is to employ GNNs only to recover the high-fidelity solution in a multi-fidelity setting, through super-resolution. Since efficient ROMs are employed to obtain the lower levels of fidelity (high viscosity solutions in our case), the solution manifold dimension reduction is performed only at those levels, avoiding the costly and heavy in memory training of autoencoders of GNNs.

We describe the implementation of augmented GNNs as in [245], with the difference that we need to train only a map from a collection of DD-ROMs solutions to the full-order vanishing viscosity solution, and not an autoencoder with pooling and unpooling layers to perform dimension reduction. The GNN we will employ is rather thin with respect to autoencoder GNNs used to perform dimension reduction. Its details are reported in Table 5.4.

We represent with

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}), \quad \mathcal{V} \in \mathbb{R}^{n_{\text{nodes}} \times f}, \quad \mathcal{E} \in \mathbb{N}^{n_{\text{edges}} \times 2}, \quad \mathcal{W} \in \mathbb{R}^{n_{\text{attr}} \times d}, \tag{5.98}$$

a graph with node features $\mathcal{V}$, edges $\mathcal{E}$ and edge attributes $\mathcal{W}$. The number f represents the nodal features dimension. We denote with $\mathbf{e}_{ij} = (i, j) \in \mathbb{N}^2$ the edge between the nodes $\mathbf{n}_i, \mathbf{n}_j \in \mathbb{R}^f$: $\mathbf{e}_{ij}$ corresponds to a row of $\mathcal{E}$, and $\mathbf{n}_i, \mathbf{n}_j$ correspond to the i-th and j-th rows of $\mathcal{V}$, for $i, j = 1, \dots, n_{\text{nodes}}$. Similarly, $\boldsymbol{\omega}_{ij}$ represents the edge attributes of edge $\mathbf{e}_{ij}$. We have $n_{\text{edges}} = n_{\text{attr}}$. For their efficiency, GNNs rely on a message-passing scheme composed of propagation and aggregation steps. Supposing that the graph is sparsely connected their implementation is efficient.

When the graph is supported on a mesh, it is natural to consider the generalized support points of finite element spaces as nodes of the graph and the sparsity pattern of the linear system associated to the numerical model as the adjacency matrix of the graph. We employ only Lagrangian nodal basis of discontinuous finite element spaces, but the framework can be applied to more general finite element spaces. As edge attributes $\boldsymbol{\omega}_{ij}$, we will employ the difference $\boldsymbol{\omega}_{ij} = \mathbf{x}_i - \mathbf{x}_j \in \mathbb{R}^d$ between the corresponding spatial coordinates associated to the nodes $\mathbf{n}_i, \mathbf{n}_j \in \mathbb{R}^f$. The nodes adjacent to node $\mathbf{n}_i$ are represented with the set $\mathcal{N}_{\text{neigh}(i)}$ for all $i = 1, \dots, n_{\text{nodes}}$.

We consider only the two following types of GNN layers: a continuous kernel-based convolutional operator l_{NNconv} [103, 232] and the GraphSAGE operator l_{SAGEconv} [118],

$$\mathcal{V}_{\text{out}} = l_{NNconv}(\mathcal{V}_{\text{inp}}, \mathcal{E}, \mathcal{W}) = \mathcal{V}_{\text{inp}} W_3 + \text{Avg}_1(\mathcal{V}_{\text{inp}}, h(\mathcal{W})) + \mathbf{b}_3, \quad h(\mathcal{W}) = \text{ReLU}(\mathcal{W} W_1 + \mathbf{b_1}) W_2 + \mathbf{b_2}, \tag{5.99}$$

$$\mathcal{V}_{\text{out}} = l_{SAGEconv}(\mathcal{V}_{\text{inp}}, \mathcal{E}, \mathcal{W}) = \mathcal{V}_{\text{inp}} W_6 + \text{Avg}_2(\text{ReLU}(\mathcal{V}_{\text{inp}} W_4 + \mathbf{b_4})) W_5 + \mathbf{b}_5, \tag{5.100}$$

with weight matrices dimensions,

$$W_1 \in \mathbb{R}^{2\times l},\ W_2 \in \mathbb{R}^{l\times(f_{\text{inp}}\times f_{\text{out}})},\ W_3 \in \mathbb{R}^{f_{\text{inp}}\times f_{\text{out}}},\ W_4 \in \mathbb{R}^{f_{\text{inp}}\times f_{\text{inp}}},\ W_5, W_6 \in \mathbb{R}^{f_{\text{inp}}\times f_{\text{out}}}, \tag{5.101}$$

$$\mathbf{b}_1 \in \mathbb{R}^{l},\ \mathbf{b}_2 \in \mathbb{R}^{(f_{\text{inp}}\times f_{\text{out}})},\ \mathbf{b}_3, \mathbf{b}_5 \in \mathbb{R}^{f_{\text{out}}},\ \mathbf{b}_5 \in \mathbb{R}^{f_{\text{inp}}}, \tag{5.102}$$

$$h(\mathcal{W}) \in \mathbb{R}^{n_{\text{edges}}\times(f_{\text{inp}}\times f_{\text{out}})}, \quad \mathcal{W} = \{W_s^h\}_{s=1}^{n_{\text{edges}}},\ W_s^h \in \mathbb{R}^{f_{\text{inp},s}\times f_{\text{out},s}},\ \forall s = 1,\dots,n_{\text{edges}}, \tag{5.103}$$

with the following average operators used as aggregation operators,

$$(\text{Avg}_1(\mathcal{V}, \{W_s^h\}_{s=1}^{n_{\text{edges}}}))_i = \frac{1}{\mathcal{N}_{\text{neigh}(i)}} \sum_{s\in\mathcal{N}_{\text{neigh}}(i)} W_s^h \mathbf{n}_s, \qquad (\text{Avg}_2(\mathcal{V}))_i = \frac{1}{\mathcal{N}_{\text{neigh}(i)}} \sum_{s\in\mathcal{N}_{\text{neigh}}(i)} \mathbf{n}_s, \qquad \forall i = 1,\dots,n_{\text{nodes}}, \tag{5.104}$$

where $\mathcal{V}_{\text{inp}} \in \mathbb{R}^{n_{\text{nodes}}\times f_{\text{inp}}}, \mathcal{V}_{\text{out}} \in \mathbb{R}^{n_{\text{nodes}}\times f_{\text{out}}}$ are the input and output nodes with feature dimensions $f_{\text{inp}}, f_{\text{out}}$. We remark that, differently from graph neural networks with heterogeneous layers, i.e., with changing mesh structure between different layers, in this network the edges $\mathcal{E}$ and edge attributes $\mathcal{W}$ are kept fixed, only the node features change. The feed-forward neural network $h : \mathbb{R}^{n_{\text{edges}}\times d} \to \mathbb{R}^{f_{\text{inp},s}\times f_{\text{out},s}}$ defines a weight matrix $W_s^h \in \mathbb{R}^{f_{\text{inp},s}\times f_{\text{out},s}}$ for each edge $s = 1,\dots,n_{\text{edges}}$. The number l is the hidden layer dimension of h.

The aggregation operators are defined from the edges $\mathcal{E}$ that are related to the sparsity pattern of the linear system of the numerical model. So, the aggregation is performed on the stencils of the numerical scheme chosen for every layer of the GNN architecture in Table 5.4. Many variants are possible, in particular, we do not employ pooling and unpooling layers to move from different meshes: we always consider the same adapted mesh.

Since our GNNs work on the nodal features, a good strategy is to augment their dimensions as proposed in [245]. In fact, in the majority of applications of GNNs for physical models the input features dimensions is the dimension of the physical fields considered and it is usually very small. Considering FS, the fields' dimension is m. To augment the input features, we will filter them with some differential operators discretized on the same mesh in which the GNN is supported. We consider the following differential operators

$$\Delta : V_h(\Omega) \to V_h(\Omega), \qquad \text{(Laplace operator)} \tag{5.105}$$

$$\mathbf{v}\cdot\nabla : V_h(\Omega) \to V_h(\Omega), \qquad \text{(Advection operator)} \tag{5.106}$$

$$\nabla_x : V_h(\Omega) \to V_h(\Omega), \qquad \text{(Gradient x-component)} \tag{5.107}$$

$$\nabla_y : V_h(\Omega) \to V_h(\Omega), \qquad \text{(Gradient y-component)} \tag{5.108}$$

for a total of four possible feature augmentation operators, where, in our case, $\mathbf{v}$ is the advection velocity from the incompressible Navier-Stokes equations (5.96). We employ the representation of the previous differential operators with respect to the polynomial basis of Lagrangian shape functions, so they act on the vectors of nodal evaluations in $\mathbb{R}^{N_h}$. As in [245], we consider three sets of possible augmentations:

$$\mathcal{O}_1 = \{\mathbb{I}_{N_h}, \Delta, \mathbf{v}\cdot\nabla, \nabla_x, \nabla_y\}, \tag{5.109}$$

$$\mathcal{O}_2 = \{\mathbb{I}_{N_h}, \nabla_x, \nabla_y\} \tag{5.110}$$

$$\mathcal{O}_3 = \{\mathbb{I}_{N_h}\} \tag{5.111}$$

where $\mathbb{I}_{N_h}$ is the identity matrix in $\mathbb{R}^{N_h}$, $|\mathcal{O}_1| = 5 = n_{\text{aug}}$, $|\mathcal{O}_2| = 3 = n_{\text{aug}}$ and $|\mathcal{O}_3| = 1 = n_{\text{aug}}$. We will reconstruct only the scalar concentration u with the GNN, so, in our case, the field dimension is 1, which is the output dimension. The input dimension depends on the number of high viscosity DD-ROMs employed that we denote with q. Given a single parametric instance $\boldsymbol{\rho} \in \mathbb{R}^P$ the associated solutions of $\{\text{D-ROM}_{\kappa_i}\}_{i=1}^q$ are $\{\mathbf{u}^{\text{RB}}(\boldsymbol{\rho}_i)\}_{i=1}^q$.

We divide the snapshots $\{\mathbf{u}^{\text{RB}}(\boldsymbol{\rho}_i)\}_{i=1}^{n_{\text{train}}+n_{\text{test}}}$ in training $\{\mathbf{u}^{\text{RB}}(\boldsymbol{\rho}_i)\}_{i\in I_{n\text{train}}}$ and test snapshots $\{\mathbf{u}^{\text{RB}}(\boldsymbol{\rho}_i)\}_{i\in I_{n\text{test}}}$, with $|I_{\text{train}}| = n_{\text{train}}$ and $|I_{\text{test}}| = n_{\text{test}}$. We have decided to encode the reconstruction of the vanishing viscosity solution $\mathbf{u}_{q+1}$ learning the difference $\mathbf{u}_{q+1}(\boldsymbol{\rho}) - \mathbf{u}_q^{\text{RB}}(\boldsymbol{\rho}) - \overline{\mathbf{u}_{q+1}}_{\text{train}}$ with the mesh-supported-augmented GNN (MSA-GNN) described in Table 5.4:

$$\mathbf{u}_{q+1}(\boldsymbol{\rho}) = ReLU\left(\mathbf{u}_q^{\text{RB}}(\boldsymbol{\rho}) + \text{MSA-GNN}\left(\{O_a\{\{\mathbf{u}_i^{\text{RB}}(\boldsymbol{\rho})\}_{i=1}^q, \{\mathbf{u}_i^{\text{RB}}(\boldsymbol{\rho}) - \mathbf{u}_{i-1}^{\text{RB}}(\boldsymbol{\rho})\}_{i=1}^{q-1}\}\}_{a=2}^{n_{\text{aug}}}\right) + \overline{\mathbf{u}_{q+1}}_{\text{train}}\right), \tag{5.112}$$

where $\overline{\mathbf{u}_{q+1}}_{\text{train}} = \frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}} \mathbf{u}_{q+1}(\boldsymbol{\rho}_i)$. Learning the difference instead of the solution itself helps in getting more informative features. The input dimension is therefore $3n_{\text{aug}} = 15$ for $\mathcal{O}_1$ and $3n_{\text{aug}} = 9$ for $\mathcal{O}_2$.

5.6.2 Decomposable ROMs approximating vanishing viscosity (VV) solution through GNNs

In this section, we test the proposed multi–fidelity approach that reconstructs the lowest viscosity level with the GNN. We consider the FS (5.94), with three levels of viscosity: from highest to lowest $\kappa_1 = 0.05$, $\kappa_2 = 0.01$ and $\kappa_3 = 0.0005$. We want to build a surrogate model that efficiently predicts

Table 5.4 Mesh supported augmented GNN

Net	Weights $[f_{\text{inp}}, f_{\text{out}}]$	Aggregation	Activation
Input NNConv	$[3n_{\text{aug}}, 18]$	Avg_1	ReLU
SAGEconv	[18, 21]	Avg_2	ReLU
SAGEconv	[21, 24]	Avg_2	ReLU
SAGEconv	[24, 27]	Avg_2	ReLU
SAGEconv	[27, 30]	Avg_2	ReLU
Output NNcrshortNNConv	[30, 1]	Avg_1	-

NNConvFilters	First Layer $[2, l]$	Activation	Second Layer $[l, f_{\text{inp}} f_{\text{out}}]$
Input NNConv	[2, 12]	ReLU	$[12, 3n_{\text{aug}} \cdot 18]$
Output NNConv	[2, 8]	ReLU	[8, 30]

the parametric solutions of the FS (5.94) for unseen values of $\boldsymbol{\rho} \in \mathbb{P}$ with fixed viscosity $\kappa_3 = 0.0005$. These solutions will be referred to as *vanishing viscosity* solutions. The other two viscosity levels are employed to build the DD-ROM$_{\kappa_1}$ and DD-ROM$_{\kappa_2}$ with viscosities $\kappa_1 = 0.05$ and $\kappa_2 = 0.01$, respectively. The parametrization affects the inflow boundary condition and is the same as the one described in section 5.5.3, see equation (5.95). We also employ the same number of training 20 and test 80 parameters.

The DD-ROMs provided for $\kappa_1 = 0.05$ and $\kappa_2 = 0.01$ can be efficiently designed with reduced dimensions $\{r_{\Omega_i}\}_{i=1}^{K}[5,5,5,5]$. To further reduce the cost, we employ an even coarser mesh for ROM$_{\kappa_1}$ and ROM$_{\kappa_2}$ and a finer mesh for the vanishing viscosity solutions. The former is represented on the left of Figure 5.20, the latter on the right. The degrees of freedom related to the coarse mesh are 43776, while the ones on the fine one are 175104.

For the training of the GNN we use the open source software library PyTorch Geometric [89]. The employment of efficient samplers that partition the graphs on which the training set is supported is crucial to lowering the otherwise heavy memory burden [118]. We preferred samplers that partition the mesh with METIS [143] as it is often employed in this context. We decided to train the GNN with early stopping at 50 epochs as our focus is also in the reduction of the training time of the NN architectures used for model order reduction. It corresponds on average to less than 60 minutes of training time. The batch size is 100 and we clustered the whole domain in 100000 subgraphs in order to fit the batches in our limited GPU memory. Each augmentation strategy and additional fidelity level, do not affect the whole training time as they only increase the dimension of the input features from a minimum of 1 (1 fidelity, no augmentation) to a maximum of 15 (all augmentations $\mathcal{O}_3$, 2 fidelities). As optimizer, we use ADAM [147] stochastic optimizers. Every architecture is trained on a single GPU NVIDIA Quadro RTX 4000.

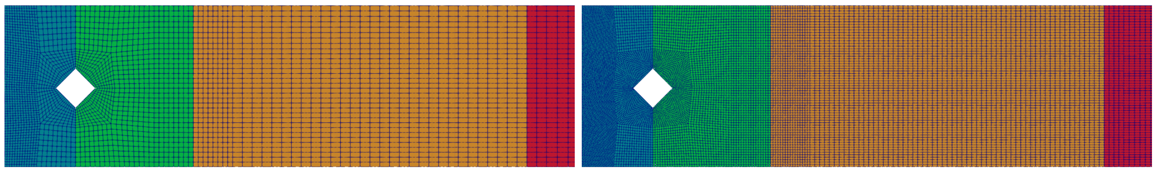


Fig. 5.20 **VV.** Left part of the computational domain partitioned in 4 for distributed parallelism: coarse mesh (**left**), fine mesh (**right**). The solution with viscosity $\kappa \in \{\mathbf{0.05}, \mathbf{0.01}\}$ are evaluated on the coarse mesh with 4868 cells and 43776 dofs, those with $\kappa = \mathbf{0.0005}$ on the finer with 19456 cells and 175104 dofs.

Figures 5.21, 5.22 and 5.23 show the results of the algorithm for parameters with index $i \in \{0, 50, 99\}$. In particular, we show on the left columns the FOM simulations, in the center column the ROM simulations and the error in the right column. Moreover, in the different rows, we have different viscosity levels. The first three rows use the classical DD-ROM approach. We can immediately see that the vanishing viscosity $\kappa = \kappa_3 = 0.0005$ level shows strong numerical oscillations along the whole solution, which are not present in the FOM method. This phenomenon is observable also for higher viscosity levels but it is less pronounced and concentrated on the left of the domain, where the discontinuities are imposed as boundary conditions (see error plots). Finally, in the last row, we see the GNN approach, which uses the first two viscosity levels to predict the vanishing viscosity one. Contrary to DD-ROM, we do not observe many numerical oscillations in the reduced solutions and they are much more physically meaningful. Thinking about extending this approach to more complicated problems, such as Euler's equations, one could guarantee the presence of the correct amount of shocks and the right location or maintaining the positivity of density and pressure close to discontinuities.

In Figure 5.24, we show a quantitative measure of the error of the reduced approaches presented in terms of relative L^2 error. Overall, we can immediately see that the new GNN approach can always reach errors of the order of 10^{-3} for the vanishing viscosity solutions, with few peaks in the extrapolatory regime of $8 \cdot 10^{-3}$, while the classical DD-ROM on the vanishing viscosity solutions perform really poorly with errors around 10%. On the other hand, the DD-ROM for higher viscosity levels have lower errors around 3% for κ_2 and 0.5% for κ_1, hence, they are still reliably representing those solutions.

On the different GNN approaches, in Figure 5.24 at the top we compare the different augmentations $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ and how many levels of viscosity we keep into considerations to derive the vanishing viscosity solution. The usage of multiple fidelity levels (two viscosity levels) is a great improvement for all the augmentations proposed and it can make gain a factor of 2 in terms of accuracy. There are slight differences with the used augmentations and, in particular, we observe that the $\mathcal{O}_1$ augmentation, with all operators, guarantees better performance, while there are no appreciable differences between $\mathcal{O}_2$ and $\mathcal{O}_3$. Clearly, one could come up with many other augmentation possibilities by choosing more operators, but at the cost of increasing the dimensions of the GNN and the offline training costs. We

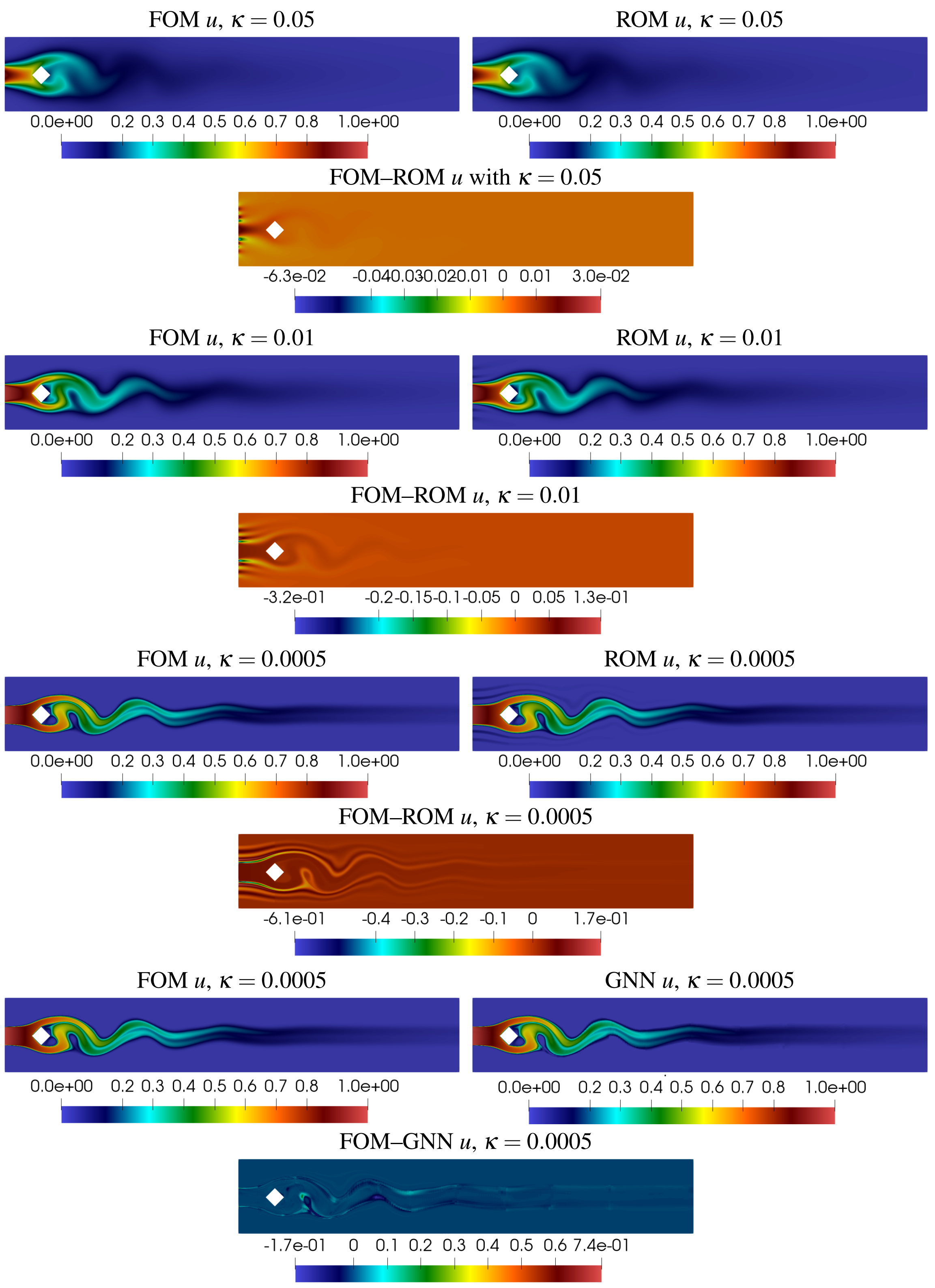


Fig. 5.21 **VV.** Scalar concentration advected by incompressible flow for $i = 0$. Comparison of ROM approach at different viscosity levels $\kappa \in \{0.05, 0.01, 0.0005\}$ and GNN for $\kappa = 0.0005$. FOMs on the left, reduced solution at the center and error on the right.

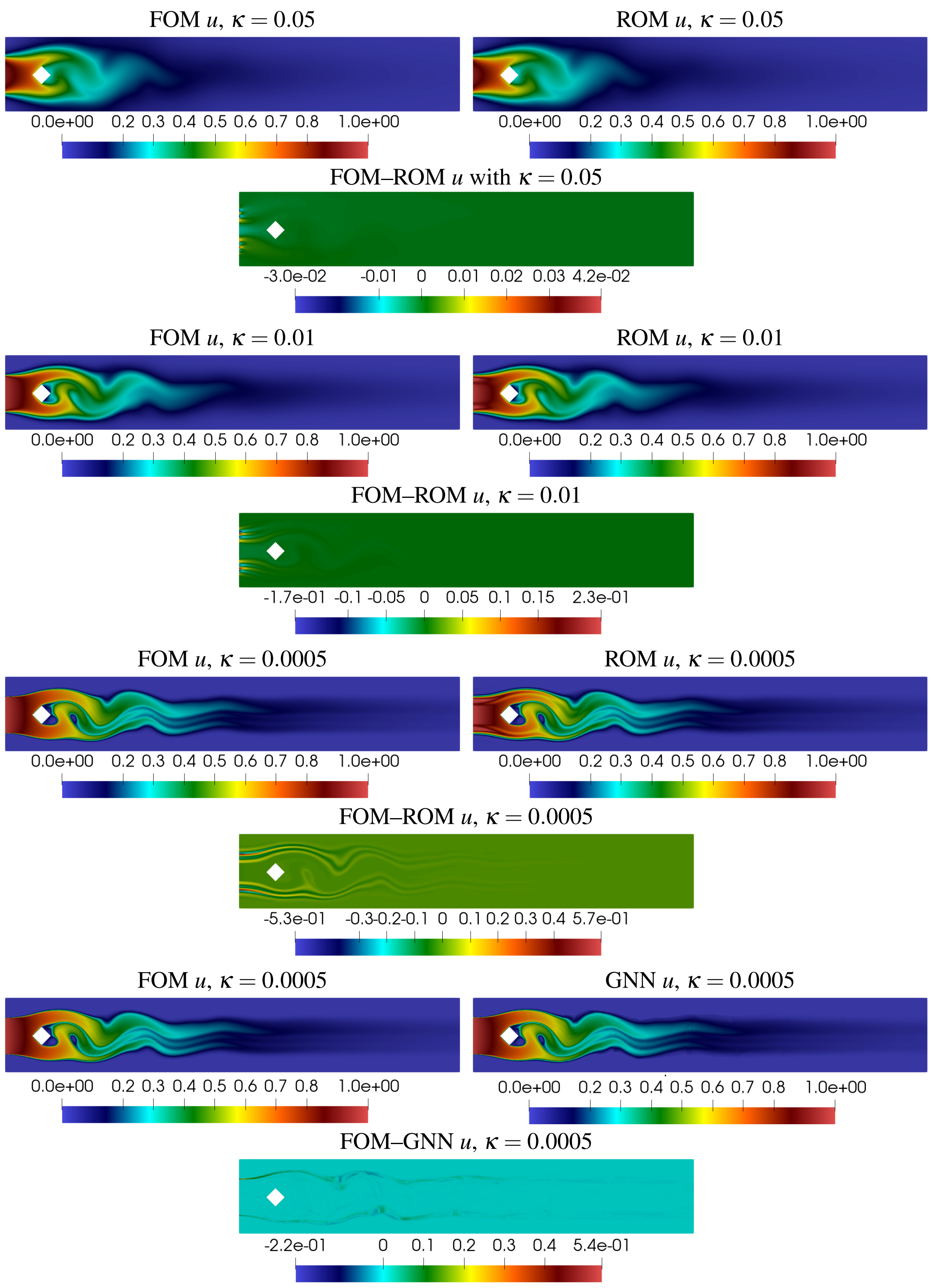


Fig. 5.22 **VV.** Scalar concentration advected by incompressible flow for $i = 50$. Comparison of ROM approach at different viscosity levels $\kappa \in \{0.05, 0.01, 0.0005\}$ and GNN for $\kappa = 0.0005$. FOMs on the left, reduced solution at the center and error on the right.

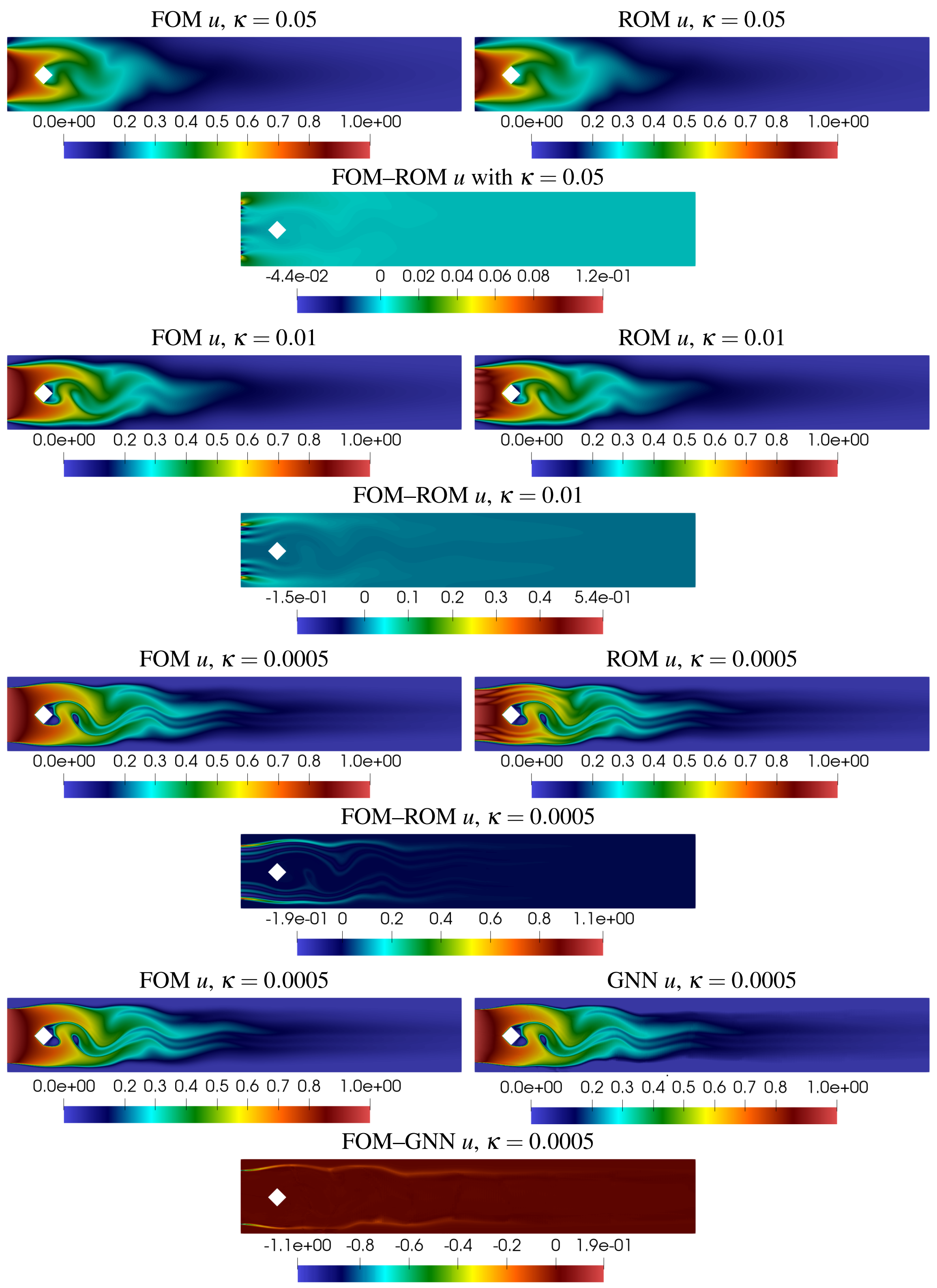


Fig. 5.23 **VV.** Scalar concentration advected by incompressible flow for $i = 99$. Comparison of ROM approach at different viscosity levels $\kappa \in \{0.05, 0.01, 0.0005\}$ and GNN for $\kappa = 0.0005$. FOMs on the left, reduced solution at the center and error on the right.

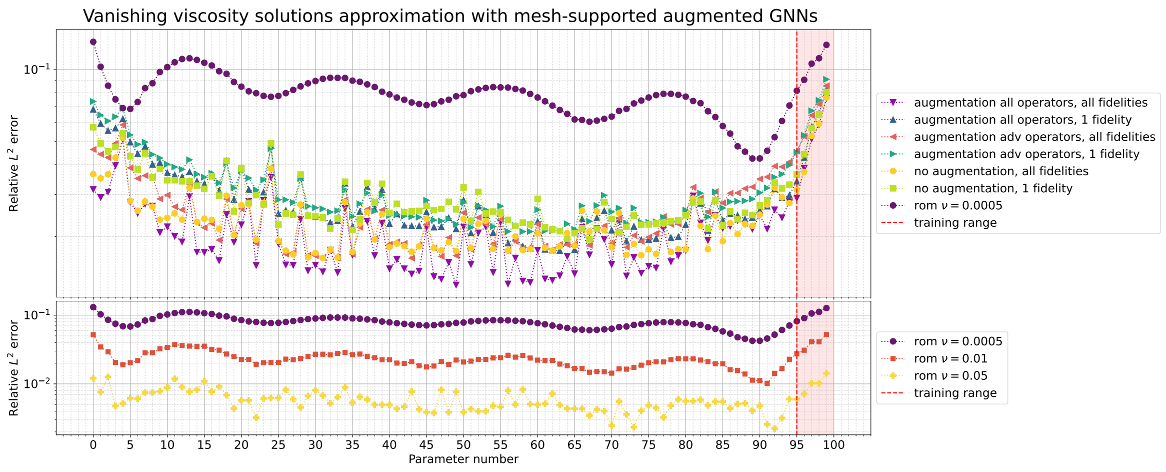


Fig. 5.24 **VV.** Relative errors for the scalar conservation advected by incompressible flow problem. The parameters corresponding to the snapshots used for the GNNs and DD-ROMs training correspond to the abscissae $0, 5, 10, \dots, 95$ the rest are test parameters. The dashed red background highlights the extrapolation range. **Top:** errors on train and test set with different GNN approaches given by the three augmentation $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ and by using either 1 viscosity level (1 fidelity) or 2 (all fidelities). **Bottom:** errors for DD-ROM approaches at different viscosity levels. The reduced dimensions of the ROMs are $\{n_{\Omega_i}\}_{i=1}^{K} = [5,5,5,5]$ with $K = 4$ partitions.

believe that all the presented options already perform much better with respect to classical approaches and can already be used without further changes.

In Table 5.5, we compare the computational times necessary to compute the FOM solutions, the DD-ROM ones, the training time for the GNN and the online costs of the GNN. As mentioned, we employ only one GPU NVIDIA Quadro RTX 4000 with 8GB of memory. Typical GNNs applications that involve autoencoders to perform nonlinear dimension reduction are much heavier. The training time of the GNNs for the different choice of augmentation operators vary between 48 minutes and 60 minutes approximately. We believe that shortly more optimized implementations will reduce the training costs of GNNs. The computational time of the evaluation of a single forward of the GNN is on average 2.661 seconds but vectorization ensures the evaluation of multiple online solutions altogether: with our limited memory budget, we could predict all the 100 training and test snapshots with just 2 batches of 50 stacked inputs each. The "Total online time" computed as previously described is 17.166 seconds that is 171.66 milliseconds per online solution with a speedup of around 56 with respect to the 9.668 seconds for the FOM.

5.7 Conclusions

We argue that Friedrichs' systems represent a valuable framework to study and devise reduced order models of many parametric PDEs at the same time: among them the ones studied in this work and others, like mixed elliptic and hyperbolic problems, complex and time-dependent FS and also

Table 5.5 **VV.** Computational costs for scalar advected by an incompressible flow problem with GNNs approximating vanishing viscosity solutions (VV). The speedup is computed as the FOMs computational time over the ROM one. The speedup of the GNN is with respect to the FOM with viscosity $\nu = 0.0005$. The FOM runs in parallel with $K = 4$ cores as the DD-ROMs, so "FOM time" and "DD-ROM time" refers to wallclock time. Regarding the GNN results, "GNN forward" refers to a single online evaluation while "Online" refers to the evaluation of the 100 training and test snapshots altogether with only two separate GNN forward evaluations with batches of 50 inputs each. The speedup is evaluated as "FOM time" over "Total online time" divided by 100.

	FOM		DD-ROM			
κ	N_h	time	n_i	time	speedup	mean L^2 error
0.05	43776	3.243 [s]	[5, 5, 5, 5]	59.912 [μs]	54129	0.00595
0.01	43776	3.236 [s]	[5, 5, 5, 5]	79.798 [μs]	40552	0.0235
0.0005	175104	9.668 [s]	[5, 5, 5, 5]	95.844 [μs]	100872	0.0796

κ	GNN training time	GNN forward	Online	GNN speedup	mean L^2 error
0.0005	≤ 60 [min]	2.661 [s]	17.166 [s]	56	0.0217

nonlinear PDEs whose linearization results in FS, e.g. the Euler equations. The advantages include the availability of *a posteriori* error estimators and the easy to preserve mathematical properties of positivity and symmetry from the full-order formulations to the reduced-order ones. We underlined in section 5.4.2 how optimally stable reduced-order models can be obtained from the ultraweak formulation. A more efficient numerical solver for Friedrichs' systems is the hybridized discontinuous Galerkin method [49]. These are possible future directions of research.

Working with discontinuous Galerkin discretizations is not only crucial from the possibly mixed elliptic and hyperbolic nature of Friedrichs' systems, but also to design domain decomposable reduced-order models with a minimum effort: in fact, penalties at the subdomains interfaces are inherited directly from the full-order models. We demonstrated with numerical experiments the limits and the ranges of application of domain decomposable ROMs: generally, with respect to single domain ROMs, there are benefits only when the model under study is truly decomposable, that is when the parameters affect independently different subdomains and the respective solutions are poorly correlated for unseen parametric instances. The results we showed in our academic benchmarks were obtained with the aim to tackle more complex multi-physics models like fluid-structure interaction systems. A typical application of DD-ROMs for FS is represented by parametric PDEs with a mixed elliptic and hyperbolic nature and possibly solution manifolds more and less linearly approximable respectively. The repartitioning strategies we developed are suited to adapt the reduced local dimension of the linear approximants, especially when the parameters influence only a limited region like in test case ADR 5.5.3. The implementation of *ad hoc* physics inspired indicators can be a future direction of research.

The Friedrichs' systems formulation itself does not solve the problems caused by a slow decaying Kolmogorov n-width. DD-ROMs can help in this regard, isolating regions with a slow Kolmogorov n-width for which nonlinear approximants can be employed and regions with a fast decaying Kolmogorov n-width for which classical linear projection-based ROMs provide efficient and reliable predictions. Related to this subject and motivated also by the heavy computational resources that graph neural networks require when employed for model order reduction, we introduced a new paradigm for surrogate modelling: the inference with GNNs of vanishing viscosity solutions from a succession of higher viscosity projection-based ROMs. The approach is, of course, general and can be applied to PDEs that are not FS. The crucial hypotheses underneath this approach is the approximability with linear spaces of the solution manifolds corresponding to higher viscosity levels. We showed that the additional computational costs are not too large in our test case in section 5.6. Possible directions of research include more complex problems and different regularization or filtering choices, other than additional viscous terms.

Chapter 6

Hyper-reduced nonlinear manifold method: teacher-student training

Non-affine parametric dependencies, nonlinearities and advection-dominated regimes of the model of interest can result in a slow Kolmogorov n-width decay, which precludes the realization of efficient reduced-order models based on linear subspace approximations. Among the possible solutions, there are purely data-driven methods that leverage autoencoders and their variants to learn a latent representation of the dynamical system, and then evolve it in time with another architecture. Despite their success in many applications where standard linear techniques fail, more has to be done to increase the interpretability of the results, especially outside the training range and not in regimes characterized by an abundance of data. Not to mention that none of the knowledge on the physics of the model is exploited during the predictive phase. In order to overcome these weaknesses, it is implemented the nonlinear manifold method introduced by Carlberg et al with hyper-reduction achieved through reduced over-collocation and teacher-student training of a reduced decoder. The methodology is tested on a 2d nonlinear conservation law and a 2d shallow water models, and compare the results obtained with a purely data-driven method for which the dynamics is evolved in time with a long-short term memory network.

Contents

6.1 Literature review

In real world engineering scenarios, when performing outer loop applications such as optimization, uncertainty quantification, sensitivity analysis, parametric partial differential equations (PDEs) often need to be solved numerically numerous times. However, relying on the mathematical properties of some parametric PDEs, the computational cost for many query problems can be drastically reduced taking into account previous results on a set of training parameters: the procedure for the design of reduced-order models (MORs) is divided in an offline (training) stage, during which a set of training solutions is collected, and an online (testing or predictive) stage, which employs the compressed information from the previous step to predict the solutions of the PDE of interest for unseen parameters. This reduction is performed numerically defining a low-dimensional global basis devised in the offline stage, and can be carried out independently of the class of numerical methods chosen: finite element (FEM), spectral element (SEM), discontinuous Galerkin (DGM), and finite volumes method (FVM). One of the most employed model-order reduction method (MOR) is the reduced basis method [126, 210].

Depending on the parametric dependency and mathematical nature of some PDEs, various issues may occur: the Kolmogorov n-width (KnW) is used to characterize the approximability of the solution manifold, that is the set of parameter-dependent solutions of the PDE, by a linear trial subspace. A slow decaying KnW is a symptom of the difficulties in the design of efficient ROMs: this results in the necessity of using a high number of reduced basis or proper orthogonal decomposition (POD) method's modes, corrupting the efficiency of the ROMs till the point that the gain into the computational cost becomes irrelevant. One class of PDEs where this behaviour is evident are time-dependent advection-dominated PDEs. Moreover, nonlinear PDEs require hyper-reduction procedures to make the reduced equations independent of the number of degrees of freedom of the full-order model (FOM).

Recently, leveraging machine learning's advances in manifold learning, a class of ROMs that employ a nonlinear trial manifold built with convolutional autoencoders (CAEs) [106] was developed by Carlberg et al [161]. One of the benefits of this approach is the possibility to employ a small latent dimension of the ROMs, thus overcoming the slow decay of the KnW for some parametric PDEs, at the expense of introducing additional nonlinearities from the neural networks (NNs) and sometimes more substantial training costs in the offline stage. The properties of the nonlinear manifold methods include the need of less stabilization mechanisms, the less intrusiveness on the FOM solvers — they are in fact equations-based rather than fully-intrusive — and the possibility to apply them for a much broader class of parametric PDEs, differently from ROMs devised specifically for advection-dominated problems.

A hyper-reduction scheme for nonlinear manifold Least-Squares Petrov-Galerkin (NM-LSPG) and nonlinear manifold Galerkin (NM-G), is introduced in [146]: it relies on Gauss-Newton tensor approximation (GNAT) [46] hyper-reduction method and shallow masked autoencoders to select only the degrees of freedom that explain the dynamics and therefore restrict efficiently the decoder and the discretized residuals. As we will see in our test cases, the reconstruction error of the autoencoder employed empirically bounds from below the errors of all the other nonlinear manifold ROMs built upon. Therefore, we devise a method that is independent on the choice of the architecture: a sparse shallow autoencoder is not necessary anymore, and any NN architecture, like CAEs, could be in principle employed. This frees the way to imposing additional inductive biases that help to speedup the offline stage and to achieve accurate approximations of the discrete solution manifolds, a crucial requirement. Moreover, in some cases, reconstructing the residuals with GNAT is not efficient, still because of a slow decaying KnW, so we choose to employ the reduced over-collocation hyper-reduction method (ROC) [51]: in this case the equation's numerical residual is not reconstructed on a global basis, but collocated on some nodes of the mesh called collocation points.

Once the CAE reaches a satisfactory approximation of the discrete solution manifold, purely data-driven NN PDE can be trained to predict the latent dynamics: the gold standard that is being established for this task are long-short term memory networks (LSTM). Their online computational cost is low even w.r.t linear ROMs, but some new issues appear: their accuracy depends on the regularity of the latent dynamics, especially when predicting the solutions for parameters outside the training range, in the extrapolation regimes; they require hyperparameters tuning, and all the connections to the PDEs model are completely lost, resulting in a loss of interpretability of the results. The nonlinear manifold hyper-reduced ROMs we develop solve these issues, at the expense of a higher computational cost in the online stage, since at each time step a physics-based residual is minimized. Moreover, a posteriori error estimates are available [161, 146]. In our test cases, we compare these two approaches to enlight their differences, weak and strong points.

The structure of this paper is as follows. In section 6.2 we delve into the topic of manifold learning which has many connections with reduced-order modelling, especially since the recent entry of machine learning in the design of ROMs. We will proceed introducing the Kolmogorov n-with (KnW), and we will show that some classes of parametric PDEs suffer from the so called slow decaying Kolmogorov n-width. In section 6.3, the nonlinear manifold (NM) reduced-order models based on the work of Carlberg et al [161] are introduced. We will focus on the nonlinear manifold least-squares Petrov-Galerkin method (NM-LSPG). Afterwords, we describe our new hyper-reduced ROMs: NM-LSPG with reduced over-collocation (NM-LSPG-ROC) and NM-LSPG with reduced over-collocation and teacher-student training of a compressed decoder (NM-LSPG-ROC-TS). These two approaches, to the best of the authors' knowledge, are introduced here for the first time. In section 6.4, the new model order reduction (MOR) methods are tested on a 2d parametric nonlinear time-dependent conservation law model and a 2d parametric nonlinear time-dependent shallow water equations model. In section 6.5, a discussion on the results obtained follows, and in the conclusive section 6.6 possible future directions of research are explored.

6.2 Manifold learning

The subject of manifold learning, classified as a topic of machine learning, had its unique flavour in model order reduction even before nowadays breakout of scientific machine learning [15]. The workhorse of the model order reduction community is POD or SVD. A lot of real applications though, required new methods to approximate the solution manifold in a nonlinear fashion. The symptom of this behaviour is a slow decaying Kolmogorov n-width. Some approaches rely on the locality of the validity region of a linear approximation with POD, both in the parameter space and in the spatial and temporal domains, others implement nonlinear dimensionality reduction methods from machine learning [182]: kernel principal component analysis (KPCA), Isomap, clustering algorithms. Nonlinear MORs include approximations by rational functions, splines or other nonlinear functions collected in a dictionary [25].

Interpolatory approaches of the solution manifold with respect to the parameters have been developed, sometimes combined with nonlinear dimension reduction techniques like KPCA and its variants: interpolation with geodesics on the Grassmann manifold [7], interpolation on the latent space obtained with Isomap dimensionality reduction method [94, 29], interpolation with optimal transport [26], dictionary-based ROM that make use of clustering in the Grassmannian manifold and classification with neural networks (NN) [79], local kernel principal component analysis [163]. At the same time, domain decomposition approaches tackled locality in space [169, 38].

One particular class of dimension reduction techniques is represented by autoencoders, and more generally by other architectures that rely on NNs. In the recent literature many achievements are brought by CAEs, and by extension by Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Bayesian convolutional autoencoders [106]: in [181] convolutional autoencoders are utilized for dimensionality reduction and long-short Term Memory (LSTM) NNs or causal convolutional neural networks are used for time-stepping; in [95] the evolution of the dynamics and the parameter dependency is learned at the same time of the latent space with a forward NN and a CNNs on randomized SVD compressed snapshots, respectively; and in [278] spatial and temporal features are separately learned with a multi-level convolutional autoencoder.

In order to extend these architectures to datasets not structured in orthogonal grids, geometric deep learning [34] is called to the task. There are not many works on geometric deep learning applied to model order reduction for mesh-based simulations that achieve the same accuracy of CNNs, yet. Promising results are reached by an architecture that employs graph neural networks (GNNs) and a physics-informed loss [201]. In the future, the potential of GNNs will probably be leveraged extending the range of applicability of nowadays frameworks.

The setting we will base our studies on, does not depend directly on the numerical method used to discretize the parametric PDEs at hand (FVM, FEM, SEM, DGM), so the mathematical formulation will generically be founded on models represented by a parametric system of time-dependent (but also time-independent) PDEs, consisting of a nonlinear parametric differential operator $\mathcal{G}$ and of the boundary differential operators $\mathcal{B}, \mathcal{B}_0$ that represent the boundary and initial conditions respectively,

$$\forall \boldsymbol{\mu} \in \mathcal{P} \qquad \begin{cases} \mathcal{G}(\boldsymbol{\mu}, U(\boldsymbol{\mu}, t, \mathbf{x})) & = 0 \qquad (\mathbf{x}, t) \in \Omega \times [0, T], \\ \mathcal{B}(\boldsymbol{\mu}, U(\boldsymbol{\mu}, t, \mathbf{y})) & = 0 \qquad (\mathbf{y}, t) \in \partial\Omega \times [0, T], \\ \mathcal{B}_0(\boldsymbol{\mu}, U(\boldsymbol{\mu}, 0, \mathbf{x})) & = 0 \qquad \mathbf{x} \in \Omega, \end{cases} \tag{6.1}$$

where $\mathcal{P}$ is the parameter space, U are the state variables and Ω is the 2D or 3D spatial domain. This formulation includes also coupled systems of PDEs. We assume that the solutions belong to a certain Banach or Hilbert space Y, varying $(\boldsymbol{\mu}, t) \in \mathcal{P} \times [0, T]$. The solution manifold $\mathcal{M}$ is represented by the set

$$\mathcal{M} = \{U(\boldsymbol{\mu}, t) \in Y \mid \boldsymbol{\mu} \in \mathcal{P}, t \in [0, T]\}. \tag{6.2}$$

6.2.1 Approximability by n-dimensional subspaces and Kolmogorov n-width

We want to remark some results available in the literature, in order to state and comment, for our needs, the problem of solution manifold approximability. In particular, our benchmarks belong to a class of parametric PDEs for which the Kolmogorov n-width decays slowly. Thus, classical Petrov-Galerkin projection with POD needs to be overcome with nonlinear methods in place of POD to achieve efficient ROMs.

Let $(X, \|\cdot\|)_X$ be a complex Banach space, and $K \subset X$ a compact subspace, the Kolmogorov n-width (KnW) of K in X is defined as

$$d_n(K)_X = \inf_{\dim(W)=n} \max_{v \in K} \min_{w \in W} \|v - w\|_X. \tag{6.3}$$

Let $(Y, \|\cdot\|_Y)$ be a complex Banach space and $L : K \subset\subset X \to Y$. In our framework K is the parameter space, possibly infinite dimensional and L is the solution map of the system of parametric PDEs at hand, from the parameter space to the solution manifold. In order to define L we have to suppose that for each parameter in K there is a unique solution in Y.

Following [59], it can be proved that for holomorphic L, thus not necessarily linear, the Kolmogorov n-width decay is one polynomial order below the Kolmogorov n-width of the parameter space K.

Theorem 6 (Theorem 1, [59]). *Suppose u is a holomorphic mapping from an open set $O \subset X$ into Y and u is uniformly bounded on O,*

$$\exists B > 0, \quad \sup_{\mathbf{x} \in O} \|u(x)\|_Y \leq B. \tag{6.4}$$

If $K \subset O$ is a compact subset of X, then for any $s > 1$ and $t < s - 1$,

$$\sup_{n \geq 1} n^s d_n(K)_X < \infty \Rightarrow \sup_{n \geq 1} n^t d_n(u(K))_K < \infty. \tag{6.5}$$

In particular, if the hypothesis of the previous theorem are satisfied and if K is a finite dimensional linear subspace, the Kolmogorov n-width decay is exponential. In general, elliptic PDEs, affinely decomposable with respect to the parameters, satisfy the hypothesis of the previous theorem [13, 155]. Not always nonlinearities cause a slow decaying KnW: using theorem 6, in [59] they prove that the parametric PDE on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^3$

$$u^3 - \nabla \cdot (\exp a) \nabla u = f, \tag{6.6}$$

$$K = \{a \in L^\infty(\Omega) : \|a\|_{C^\alpha} \leq M\}, \quad f \in H^{-1}(\Omega), \tag{6.7}$$

with homogeneous Dirichlet boundary conditions, where C^α is the space of Hölder functions, satisfies the hypothesis of the previous theorem. Actually, for Hölder functions the KnW is bounded above by $n^{-\alpha/3}$, which is not a fast convergence, but if instead a belongs to the Sobolev space $W^{s,\infty}(\Omega)$ then the upper bound is $n^{s/3}$ [99]. We will consider a good KnW decay if it has a higher infinitesimal order than n^{-1}.

The same is not valid in the case of the simplest linear advection problem. We briefly report some results from the literature on classical hyperbolic PDEs, for $(t, \mu) \in K = [0, 1]^2$ with the standard norm and $Y = L^2([0, 1])$,

$$L : (t, \mu) \mapsto u(t, x, \mu) \quad \text{s.t.} \quad \partial_t u - \mu \partial_x u = 0 \qquad d_n(\mathcal{M})_{L^2} > n^{-\frac{1}{2}}, \tag{6.8}$$

$$L : (t, \mu) \mapsto u(t, x, \mu) \quad \text{s.t.} \quad \partial^2_{tt} u - \mu \partial^2_{xx} u = 0 \qquad d_n(\mathcal{M})_{L^2} > n^{-\frac{1}{2}}, \tag{6.9}$$

where, the results are respectively proven in [185] and [110].

This behaviour is not restricted only to advection-dominated problems. Intuitively also solution manifolds that are characterized by a parametrized locality in space suffer from slow decaying KnW, like elliptic problems with singular sources parametrically moving in the domain [92].

Our newly developed ROMs, should solve the issue of slow decaying KnW in the applications, guaranteeing a low latent or reduced dimension of the approximate solution manifold. This, because the linear trial manifold, frequently generated by the leading POD modes, is substituted with a nonlinear trial manifold, parametrized by the decoder of an autoencoder. The test cases we present in section 6.4, were chosen in order to be advection-dominated and particularly not suited for linear

ROMs, as shown in [146] for the 2d Burgers' equation, that has a close relationship with the NCL problem.

Remark 13 (Extensions of the Kolmogorov n-width to nonlinear approximations). The autoencoders we implement in our test cases are at least continuous as composition of continuous activations and linear functions. In the literature there exist possible nonlinear extensions of the KnW such as the manifold width [75],

$$\delta_n(L(K),X) = \inf_{\substack{\psi\in\mathcal{C}(X,\mathbb{R}^n)\\ \phi\in\mathcal{C}(\mathbb{R}^n,X)}} \sup_{x\in L(K)} \|x-(\phi\circ\psi)(x)\|_X, \tag{6.10}$$

library widths [244], and entropy numbers, for more insights see [25].

6.2.2 Singular values decomposition and discrete spectral decay

From the discrete point of view the same problematic in tackling the reduction of parametric PDEs with slow KnW decay is encountered: in this case the discrete solution manifold is actually the set of discrete solutions of the full-order model for a selected finite set of parameters. Singular Value Decomposition (SVD) or eigenvalue decomposition (for symmetric positive definite matrices), usually employed on the snapshots matrix for the evaluation of the reduced modes, are characterized by the fact that modes are linear combinations of the snapshots. This is not enough to approximate snapshots that are orthogonal with respect to the collection of the training set.

In practice, looking at the residual energy retained by the discarded modes, is an indicator of approximability with linear subspaces. Let us assume that d is the total number of degrees of freedom of the discrete problem, N_{train} is the number of training snapshots, and, r, the reduced dimension such that, $r\leq N_{\text{train}}\ll d$. Then, $X\in\mathbb{R}^d\times\mathbb{R}^{N_{\text{train}}}$ is the matrix that has the training snapshots $\{U_i\}_{i=1}^{N_{\text{train}}}$ as columns, $V\in\mathbb{R}^d\times\mathbb{R}^r$ are the reduced modes from the SVD of X, and $\{\sigma_i\}_{i=1}^d$ are the increasingly ordered singular values. It is valid the following relationship of the residual energy (to the left) with the KnW (to the right),

$$\sqrt{\sum_{i=r+1}^{d}\sigma_i^2} = \|(I-VV^T)X\|_F \geq \max_{i=1,\dots,N_{\text{train}}} \|(I-VV^T)U_i\|_2 \geq d_n(\{U_i\}_{i=1}^{N_{\text{train}}})_{\mathbb{R}^d},$$

where $\|\cdot\|_F$ is the Frobenious norm.

Even though some problems have a slow decaying KnW, this affects only the asymptotic convergence of the ROMs w.r.t. the reduced dimension, while for some applications a satisfying accuracy of the projection error is reached with less than 100 modes, as shown in our benchmarks. So what is actually lost in MOR for problems with a slow decaying KnW is the fast convergence of the projection error w.r.t. the reduced dimension, not the possibility to perform a MOR with enough modes. Moreover, the discrete solution manifold's KnW depends on the time step and spatial discretization size, so that, especially when a coarse mesh is employed, the Knw decays faster w.r.t. the KnW of the continuous solution manifold.

6.2.3 Convolutional Autoencoders

We have chosen to overcome the slowly decaying KnW problem employing autoencoders [106] as nonlinear dimension reduction method substituting POD. Some ROMs predict the latent dynamics on a linear trial manifold with artificial neural networks [202], so are still classified as linear ROMs and, in fact, they are still affected by the slow KnW decay. We remark that while some nonlinear approaches to model order reduction are specifically tailored for advection-dominated problems [254, 191], autoencoders are a more general approach. On the other hand, they are also particularly suited to advection-dominated problems with respect to local and/or partitioned ROMs that implement domain decomposition, even when nonlinear dimension reduction techniques are employed locally [163]: this is because considering a non-discrete parametric space, like the time interval of a simple linear advection problem, an infinite number of local linear and/or nonlinear ROMs would be needed to counter the slow decaying KnW. As anticipated, we implement convolutional autoencoders [106] in libtorch the C++ frontend of PyTorch [196]. We remark that the procedure we developed, considering also teacher-student training of a reduced decoder in section 6.3.4, can be generally extended to any architecture that can approximate with a sufficiently good accuracy the solution manifold through a low-dimensional latent representation. The choice of a CAE is particularly beneficial when the solution snapshots are associated to a structured mesh and when the components of the vectorial solution fields to be approximated have similar features so that the convolutional filters can be shared among the channels of the CAE.

Let us define $X_h \subset \mathbb{R}^d$ as the state discretization space with d the number of degrees of freedom and h the discretization step. The snapshots are divided in a training set $\mathcal{U}_{\text{train}} = \{\mathbf{U}_i\}_{i=1,\dots,N_{\text{train}}} \subset X_h$ and a test set $\mathcal{U}_{\text{test}} = \{\mathbf{U}_i\}_{i=1,\dots,N_{\text{test}}} \subset X_h$. If the problem has different states and/or vectorial states the training and test set are split in channels for each state and/or component. For example in the 2d nonlinear conservation law, we consider two channels, one for each velocity component. In the shallow water test case, we consider three channels, one for each velocity component and one for the free surface height. So, in general, we reshape the snapshots such that $\mathcal{U}_{\text{train}}, \mathcal{U}_{\text{test}} \subset (\mathbb{R}^{d/c})^c$ where c is the number of channels.

As preprocessing step the snapshots are centered and normalized to assume values in the interval $[-1,1]$

$$\frac{2}{\mathcal{U}_{\text{max}} - \mathcal{U}_{\text{min}}} \left(\mathcal{U}_{\text{train}} - \mathcal{U}_{\text{mean}} - \frac{\mathcal{U}_{\text{min}} + \mathcal{U}_{\text{max}}}{2} \right), \tag{6.11}$$

where $\mathcal{U}_{\text{mean}}, \mathcal{U}_{\text{max}}, \mathcal{U}_{\text{min}} \in (\mathbb{R}^{d/c})^c$ are evaluated channel wise. The same values obtained from the training snapshots $\mathcal{U}_{\text{train}}$, are employed to center and normalize the test set $\mathcal{U}_{\text{test}}$.

We define the encoder $\psi : (\mathbb{R}^{d/c})^c \to \mathbb{R}^r$ and the decoder $\phi : \mathbb{R}^r \to (\mathbb{R}^{d/c})^c$, where $r \ll d$ is the reduced or latent dimension, as neural networks made by subsequent convolutional layers and linear layers at the end and at the beginning, respectively. For the particular architecture used in the applications we defer it to the section 6.4.3. In Figure 6.1 is represented the convolutional autoencoder applied for the 2d nonlinear conservation law test case, with an approximate size of the filters and the actual number of layers [1].

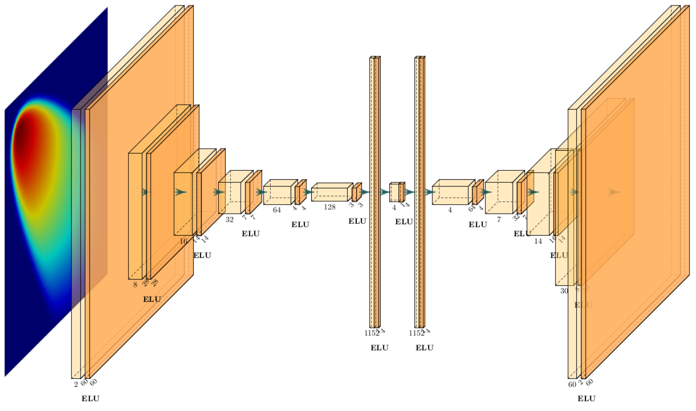


Fig. 6.1 The convolutional autoencoder architecture employed for the 2d nonlinear conservation law test case: the same number of convolutional layers are shown, while the sizes of each layer are rescaled for a better graphical representation of the architecture.

Remark 14 (Regularity). Regarding the regularity of the CAE, related to the choice of activation functions, it is proved in Theorem 4.2 from [161] that NM-LSTM and nonlinear manifold Galerkin methods are asymptotically equivalent provided that the decoder is twice differentiable. Since we are only employing the NM-LSTM method, our only concern in the choice of activation functions is that the reconstruction error is sufficiently low, so that the accuracy of the whole procedure is not undermined.

For each batch $\{\mathbf{U}_i\}_{i=1}^b \subset \mathcal{U}_{\text{train}}$, the loss employed is the sum of the reconstruction error, and a regularizing term for the weights:

$$\mathcal{L}(\{\mathbf{U}_i^{t_i}\}_{i=1}^b;\Theta) = \frac{1}{b}\sum_{i=1}^b \frac{\|\mathbf{U}_i^{t_i} - (\phi\circ\psi)(\mathbf{U}_i^{t_i})\|_2^2}{\|\mathbf{U}_i^{t_i}\|_2^2} + \lambda_1\|\Theta\|_2^2, \tag{6.12}$$

where Θ represents the weights of the convolutional autoencoder. The choice of the relative mean squared reconstruction error is important when, varying the parameter $\{\boldsymbol{\mu}_i\}_{i=1}^{N_{\text{train}}}$, the snapshots $\{\mathbf{U}_i\}_{i=1}^{N_{\text{train}}}$ have different orders of magnitude: for example this is the case of flows propagating from a local source on the whole domain with a constant zero state as initial condition.

The training is performed with Adam stochastic optimization method [147]. After the training the whole evolution of the dynamics is carried out with a nonlinear optimization algorithm minimizing

[1]Figure 6.1 and Figure 6.3 were made with the open-source package from GitHub https://github.com/HarisIqbal88/PlotNeuralNet.

the residual on the latent domain, as described in section 6.3. For each new parametric instance and associated initial state $\mathbf{U}_0 \in X_h$, the latent initial condition is found with a single forward of the encoder $z_0 = \psi(\mathbf{U}_0)$ after centering and normalizing $\mathbf{U}_0$. Then, for each time instant t, the decoder

$$\phi(z(t)) = \mathbf{U}(t) \in (\phi \circ \psi)(X_h) \subset \mathbb{R}^d, \tag{6.13}$$

is used as parametrization of an approximate solution manifold as will be explained in section 6.3, including in ϕ the renormalization of the output.

Remark 15 (Initial condition). With respect to the initial implementation of Carlberg et al [161] our approach for the definition of the latent parametrization of the solution manifold is different in the management of the initial condition. Instead of using directly the decoder ϕ, they define the map from the latent to the state space $f : \mathbb{R}^r \to \mathbb{R}^d$, including the renormalization in the ϕ map for brevity, as

$$f(z) = \phi(z) + \mathbf{U}_0(\mu) - \phi(\psi(\mathbf{0})), \quad \text{s.t.} \quad f(\psi(\mathbf{0})) = \mathbf{U}_0(\mu), \tag{6.14}$$

so that the reconstruction is exact at the initial parametric time instances. So, supposing that the initial condition is parametrically dependent, all initial conditions coincide to $\psi(\mathbf{0})$ in the latent space, and the decoder has to learn variations from the initial latent condition. We prefer instead to split the initial conditions in the latent space in order to aid the nonlinear optimization algorithm, and leave to the training of the CAE the accurate approximation of the initial condition. This splitting of the initial conditions can be seen in Figure 6.6 and 6.13. The additional cost of our implementation is the forward of the initial parametric condition through the encoder as first step.

Remark 16 (Inductive biases). Imposing inductive biases to increase the convergence speed of a deep learning model to the desired solution has always been a winning strategy in machine learning. This is translated in the context of physical models with the possibility to include, among others, the following inductive biases: first principles (conservation laws [161], equations governing the physical phenomenon), geometrical simmetries (group invariant filters [233, 90]), numerical schemes/residuals (discrete residuals, latent time advancement with Runge-Kutta schemes), latent regularity (minimize the curvature of latent trajectories), latent dynamics (linear or quadratic latent dynamics [109]). As inductive bias, we will impose the positivity of the state variables that are known to be positive throughout their trajectory with a final ReLU activation.

6.3 Evolution of the latent dynamics with NM-LSPG-ROC

The nonlinear manifold method introduced by Carlberg et al [161] does not perform a complete dimension reduction since at each time step the decoder reconstructs the state from the latent coordinates to the whole domain, still depending on the number of degrees of freedom of the FOM. We revisit the nonlinear manifold least-squares Petrov-Galerkin method (NM-LSPG) with small modifications and introduce two novel hyper-reduction procedures: one combines teacher-student training of a compressed decoder with the reduced over-collocation method (NM-LSPG-ROC-TS), the other implements only the hyper-reduction of the residual with reduced over-collocation (NM-LSPG-ROC).

6.3.1 Nonlinear manifold least-squares Petrov-Galerkin

We assume that the numerical method of preference discretizes the system 6.1 in space and in time with an implicit scheme, $G_{h,\delta t} : \mathcal{P} \times X_h \times X_h^{|I_t|} \to X_h$

$$G_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s\in I_t}) = \mathbf{0}, \tag{6.15}$$

where h, δt are the spatial and temporal discretization steps chosen, $X_h \subset \mathbb{R}^d$ is the state discretization space (d is the number of degrees of freedom), and I_t is the set of past state indexes employed in the temporal numerical scheme to solve for $\mathbf{U}_h^t$. We remark that the numerical discretization employed can differ from the one used to solve for the full-order training snapshots. The method is thus equations-based rather than fully intrusive. This will be the case for the 2d shallow water equations model in Subsection 6.4.2.

For each discrete time instant t the following nonlinear least-squares problem is solved for the latent state $\mathbf{z}^t \in Z$, with the Levenberg-Marquardt algorithm [209]

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}} \, \|G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})\|_{X_h}^2. \tag{6.16}$$

That is for each time instant the following intermediate solutions $\{\mathbf{z}^{t,k}\}_{k\in\{0,\dots,N(t)\}}$, $\mathbf{z}^{t,0} = \mathbf{z}^{t-1,N(t-1)}$ of the linear system in $\mathbb{R}^r$ are computed,

$$\left((dG^{t,k-1}d\phi^{t,k-1})^T dG^{t,k-1}d\phi^{t,k-1} + \lambda\, I_d\right)\delta\mathbf{z}^{t,k} = -(dG^{t,k-1}d\phi^{t,k-1})^T G^{t,k}, \tag{6.17}$$

$$\mathbf{z}^{t,k} = \mathbf{z}^{t,k-1} + \alpha^k\, \delta\mathbf{z}^{t,k}, \tag{6.18}$$

where

$$d\phi^{t,k-1} := \frac{d\phi(\mathbf{z}^{t,k-1})}{d\mathbf{z}^{t,k-1}} \in \mathbb{R}^{d\times r}, \tag{6.19}$$

$$dG^{t,k-1} := \left.\frac{dG_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s\in I_t})}{d\mathbf{U}_h^t}\right|_{(\mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s\in I_t}) = (\phi(\mathbf{z}^{t,k-1}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})} \in \mathbb{R}^{d\times d}, \tag{6.20}$$

λ is a factor that evolves during the nonlinear optimization and balances between a Gauss-Newton and a steepest descent method, and finally α^k is a parameter found with a trust-region method. In the implementation in Eigen [111], $\lambda\ I_d$ is scaled with respect to the diagonal elements of $(dG^{t,k-1}d\phi^{t,k-1})^T dG^{t,k-1}d\phi^{t,k-1}$. All the tolerances for convergence are set to machine precision, and the maximum number of residual evaluations is set to 7 unless explicitly stated differently in the numerical results section 6.4.

Remark 17 (Least-squares Petrov-Galerkin). The method is called manifold LSPG because it refers to the LSPG method usually applied when the manifold is linear. It consists in multiplying the residual to the left with a different matrix Ψ with respect to the linear embedding Φ of the reduced coordinates into the state space,

$$\Psi^T G_{h,\delta t}(\boldsymbol{\mu}, \Phi\mathbf{z}) = 0, \tag{6.21}$$

where, $\Phi \in \mathbb{R}^{d\times r}$ is the basis of the linear reduced manifold contained in $X_h \subset \mathbb{R}^d$, $\mathbf{z} \in \mathbb{R}^r$, and Ψ is to be defined: the left subspace $\Psi \in \mathbb{R}^{d\times r}$ is used to enforce the orthogonality of the nonlinear residual to a left subspace $\mathcal{L} \subset \mathbb{R}^d$. Applying Newton's method because of the nonlinearities, the problem is translated into the iterations for $k = 1, \dots, K$:

$$\Psi^T dG\Phi\ \delta\mathbf{z}^k = -\Psi^T G^k, \tag{6.22}$$

$$\mathbf{z}^k = \mathbf{z}^{k-1} + \alpha^k\ \delta\mathbf{z}^k. \tag{6.23}$$

The step length α^k is computed after a line search along the direction p^k. Usually Φ is chosen from a POD basis of the state variable $\mathbf{z} \in X_h$. For the left subspace, that imposes orthogonality constraints, different choices can be applied. In general, given the system

$$dG\Phi\ \delta\mathbf{z}^k = -G^k, \tag{6.24}$$

the least squares solution is the one orthogonal to the range of $dG\Phi$. In the case of dG symmetric positive definite we have that $\Psi \subset < dG\Phi > = < \Phi >$ i.e. $\Psi = \Phi$ and the method is called Galerkin projection, but in general if this is not true then the optimal left subspace remains $\Psi = dG\Phi$. For examples where the Galerkin projection is not optimal see [45], section 3.4 *Numerical comparison of left subspaces*. This is often the case for advection-dominated discretized systems of PDEs: in these cases LSPG is preferred to Galerkin projection.

Remark 18 (Manifold Galerkin). The manifold Galerkin method proposed in [161] assumes that the columns of the Jacobian of the decoder are good approximations of the state velocity space: if a spatial discretization is applied and the residual has the form $\mathcal{G}_h(\boldsymbol{\mu}, \mathbf{U}) = \dot{\mathbf{U}} - \mathbf{f}(\boldsymbol{\mu}, \mathbf{U})$, where $\mathbf{f}$ is a generic, possibly nonlinear, vector field, then

$$\dot{\mathbf{z}} = \underset{\mathbf{v}\in\mathbb{R}^r}{\operatorname{argmin}} \|d\phi(\mathbf{z})\mathbf{v} - \mathbf{f}(\boldsymbol{\mu}, \phi(\mathbf{z}))\|^2, \tag{6.25}$$

is solved for the latent state velocity, under the hypothesis that $d\phi(\mathbf{z})$ has full-rank and $d\phi$ is a good approximation of the full-order state velocity even if the autoencoder is trained only on the values of the state for different times and parameters, without considering its velocity. If ϕ is linear, we obtain the Galerkin method presented in the remark 17, after having applied a temporal discretization scheme and multiplied the resulting equation to the left with $d\phi(\mathbf{z}) = \Phi$ as is the case for linear Galerkin projection: the discretize-then-project and project-then-discretize approaches are equivalent in this case [161].

The LSPG performs better as shown in [161], even if they are asymptotically equivalent also in the case of a nonlinear manifold, provided ϕ is twice differentiable. So we have chosen to employ only the NM-LSPG method and do not compare it with the nonlinear manifold Galerkin (NM-G) method.

For the numerical tests we have performed, the numerical approximation of the Jacobian of the residual $G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})$ is accurate enough. So in the implementation, at each iteration step the Jacobian of the residual with respect to the latent variable, that is $dG^{t,k-1}d\phi^{t,k-1}$ of equation 6.17, is approximated with finite differences. The step size is taken sufficiently lower than the distance between consecutive latent states.

6.3.2 Reduced over-collocation method

At the point of equation 6.17, the model still depends on the number of degrees of freedom of the full-order model d, since at each time step and optimization step the latent reduced variable $\mathbf{z} \in \mathbb{R}^r$ is forwarded to the reconstructed state $\mathbf{U}_h = \phi(\mathbf{z}) \in \mathbb{R}^d$. A possible solution is represented by the reduced over-collocation method [51], for which the least squares problem 6.16 is solved only on a limited number of points $r < r_h << d$,

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{6.26}$$

where P_{r_h} is the projection onto r_h standard basis elements in $\mathbb{R}^d$ associated to the over-collocation nodes or magic points and selected as described later. Afterwards the Levenberg-Marquardt algorithm is applied as described in the previous section, to solve the least squares problem 6.26.

At this point we make the assumption that the method used to discretize the model has a local formulation so that each discrete differential operator can be restricted to the nodes/magic points of the hyper-reduction and consequently, the least-squares problem 6.26 reduces to

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, P_{r_h}(\phi(\mathbf{z})), \{P_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{6.27}$$

$$= \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|\tilde{G}_{h,\delta t}(\boldsymbol{\mu}, \tilde{\phi}(\mathbf{z}), \{\tilde{\phi}(\mathbf{z}^s)\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{6.28}$$

where the projected residual $\tilde{G} = P_{r_h} \circ G$ is introduced and the compressed decoder $\tilde{\phi}$ is defined to substitute $P_{r_h} \circ \phi$ with another embedding from the latent space to the hyper-reduced space in $\mathbb{R}^{r_h}$, such that the whole structure of the decoder is reduced as described in subsection 6.3.4 to further decrease the computational cost.

The Levenberg-Marquardt method is applied also to the hyper-reduced system

$$\left((d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1})^T d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1}+\lambda\, I_d\right)\delta\mathbf{z}^{t,k}=-(d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1})^T\tilde{G}^{t,k}, \tag{6.29}$$

$$\mathbf{z}^{t,k}=\mathbf{z}^{t,k-1}+\alpha^k\,\delta\mathbf{z}^{t,k}, \tag{6.30}$$

and as for the manifold LSPG method, the Jacobian matrix $d\tilde{G}^{t,k-1}d\tilde{\phi}^{t,k-1}$ is numerically approximated at each optimization step in the implementations.

Remark 19 (Submesh needed to define the hyper-reduced differential operators). To compute $\tilde{G}_{h,\delta t}$ in the nodes/magic points of the reduced over-collocation method, some adjacent degrees of freedom are needed by the discrete differential operators involved. So, actually, at each time step not only the values of the state variables at the magic points are needed, but also at the adjacent degrees of freedom in the mesh with possible overlappings. We represent the restriction to this submesh of magic points and adjacent degrees of freedom with the projector $P^s_{r_h}\in\mathbb{R}^{s_h\times d}$, where s_h is the number of degrees of freedom of the submesh. Equation 6.28 becomes

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}}\,\|P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}. \tag{6.31}$$

The stencil around each magic point to consider depends on the type of numerical scheme. Since we are using the finite volume method we have to consider the degrees of freedom of the adjacent cells. For example, for Cartesian grids, the schemes chosen for the 2d nonlinear conservation law have a stencil of 1 layer of adjacent cells, 4 additional nodes in total for a cell of the interior of the mesh. The 2d shallow water equations case requires a stencil of 2 layers instead, 12 additional nodes in total for a cell of the interior of the mesh. The two cases are shown in Figure 6.2.

6.3.3 Over-collocation nodes selection

The nodes/magic points of the over-collocation hyper-reduction method should be defined such that

$$\begin{aligned}P_{r_h}=\underset{P^TP\in\mathcal{S}}{\operatorname{argmin}}\ &\max_{(\mathbf{z}^t,\{\mathbf{z}^s\}_{s\in I_t})\in\mathcal{T}}\\ &\|G_{h,\delta t}(\boldsymbol{\mu},(\phi(\mathbf{z})),\{(\phi(\mathbf{z}^s))\}_{s\in I_t})-P^TPG_{h,\delta t}(\boldsymbol{\mu},P(\phi(\mathbf{z}^t)),\{P(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}\end{aligned} \tag{6.32}$$

where $\mathcal{S}=\{P\in\mathbb{R}^{r_h\times d}\mid P=(\mathbf{e}_{i_1}|\dots|\mathbf{e}_{i_{r_h}})^T\}$ is the space of projectors onto r_h coordinates associated to the standard basis $\{\mathbf{e}_i\}_{i\in\{1,\dots,d\}}$ of $\mathbb{R}^d$ and $\mathcal{T}$ is the space of discrete solution trajectories varying with respect to $\boldsymbol{\mu}$ and the intermediate optimization steps

$$\begin{aligned}\mathcal{T}=\{&(\boldsymbol{\mu},t,k,\mathbf{z}^{t,k},\{\mathbf{z}^{s,k}\}_{s\in I_t})\in\mathcal{P}\times V_{h,\boldsymbol{\mu}}\times V_{h,\boldsymbol{\mu},t}\times\mathbb{R}^d\times\mathbb{R}^{d\times|I_t|}\ |,\\ &\left((dG^{t,k-1}d\phi^{t,k-1})^TdG^{t,k-1}d\phi^{t,k-1}+\lambda\, I_d\right)\delta\mathbf{z}^{t,k}=-(dG^{t,k-1}d\phi^{t,k-1})^TG^{t,k}\},\end{aligned}$$

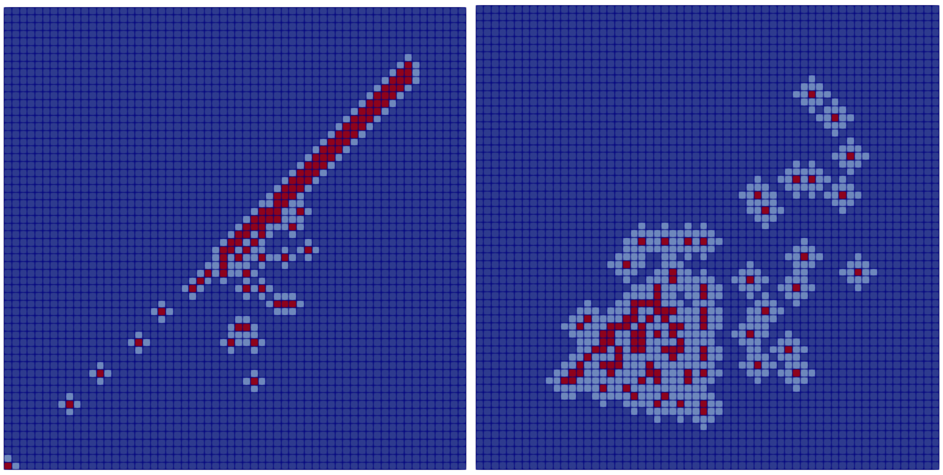


Fig. 6.2 Left: 100 magic points of the 2d nonlinear conservation law test case. Right: 100 magic points of the 2d shallow water test case. The magic points are represented in red, the stencils of the cells and associated degrees of freedom needed for the evaluation of the discrete differential operators are in light-blue. The discarded nodes in the evolution of the dynamics with NM-LSPG-ROC are in blue. The stencil is made by 1 layer of cells in the NCL case and 2 layers in the SW case.

where $V_{h,\boldsymbol{\mu}}$ is the discrete space of time instants, possibly depending on h and the parameter $\boldsymbol{\mu}$, and $V_{h,\boldsymbol{\mu},t}$ is the discrete space of optimization steps at time t. Essentially we want that the nodes/magic points approximate the residuals among all time steps, optimization steps and parameter instances.

There are many possible algorithms to solve Equation 6.32 for P_{r_h}. Usually they are not optimal and compromise between computational cost and accuracy, depending on the problem at hand. Among others, these algorithms are part of the hyper-reduction methods such as empirical interpolation method [20], discrete empirical interpolation method [48], Gauss-Newton tensor approximation (GNAT) [46], space-time GNAT [55] and solution-based nonlinear subspace GNAT [56](SNS-GNAT).

In particular, if $\mathcal{G}_h(\boldsymbol{\mu},\mathbf{U}) = \dot{\mathbf{U}} - \mathbf{f}(\boldsymbol{\mu},\mathbf{U})$, then, following some considerations that justify SNS-GNAT [56], the training modes employed to find the nodes/magic points of the over-collocation method are represented by the state snapshots instead of the residual fields. We could in principle use the reduced fields $\phi(\mathbf{z})$ but in practice, for the test case we considered, the full-order state snapshots were enough, without even saving the intermediate optimization states.

The procedure is applied at the same time for all the components of the state field $\mathbf{U} \in X_h$ and follows a greedy approach. We remark that the spatial discretization must not vary, so that the degrees of freedom correspond to the same spatial and physical quantity over time and for every parameter instance. Some approaches tackle also geometry deformations, but keeping the same number of degrees of freedom in a reference system [239].

The algorithm 6 is an adaptation of GNAT from Algorithm 3 in [46] to the simpler case in which the Jacobian matrix is not considered in the hyper-reduction (since in the LM method the Jacobian matrix is approximated with finite differences from the residual, see equation 6.30). Also, with respect to Algorithm 3 in [46], the new node/magic point at line 21 in Algorithm 6 is found without computing also the reconstruction error of the degrees of freedom associated to its stencil.

Algorithm 6: Greedy nodes/magic points evaluation for ROC method.

```
input ::
    $\mathcal{U}_{\text{train}} := \{\mathbf{U}_{\boldsymbol{\mu},t}\}_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}},\ t\in V_{\boldsymbol{\mu},h}}$, training state fields,
    $n_{r_{\text{init}}}$ nodes/magic points at the boundaries,
    $n_{r_h}$, number of nodes/magic points,
    $N_{\text{modes}}$ number of training modes.
output ::$r_h$ nodes/magic points used to define $P_{r_h}$.
```

Remark 20 (Comparison between GNAT and reduced over-collocation). We must say that the GNAT method, employed for the implementation of NM-LSPG with shallow masked autoencoders in [146], is a generalization of the reduced over-collocation method. In some cases though, they may perform similarly. For $P \in \mathcal{S}$, let us define

$$\mathbf{r} = r_{t,\boldsymbol{\mu},k} = G_{h,\delta t}(\boldsymbol{\mu}, P(\phi(\mathbf{z}^{t,k})), \{P(\phi(\mathbf{z}^{s,k-1}))\}_{s\in I}),\ \forall (t,k,\boldsymbol{\mu}) \in V_{h,\boldsymbol{\mu}} \times V_{h,\boldsymbol{\mu},t} \times \mathcal{P},$$

and the GNAT projection operator $\mathbb{P} = \Phi(P\Phi)^{\dagger}P$, where Φ is the matrix where the columns correspond to a chosen basis (it could be the FOM residual snapshots, ROM residual snapshots, ROM Jacobian and residual snapshots, see [46]). In particular, if $\Phi = P_{r_h}^T = P^T$ we have that

$$\mathbb{P} = \Phi(P_{r_h}\Phi)^{\dagger}P_{r_h} = \Phi(P_{r_h}\Phi)^{-1}P_{r_h} = P_{r_h}^T(P_{r_h}P_{r_h}^T)^{-1}P_{r_h} = P_{r_h}^T P_{r_h}, \tag{6.33}$$

that is the reduced over-collocation projection in the chosen nodes/magic points and extended to 0 in the remaining degrees of freedom. In this sense, the GNAT method includes the reduced over-collocation one.

However, for the class of problems we are considering, the GNAT method suffers from the slow decaying KnW. We have the inequalities

$$\begin{aligned}
d_n^2(\{r_{t,\boldsymbol{\mu},k}\}_{t,\boldsymbol{\mu},k}) &= \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - V_nV_n^T\mathbf{r}\|_2^2 \\
&\leq \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - V_nV_n^T\mathbf{r}\|_2^2 + \|V_nV_n^T\mathbf{r} - \mathbb{P}\mathbf{r}\|_2^2 \\
&= \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|\mathbf{r} - \mathbb{P}\mathbf{r}\|_2^2 = \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|(I-\mathbb{P})(I - V_nV_n^T)\mathbf{r}\|_2^2 \\
&\leq \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|(I-\mathbb{P})\|_2^2 \|(I - V_nV_n^T)\mathbf{r}\|_2^2 \\
&= \inf_{\dim V_n = n}\max_{t,\boldsymbol{\mu},k} \|\mathbb{P}\|_2^2 \|(I - V_nV_n^T)\mathbf{r}\|_2^2,
\end{aligned}$$

where $(t,k,\boldsymbol{\mu}) \in V_{h,\boldsymbol{\mu}} \times V_{h,\boldsymbol{\mu},t} \times \mathcal{P}$ whenever the maximum is taken. The term $\mathbf{r} - \mathbb{P}\mathbf{r}$ is the GNAT approximation error. The rightmost term is the usual bound on the hyper-reduction error [48], where it was used the fact that $\|\mathbb{P}\| = \|I - \mathbb{P}\|$. The second equality is valid because $\mathbf{r} - V_n V_n^T \mathbf{r}$ and $V_n V_n^T \mathbf{r} - \mathbb{P}\mathbf{r}$ are orthogonal. The third equality is obtained from the relations

$$\begin{aligned} & \mathbf{r} = (\mathbf{r} - \mathbf{r}_*) + \mathbf{r}_* = \mathbf{w} + \mathbf{r}_*, \qquad \text{with } \mathbf{r}_* := V_n V_n^T \mathbf{r}, \\ \Rightarrow \quad & \mathbf{r} - \mathbb{P}\mathbf{r} = \mathbf{w} + \mathbf{r}_* - \mathbb{P}\mathbf{w} - \mathbb{P}\mathbf{r}_* = \mathbf{w} - \mathbb{P}\mathbf{w}, \end{aligned}$$

where the last equality follows from $\mathbb{P}\mathbf{r}_* = \mathbf{r}_*$. Here we have supposed the nodes/magic points to be independent of V_n. So in the case of slow decaying Kolmogorov n-width, minimizing the hyper-reduced residual $\mathbb{P}\mathbf{r} \in < V_n > \subset \mathbb{R}^d$ is less efficient due to the slow convergence in n of the best approximation error $\|\mathbf{r} - V_n V_n^T \mathbf{r}\|_2^2$. This is one of the reasons why we employed ROC for the SWE test case; for the NCL test case GNAT and ROC performed similarly.

6.3.4 Compressed decoder teacher-student training

In order to make the whole methodology independent of the number of degrees of freedom, the decoder has to be substituted with a map $\tilde{\phi} : \mathbb{R}^R \to P_{r_h}(X_h) \subset \mathbb{R}^{r_h}$ from the latent space to the space of discrete full-order solutions evaluated only at the submesh containing the magic points and the needed adjacent degrees of freedom. As architecture, we choose a feedforward neural network (FNN) with one hidden layer, but actually the only requirement is that the computational cost is low enough such that not only a theoretical dimension reduction is achieved, but also a speedup is reached.

In the literature, the procedure for the training of the compressed decoder $\tilde{\phi}$ from the decoder ϕ is called teacher-student training [108]. In principle, the compressed decoder can be composed of layers inherited by the original decoder, such that the learning process involves only the final new additional layers. In our case, we preferred to train the compressed decoder anew: the latent projections of the training snapshots with the encoder $\psi(\mathcal{U}_{\text{train}}) = \{z_i\}_{i=1}^{N_{\text{train}}}$ are the inputs and the restriction of the snapshots to the submesh $P^s_{r_h}(\mathcal{U}_{\text{train}}) = \{\tilde{U}_i\}_{i=1}^{N_{\text{train}}}$ are the targets, see Equation 6.31. A schematic representation of the teacher-student training is represented in figure 6.3. Moreover, to speedup the offline stage, we use the training FOM snapshots restricted to the magic points as training outputs for the teacher-student training, while usually the reconstructed snapshots from the CAE -that would additionally need to be computed- are employed.

Again, for the training, we use a relative mean square loss with an additional regularizing term

$$\mathcal{L}(\{z_i\}_{i=1}^b; \tilde{\Theta}) = \frac{1}{b} \sum_{i=1}^{b} \frac{\|\tilde{U}_i - \tilde{\phi}(z_i)\|_2^2}{\|\tilde{U}_i\|_2^2} + \lambda_1 \|\tilde{\Theta}\|_2^2, \tag{6.34}$$

where $\tilde{\Theta}$ are the weights of the compressed decoder.

Remark 21 (Jacobian evaluation in Levenberg-Marquardt algorithm). The main reason why finite differences approximations of Jacobians are implemented in the NM-LSPG case, as explained at the end of subsection 6.3.1, is that the computational cost of evaluating the Jacobian of the full decoder

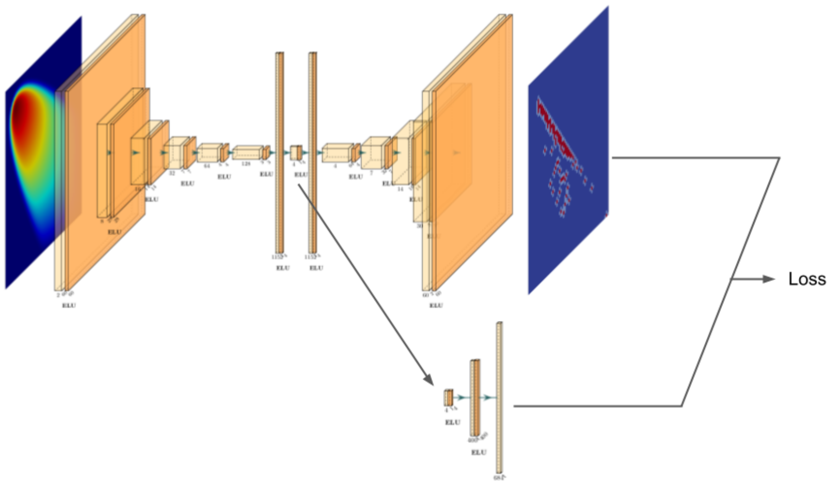


Fig. 6.3 Teacher-student training of the compressed decoder for the 2d nonlinear conservation law test case. The magic points, which the snapshots are restricted to, are shown in red over the domain.

is too high. In principle Jacobian evaluations of the compressed decoder are cheaper and could be employed, instead of relying again on finite differences approximations.

Remark 22 (Shallow masked autoencoders). We are motivated to write this article to extend the results in [146] to a generic architecture composed by neural networks. They performed the hyper-reduction of the nonlinear manifold method [161] with a shallow masked autoencoder, so that correctly masking the weights matrices of the decoder, its outputs correspond only to the submesh needed by the GNAT method, thus eliminating the dependence on the FOM's degrees of freedom. We want to reproduce, in some sense, this approach for an arbitrary autoencoder architecture, in this case a CAE, in order to tackle with the latest architectures developed in the literature the problem of solution manifold approximability: we think this is a major concern when trying to apply nonlinear MOR to real applications. In fact, as will be clear in the numerical results section 6.4, the reconstruction error of the autoencoder bounds from below the prediction error of our newly developed ROMs.

It can be seen that the new model order reduction is composed of two distinct procedures to achieve the independence on the number of degrees of freedom: first the residual from NM-LSPG in Equation 6.16 is hyper-reduced with ROC in Equation 6.26 and secondly the CAE's decoder is compressed with teacher-student training. In principle, we could substitute the use of the compressed decoder with the restriction of the final layer of the CAE's decoder into the magic points, while keeping the hyper-reduction with ROC of the residual. In this case, the whole methodology would still be dependent on the total number of degrees of freedom, but in practice a CAE's decoder forward is relatively cheap compared to the evaluation of the full residual. So, the hyper-reduction performed with ROC or GNAT only at the equations/residuals level, is already beneficial to reduce the computational cost. We will compare this variant of the NM-LSPG-ROC method with the one that employs the compressed decoder, also to verify the consistency of the teacher-student training that is omitted in the first case.

In the numerical results section 6.4 we will adopt the acronym NM-LSPG-ROC-TS or NM-LSPG-GNAT-TS for the method that employs the compressed decoder and NM-LSPG-ROC or NM-LSPG-GNAT for the method that performs the hyper-reduction only at the equations/residuals level.

6.4 Numerical results

We test the new methodology on two benchmarks with a relatively slow KnW: the first model is governed by a nonlinear conservation law 6.4.1 the second by the shallow water equations 6.4.2. Both are parametric, nonlinear and time-dependent, and the only other (non-temporal) parameter is a multiplicative constant of the initial condition. The mesh employed is the same: a 60×60 structured orthogonal grid.

All the CFD simulations are obtained by the use of an in-house open source library `ITHACA-FV` (In real Time Highly Advanced Computational Applications for Finite Volumes) [238, 237], developed in a finite volume environment based on the open-source library `OpenFoam` [264]. Regarding the implementation of the convolutional autoencoders and compressed decoders (CAE) we used libtorch, PyTorch C++ frontend, while for the training of the long-short term memory network (LSTM) we used PyTorch [196]. All the CFD simulations were performed on a Intel(R) Core(TM) i7-8750H CPU with 2.20GHz and all the neural networks trainings on a GeForce GTX 1060 GPU. Further reductions in the computational costs could be achieved exploiting the parallel implementation of the training procedures in PyTorch. The details of the architectures of the neural networks that will be employed are reported in the section 6.4.3.

The following notations are introduced: N^{μ}_{train}, N^{μ}_{test} are the numbers of train and test parameters, respectively; N^{t}_{train}, N^{t}_{test} are the number of time instances associated to the train and test parameters, respectively. The total number of training and test snapshots is thus $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}}$, and $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}}$, respectively.

The accuracy of the reduced-order models devised is measured with the mean realtive L^2-error and the maximum relative L^2-error, where the mean and max are taken with respect to the time scale: since the test cases depend on a non-temporal parameter, for each instance of these parameters a time-series corresponding to the discrete dynamics is associated; the mean and maximum are evaluated w.r.t the elements of these time-series. Let $\{u^{t_i}_{\mu}\}_{i=1,\dots N^t}$ and $\{U^{t_i}_{\mu}\}_{i=1,\dots N^t}$ be the predicted and true time-series N^t elements long, associated to the train or test parameter μ, the mean relative L^2-errors and maximum relative L^2-errors are then defined as

$$\varepsilon_{mean}(u_{\mu}, U_{\mu}) = \frac{1}{N^t} \sum_{i=1}^{N^t} \frac{\|u^{t_i}_{\mu} - U^{t_i}_{\mu}\|_{L^2}}{\|U^{t_i}_{\mu}\|_{L^2}}, \; \varepsilon_{max}(u_{\mu}, U_{\mu}) = \max_{i=1,\dots,N^t} \frac{\|u^{t_i}_{\mu} - U^{t_i}_{\mu}\|_{L^2}}{\|U^{t_i}_{\mu}\|_{L^2}}. \tag{6.35}$$

Remark 23 (Levenberg-Marquardt parameters). Regarding the Levenberg-Marqurdt nonlinear optimization algorithm, we remark that we approximate the Jacobians with forward finite differences, and the optimization process, for each time step, is stopped when the maximum number of residual evaluations is reached. This number is set to 7, including the evaluations related to the Jacobian computations. When 7 residual evaluations are not enough for the method to converge, it is explicitly reported.

6.4.1 Nonlinear conservation law (NCL)

We test our procedure for nonlinear model order reduction on a 2d nonlinear conservation law model (NCL). Two main reasons are behind this choice: the slow Kolmogorov n-width decay of the continuous solution manifold, and the possibility to compare our results with a similar test case realized with an implementation of nonlinear manifold based on shallow masked autoencoders and GNAT [146].

The parametrization affects the initial velocity as a scalar multiplicative constant $\mu \in [0.8, 2]$:

$$\begin{cases} \partial_t \mathbf{u} + \frac{1}{2}\nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \nu \Delta \mathbf{u} & (\mathbf{x}, t) \in [0,1]^2 \times [0,2], \\ \mathbf{u}(\mathbf{x}, 0) = 0.8 \cdot \mu \cdot \sin(2\pi x)\sin(2\pi y)\chi_{[0,0.5]^2} & \mathbf{x} \in [0,1]^2, \\ \mathbf{u}(\mathbf{x}, t) = 0 & (\mathbf{x}, t) \in \partial[0,1]^2 \times [0,2], \end{cases} \tag{6.36}$$

where the viscosity $\nu = 0.0001$. We will collect $N^{\mu}_{\text{train}} = 12$ equispaced training parameters from the range $\mu \in [0.8, 2]$ and $N^{\mu}_{\text{test}} = 16$ equispaced test parameters from the range $\mu \in [0.6, 2.2]$. The first two and the last two parameters will account for the extrapolation error. The time step is equal to $\Delta t = 1\text{e}-3$ seconds, but the training snapshots are collected every 4 time steps and the test snapshots every 20, thus $N^{t}_{\text{train}} = 501$, and $N^{t}_{\text{test}} = 101$. In the predictive online phase, the dynamics will be evolved with the same time step $\Delta t = 1\text{e}-3$. For easiness of representation, the train parameters are labelled from 1 to 12, and the test parameters are labelled from 1 to 16.

To have a qualitative view on the range of the solution manifold, we report the initial and final time snapshots for the extremal training parameters of the range $\mu \in [0.8, 2]$, in Figure 6.4.

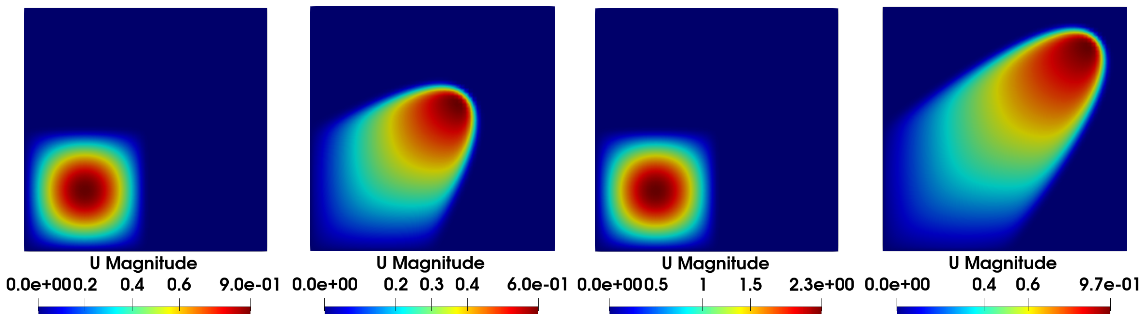


Fig. 6.4 **NCL.** From left to right: FOM solution of equation 6.36 at $(t,\mu) \in \{(0,0.8),(2,0.8)\}$ and $(t,\mu) \in \{(0,2),(2,2)\}$.

In this test case the GNAT method performed slightly better than the ROC method for hyper-reduction, so we employed the former to obtain the results shown.

Full-order model

We solve the 2d nonlinear conservation law for different values of the parameter μ with `OpenFoam` [264] open-source software for CFD. We employ the finite volumes method (FVM) in a structured orthogonal grid of 60×60 cells. If we represent with M the mass matrix, with D the diffusive matrix term, and with $C(U^{t-1})$ the advection matrix, then, at every time instant t, the discrete equation

$$\frac{M}{\Delta t}U^t + C(U^{t-1})U^t - \nu D U^t = \frac{M}{\Delta t}U^{t-1}, \tag{6.37}$$

is solved for the state U^t with a semi-implicit Euler method. The time step is $1\text{e}-3$, the initial and final time instants are 0 and 2 seconds. The linear system is solved with the iterative method BiCGStab preconditioned with DILU, until a tolerance of $1\text{e}-17$ on the FVM residual is reached.

The stencil of the numerical scheme at each cell involves the adjacent cells that share an interface (4 for an interior cell, 3 for a boundary cell and 2 for a corner cell): the value of the state at the interfaces is obtained with the bounded upwind method for the advection term and the surface normal gradient is obtained with central finite differences of two adjacent cell centers. So, in order to implement the reduced over-collocation method, for each node/magic point we have to consider an additional number of maximum 4 cells, that is 8 degrees of freedom to keep track of during the evolution of the latent dynamics; of course in practice they may overlap reducing the computational cost further.

The residual of the NMLSPG methods is evaluated with the same numerical scheme of the FOM. In the SWE test case the FOM and the ROMs employ different numerical schemes 6.4.2.

Manifold learning

As first step of the procedure the discrete solution manifold is learned through the training of a convolutional autoencoder (CAE) whose specific architecture is reported in Table 6.7. The CAE is trained with the ADAM [147] stochastic optimization algoritm for 2000 epochs, halving the learning rate by a factor of 2 if after 200 epochs the loss does not decrease. The initial learning rate is 1e−3, its lower bound is 1e−6. The number of training snapshots is $N_{\text{train}} = 12 \times 501 = 6012$, the batch size 20. It could be further refined in order the increase the efficiency of the whole procedure.

We choose as latent dimension 4, 2 dimensions greater than the number of parameters (the scalar multiplying the initial condition and time). We don't perform a convergence study of the accuracy with respect to the latent dimension since our focus is on the implementation of the NM-LSPG-ROC and NM-LSPG-ROC-TS model-order reduction methods: we are satisfied as long as the accuracy is relatively high, while the reduced dimension corresponds to an inaccurate linear approximating manifold spanned by the same number of POD modes.

In Figure 6.5 is shown the reconstruction error of the CAE and its decay with respect to the number of POD modes chosen $[4, 10, 25, 50, 100]$. To reach the same accuracy of the CAE with latent dimension 4, around 50 POD modes are needed. In order to state that the slow KnW decay problem is overcome by the CAE, the asymptotic convergence of the reconstruction error w.r.t. the latent dimension should be studied as was done for similar problems in [161, 146]. Instead, we will empirically prove that we can devise an hyper-reduced ROM with latent dimension 4 and accuracy lower than the 2% for the mean relative $L2$-error, a task that would be impossible for a POD based ROM with the same reduced dimension, since the reconstruction error is near 20% for all the test parameters.

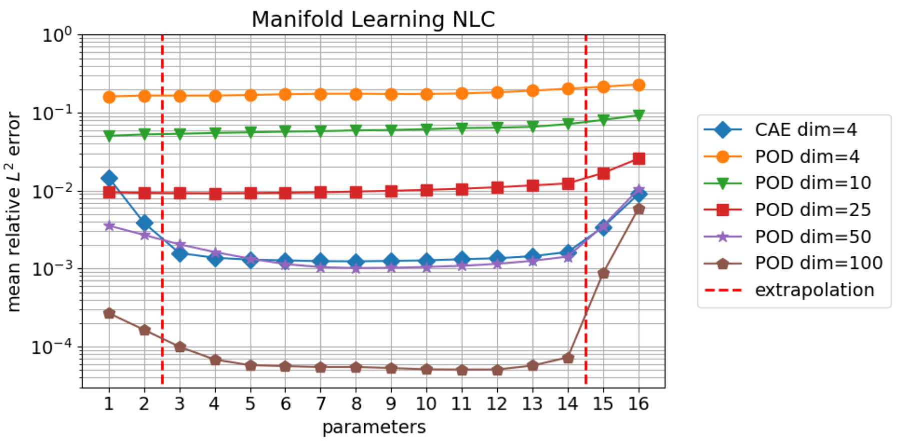


Fig. 6.5 **NCL.** Comparison between the CAE's and POD's projection errors of the discrete solution manifold represented by the $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 12 \cdot 501$ training snapshots. The error is evaluated on the 16 test parameters, each associated with a time series of $N^{t}_{\text{test}} = 101$, for a total of $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}} = 16 \cdot 101 = 1616$ test snapshots. The mean is performed over the time scale.

Without imposing any additional inductive bias a part from the regularization term in the loss from Equation 6.12 and the positiveness of the velocity components, the latent trajectories reported in Figure 6.6 for the odd parameters of the test set, are qualitatively smooth. An important detail to observe is that the initial conditions are well separated one from another in the latent space, see Remark 15, and that the dynamics is nonlinear. It also can be noticed that the 2 extremal parameters, corresponding to the extrapolation regime and represented in the plot by the two most outer trajectories that enclose the other 6, have smooth latent dynamics analogously to the others even though the reconstruction error starts degrading, as can be seen from Figure 6.5.

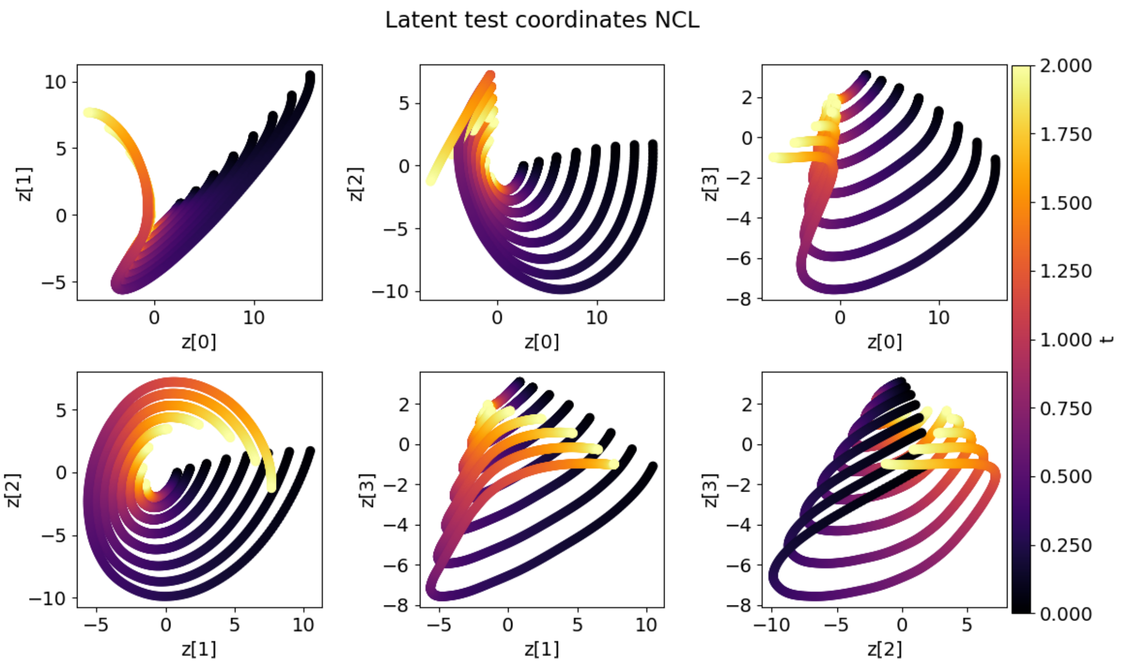


Fig. 6.6 **NCL.** Correlations among the 4 latent coordinates of the odd parameters of the test set, 8 in total. They are obtained projecting the test snapshots into the latent space of the CAE with the encoder. The coloring corresponds to the time instants from 0 to 2.

Hyper-reduction and teacher-student training

The selection of the magic points is carried out with the greedy Algorithm 6. The FOM snapshots employed correspond to the training parameters 1 and 12, but are sampled every 10 time step instead of every 4, as for the training snapshots of the CAE.

We perform a convergence study increasing the number of magic points from 50 to 100 and 150. The corresponding submesh sizes, i.e. the number of cells involved in the discretization of the residuals, are reported in the Table 6.1. The submesh size is bounded above with the total number of the cells in the mesh, that is 3600.

After the computation of the magic points, the FOM snapshots are restricted to those cells and employed as training outputs of the compressed decoder, as described in Section 6.3.4. The actual dimension of the outputs is twice the submesh size, since for each cell there are 2 degrees of freedom corresponding to the velocity components. The inputs are the 4-dimensional latent coordinates of the encoded FOM training snapshots, for a total of 6012 training input-output pairs. The architecture of the compressed decoder is a feedforward neural network (FNN) with one hidden layer, whose number of nodes is reported in Table 6.1, under 'HL size'. The compressed decoders architecture's specifics are also summarized in Table 6.7.

Each compressed decoder is trained for 3000 epochs, with an initial learning rate of $1e-4$, that halves if the loss from Equation 6.34 does not decrease after 200 epochs. The batch size is 20. The duration of the training is reported in Table 6.1 under 'TS total epochs', that stands for Teacher-Student training total epochs, along with the average cost for an epoch, under 'TS avg epoch'. The accuracy of the predictions on the test snapshots restricted to the magic points is assessed in Figure 6.7.

Table 6.1 **NCL.** In this table are reported for the NCL test case: the submesh size and the hidden layer (HL) number of nodes of the compressed decoder; the Teacher-Student training (TS) duration in seconds (TS total epochs), the Teacher-Student training average epoch duration in seconds (TS avg epoch); the average time step for NM-LSPG-GNAT with compressed decoder (avg GNAT-TS) in milliseconds, and the average time step for NM-LSPG-GNAT with the hyper-reduced residuals but full CAE decoder (avg GNAT-no-TS) in milliseconds. In red, the results are obtained with 13 maximum residual evaluations of the Levenberg-Marquardt algorithm, instead of the fixed 7, see Remark 23.

MP	Submesh size	HL size	TS total epochs
50	139	300	1682 s
100	246	350	2482 s
150	335	400	4245 s

MP	TS avg epoch	avg GNAT-TS GNAT	avg GNAT-no-TS
50	0.560 s	1.88 ms	4.496 ms
100	0.827 s	4.599 ms	4.499 ms
150	1.415 s	2.318 ms	4.208 ms

The convergence with respect to the number of magic points is shown in Figure 6.8. Since, especially for the NM-LSPG-GNAT-TS reduced-order model, the relative L^2-error is not uniform along the time scale, we report both the mean and max relative L^2-errors over the time series associated to each one of the 16 test parameters.

From Figure 6.8 and Table 6.1 it can be seen that NMLSPG-GNAT is more accurate than NM-LSPG-GNAT-TS even tough computationally more costly in the online stage. We underline that in the offline stage NM-LSPG-GNAT-TS requires the training of the compressed decoder. However, this could be performed at the same time of the CAE training, see the discussion section 6.5. The NM-LSPG-GNAT reduced-order model achieves better results also in the extrapolation error, sometimes even lower than the NM-LSPG method: this remains true even when increasing the maximum residual evaluations of the LM algorithm, and it may be related to the nonlinearity of the decoder that introduces difficult to interpret correlations of the latent dynamics with the output solutions restricted to the magic points.

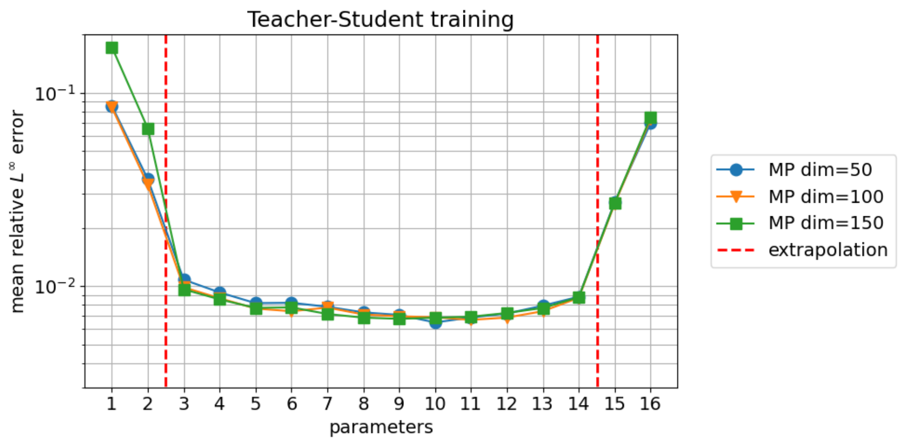


Fig. 6.7 **NCL.** Prediction accuracy on the test snapshots restricted to the magic points. The accuracy is measured with the relative L^∞-error averaged over the time trajectories associated to each one of the $N^\mu_{\text{test}} = 16$ test samples.

All the simulations of NM-LSPG, NM-LSPG-GNAT-TS and NM-LSPG-GNAT methods converge with a maximum of 7 residual evaluations of the LM algorithm, for all magic points reported, that is 50, 100, and 150 and for all the 16 test parameters. However, 3 test parameters could not converge for the method NM-LSPG-GNAT-TS with 100 magic points, so we increased the maximum function evaluations to 13 for all the test points. The higher computational cost per time step is shown in Table 6.1. Apart from those 3 test points not converging, the accuracy remains the same for the other 13 test parameters, so we have chosen to report the results in the case of 13 residual evaluations for all the 16 test parameters in Figure 6.8.

Comparison with data-driven predictions based on a LSTM

To assess the quality of the reduced-order models devised, we compare the accuracy in the training and extrapolation regimes, and the computational cost of the offline and online stages with a purely data-driven ROM in which the solutions manifold is approximated by the same CAE, but the dynamics is evolved in time with a LSTM neural network. The architecture of the LSTM employed is reported in Table 6.9. The results are summarized in Figure 6.9, the computational costs in Table 6.2.

The LSTM is trained for 10000 epochs with the ADAM stochastic optimization algorithm and an initial learning rate of 0.001, halved if after 500 epochs the loss does not decrease. The time series used for the training are the same $N_{\text{train}} = 6012$ training snapshots employed for the CAE. We remark that the LSTM cannot approximate the dynamics for an arbitrary time step, but it is fixed, depending on the training time step used, in this case 0.004 seconds.

The offline stage's computational cost is determined by the heavy CAE training for both the procedures, see the Discussion section 6.5 for possible remedies. The LSTM-NN achives a speedup close to 3 with respect to the FOM, differently from the NM-LSPG-GNAT and NM-LSPG-GNAT-TS

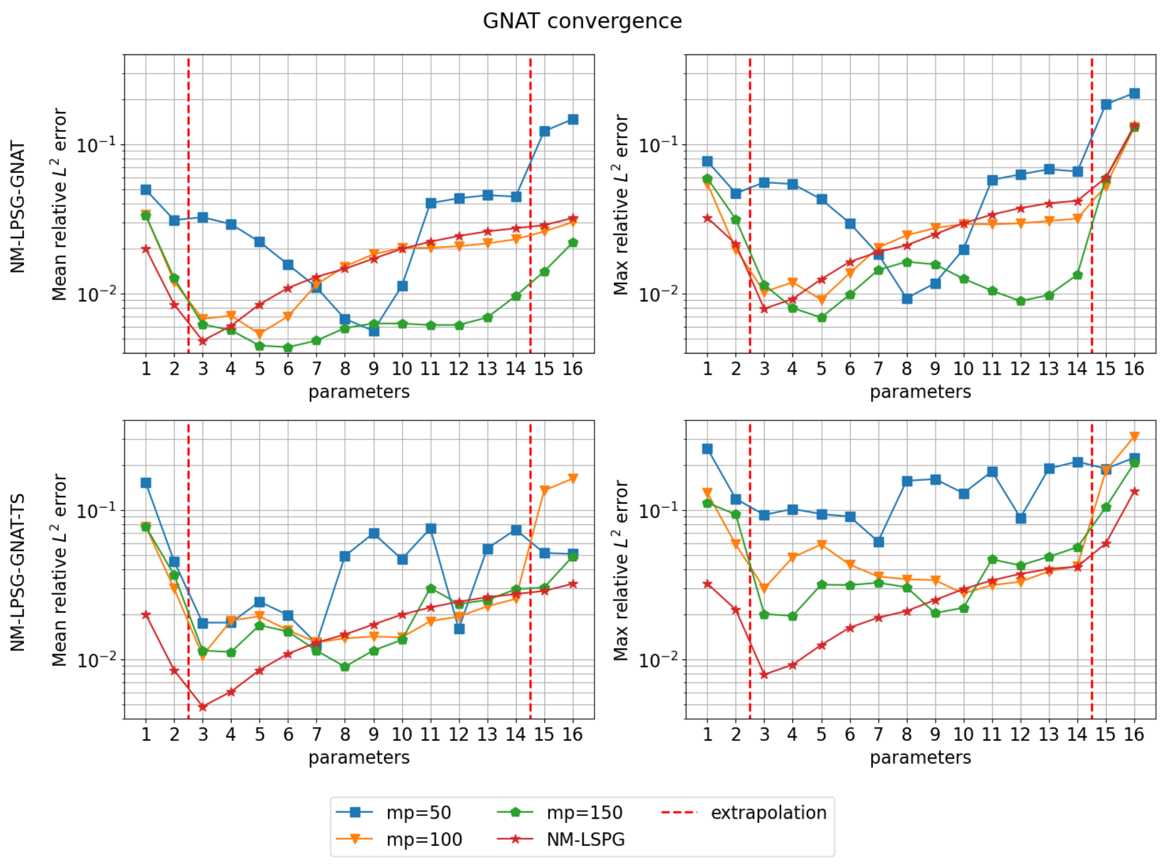


Fig. 6.8 **NCL.** First row: NM-LSPG-GNAT mean and max relative L^2-error. Second row: NM-LSPG-GNAT-TS mean and max relative L^2-error. In red also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the $N^{\mu}_{\text{test}} = 16$ test parameters.

methods. However, since the models are hyper-reduced, increasing the degrees of freedom refining the mesh should increase the computational cost of the FOM and NM-LSPG methods only, with due precautions. The average cost of the evaluation of the dynamics for the LSTM model with a time step of 0.004 seconds is associated to the label 'avg LSTM-NN full-dynamics'; thanks to vectorization, the dynamics for all the $N^{\mu}_{\text{test}} = 16$ parameters is evaluated with a single forward of the LSTM, thus the low computational cost reported.

Table 6.2 **NCL.** Offline stage: full-order model (FOM) computation of the $N_{\text{train}} = 6012$ snapshots (FOM snapshots evaluation), average cost of a single epoch for the training of the CAE with a batch size of 20 (CAE training single epoch avg), and total cost for 3000 epochs (CAE training); average epoch's cost for the LSTM training with a batch size of 100, and total cost for 10000 epochs. Online stage: for the FOM, NM-LSPG and LSTM-NN models it is reported the average of a single time step cost over all the $N_{\text{test}} = 1616$ test parameters and time series, and the average cost of the full dynamics over the $N^{\mu}_{\text{test}} = 16$ test parameters. The LSTM-NN full-dynamics is evaluated with a single forward.

Offline stage	Time
FOM snapshots evaluation	29.04 [s]
CAE training single epoch avg	10.1 [s]
CAE training	20213.3 [s]
LSTM-NN training single epoch avg	0.139 [s]
LSTM-NN training	1365 [s]

Online stage	Time
avg FOM time step	1.210 [ms]
avg FOM full dynamics	2.42 [s]
avg NM-LSPG time step	22.126 [ms]
avg NM-LSPG full-dynamics	133.2 [s]
avg LSTM-NN time step	0.432 [ms]
LSTM-NN full-dynamics	6.982 [ms]

The CAE reconstruction error in blue in Figure 6.9, lower bounds all the other models' errors. This is the reason why having a good accuracy of the CAE's solution manifold approximation is mandatory to build up nonlinear manifold methods. In this sense NM-LSPG-ROC-TS and NM-LSPG-GNAT-TS with respect to NM-LSPG-GNAT with shallow autoencoders [146] offer the possibility to choose an arbitrary architecture for the autoencoder, thus allowing a more accurate solution manifold approximation.

While in the training range from test parameter 3 to 14, the accuracy of the LSTM-NN is significantly better than NM-LSPG-GNAT and NM-LSPG-GNAT-TS models', in the extrapolation regime we observe that the predictions of the fully data-driven model degrades. The extrapolation error of the LSTM-NN model depends on the architecture chosen, regularization applied, training procedure, and hyperparameters tuning. What can be assessed from the results is that, outside the training range, the LSTM-NN's accuracy is dependent on all these factors, with sometimes a difficult interpretation of the results, while NM-LSPG-GNAT relies only on the number of magic points employed and the dynamics is evolved in time minimizing a physical residual directly related to the NCL model's equations.

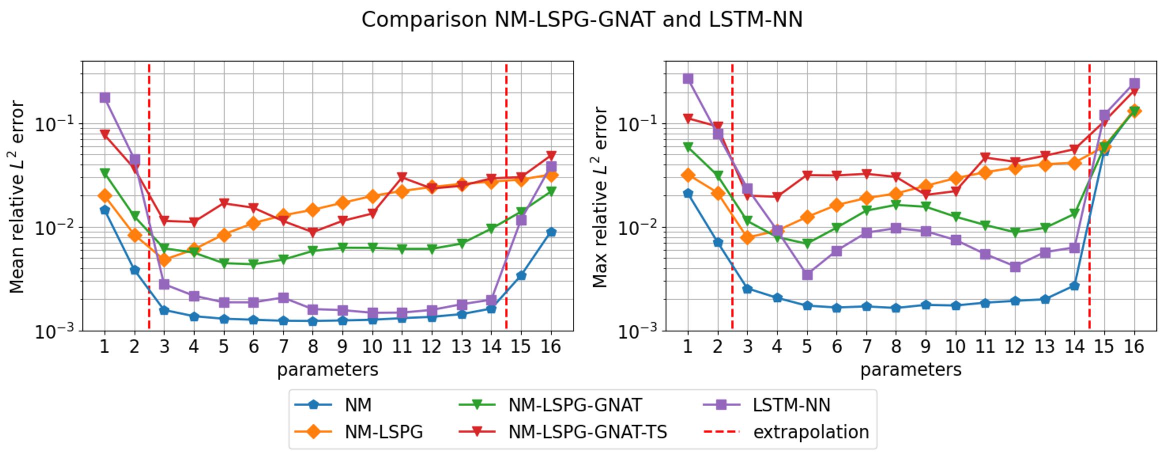


Fig. 6.9 **NCL.** Comparison of the accuracies between all the ROMs presented on the test set of $N^{\mu}_{\text{test}} = 16$ parameters, each associated to a time series of $N^{t}_{\text{test}} = 101$ intermediate solutions. The number of magic points employed for NM-LSPG-GNAT and NM-LSPG-GNAT-TS is 150. The mean and maximum relative L^2-errors are taken with respect to the time scale.

6.4.2 Shallow Water Equations (SWE)

The second test case we present is a 2d nonlinear, time-dependent, parametric model based on the shallow water equations (SWE). Also in this case, the non-temporal parameter affects the initial conditions, $\mu \in [0.1, 0.3]$, $t \in [0, 0.2] = I$:

$$\begin{cases} \partial_t(h\mathbf{u}) + \nabla\cdot(h\mathbf{u}\otimes\mathbf{u}) + \frac{1}{2}\,|\,\mathbf{g}\,|\,h\nabla h = 0 & (\mathbf{x},t) \in [0,1]^2 \times I, \\ \partial_t h + \nabla\cdot(\mathbf{u}h) = 0 & (\mathbf{x},t) \in [0,1]^2 \times I, \\ \mathbf{u}(\mathbf{x},0) = 0 & \mathbf{x} \in [0,1]^2, \\ h(\mathbf{x},0) = \mu\left(\frac{1}{e^1}\cdot e^{-\frac{1}{0.04-\|\mathbf{x}-\mathcal{O}\|_2^2}}\chi_{\|\mathbf{x}\|_2^2<0.2} + \chi_{\|\mathbf{x}\|_2^2\geq 0.2}\right) & \mathbf{x} \in [0,1]^2, \\ \mathbf{u}(\mathbf{x},t)\cdot\mathbf{n} = 0 & (\mathbf{x},t) \in \partial[0,1]^2 \times I, \\ \nabla h(\mathbf{x},t)\cdot\mathbf{n} = 0 & (\mathbf{x},t) \in \partial[0,1]^2 \times I, \end{cases} \tag{6.38}$$

where h is the water depth, $\mathbf{u}$ is the velocity vector, $\mathbf{g}$ is the gravitational acceleration, and $\mathcal{O}$ is the point $(0.5, 0.5) \in \Omega$. We consider a constant bathymetry $h_0 = 0$, so that the free surface height $h_{\text{total}} = h + h_0$ is equal to the water depth h.

The time step that will be employed for the evolution of the dynamics of the FOM is $1\mathrm{e}-4$ seconds. The training and test snapshots are sampled every 4 time steps. The training non-temporal parameters are $N^{\mu}_{\text{train}} = 10$ in number, and they are sampled equispacedly in the training interval $\mu \in [0.1, 0.3]$, for a total of $N_{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 10 \cdot 501 = 5010$ training snapshots.

Due to an inaccurate reconstruction error of the CAE for the first time instants, the predictions of the dynamics of the reduced model are evaluated from the time instant $t_0 = 0.01$ seconds. The test non-temporal parameters are $N^{\mu}_{\text{test}} = 8$ in number and sampled equispacedly in the test interval

$\mu \in [0.05, 0.35]$, for a total of $N_{\text{test}} = N^{\mu}_{\text{test}} \cdot N^{t}_{\text{test}} = 8 \cdot 475 = 3800$ test snapshots, since the first 26 are cut from the time series, $N^{t}_{\text{train}} = 501$ snapshots long. Again the first 2 and the last 2 parameters correspond to the extrapolation regime. The initial latent variables are obtained projecting with the encoder into the latent space the test snapshots corresponding to the time instants $t_0 = 0.01$ instead of $t = 0$. The training and test time series are labelled from 1 to 10 and from 1 to 8 with an increasing order.

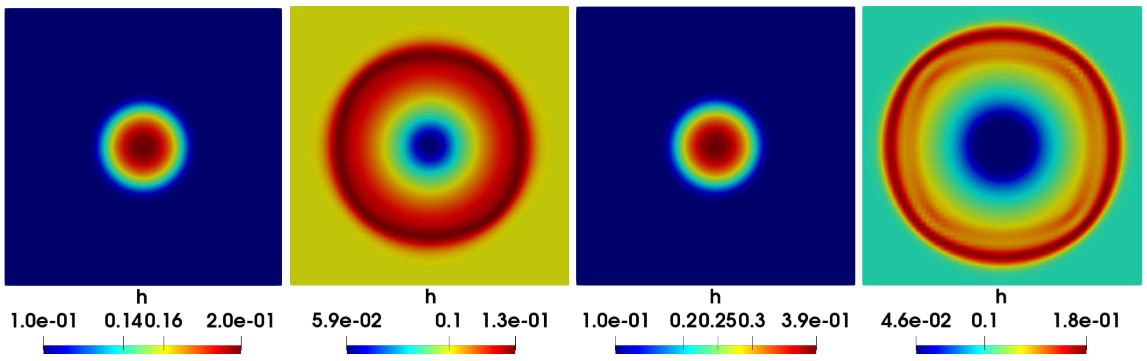


Fig. 6.10 **SWE.** From left to right: FOM solution of equation 6.36 at $(t, \mu) \in \{(0.01, 0.1), (0.2, 0.1)\}$ and $(t, \mu) \in \{(0.01, 0.3), (0.2, 0.3)\}$.

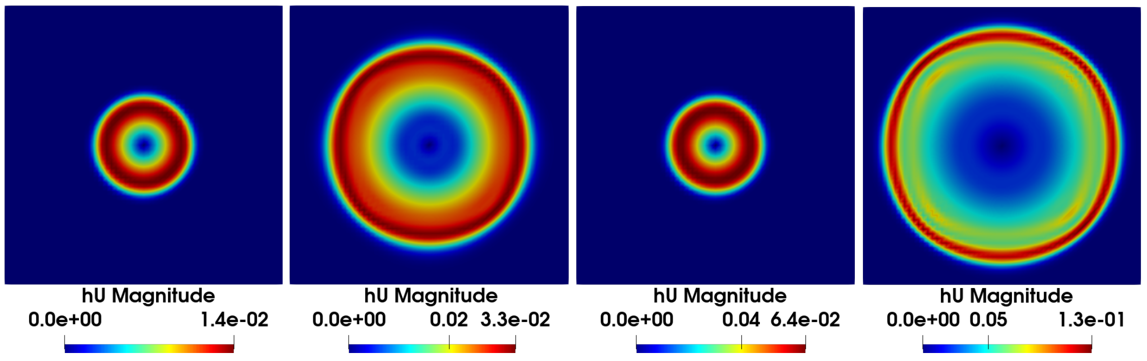


Fig. 6.11 **SWE.** From left to right: FOM solution of equation 6.36 at $(t, \mu) \in \{(.010, 0.1), (0.2, 0.1)\}$ and $(t, \mu) \in \{(0.01, 0.3), (0.2, 0.3)\}$.

In this test case the ROC hyper-reduction is more accurate with respect to the GNAT one, so the results are reported w.r.t. this hyper-reduction method.

Full-order model

One detail that we didn't stress in the previous test case is that the FOM and the NM-LSPG ROM can discretize the residuals of the SWE differently: only the consistency of the discretization is required, characterizing the NMLSPG family as equations-based rather than fully intrusive.

The FOM solutions are computed with the `OpenFoam` solver *shallowWaterFoam* [264], while the ROMs discretize the residual with a much simpler numerical scheme.

The FOM numerical scheme is the PIMPLE algorithm, a combination of PISO [197] (Pressure Implicit with Splitting of Operator) and SIMPLE [138] (Semi-Implicit Method for Pressure-Linked Equations). For the shallow water equations the free surface height h plays the role of the pressure in the Navier-Stokes equations, regarding the PIMPLE algorithm implementation. The number of outer PISO corrections is 3.

The time discretization is performed with the semi-implicit Euler method. The nonlinear advection terms are discretized with the Linear-Upwind Stabilised Transport (LUST) scheme that requires a stencil with 2 layers of adjacent cells for the hyper-reduction. The gradients are linearly interpolated through Gauss formula. The solutions for hU are obtained with Gauss-Seidel iterative method, and for h with the conjugate gradient method preconditioned by the Diagonal-based Incomplete Cholesky (DIC) preconditioner. For both of them the absolute tolerance on the residual is $1\text{e}-6$ and the relative tolerance of the residual w.r.t. the initial condition is 0.1.

The residual of ROMs is instead discretized as follows. If we represent with M_{hU}, M_h the mass matrices, with $G(h)$ the discrete gradient vector of h, and with $C_{hU}((hU)^{t-1})$, $C_h(U^{t-1})$ the advection matrices, then, at every time instant t, the discrete equations

$$\frac{M_{hU}}{\Delta t}(hU)^t + C_{hU}((hU)^{t-1})U^t + ghG(h) = \frac{M_{hU}}{\Delta t}(hU)^{t-1}, \tag{6.39}$$

$$\frac{M_h}{\Delta t}h^t + C_h(U^{t-1})h^t = \frac{M_h}{\Delta t}h^{t-1}, \tag{6.40}$$

are solved for the state $((hU)^t, h^t)$ with a semi-implicit Euler method. The same numerical schemes and linear systems iterative solvers of the FOM are employed. In principle, they could be changed.

Since now the stencil of a single cell needs two layers of adjacent cells for the discretizations, for each internal magic point, 12 additional cells need to be considered for the hyper-reduction.

Manifold Learning

The CAE architecture for the SWE model is reported in Table 6.6. This time one encoder and two decoders, one for the velocity U and one for the height h are trained. Moreover, to increase the generalization capabilities we converted 2 layers of the decoder for U in recurrent convolutional layers as shown in the section 6.4.3. Even with this modification the initial time steps, from 0 to 0.01 are associated to a high reconstruction error: the relative L^2-error is around 0.1 for every test parameter at the initial time instants, slowly decreasing towards the accuracy shown in Figure 6.12 after $t = 0.01$, chosen as initial instant from here onward.

The CAE is trained for 500 epochs with a batch size of 20 and an initial learning rate of $1\text{e}-4$, that halves if the loss from Equation 6.12 does not decrease after 50 epochs. In this case the state is (U, h) so the encoder has three channels, two for the velocity components, U_1, U_2, and one for the height, h. The high computational cost is shown in Table 6.4. We have to observe that nor the architecture is parsimonious for a good approximation of the discrete solution manifold in the time interval $[0.01, 0.2]$, neither the number of training snapshots 5010 is optimized to reach the highest efficiency with the lowest computational cost. Our focus is obtaining a satisfactory reconstruction error in order to build up our ROMs.

The solution manifold parametrized by the decoder achieves the reconstruction error of a linear manifold spanned by around 20 POD modes. In fact, the decay of the reconstruction error associated to the POD approximations is faster than the previous test case in the time interval $[0.01, 0.2]$.

It can be seen from the representation of the latent dynamics associated to the train parameters in Figure 6.13, that the initial solutions overlap. This and the low accuracy could be explained by the fact that the FOM dynamics has different scales, especially for the velocity U that from the initial constant zero solution reaches a magnitude of 10^{-1} meter per seconds. Further observations and possible solutions are presented in the Discussion section 6.5.

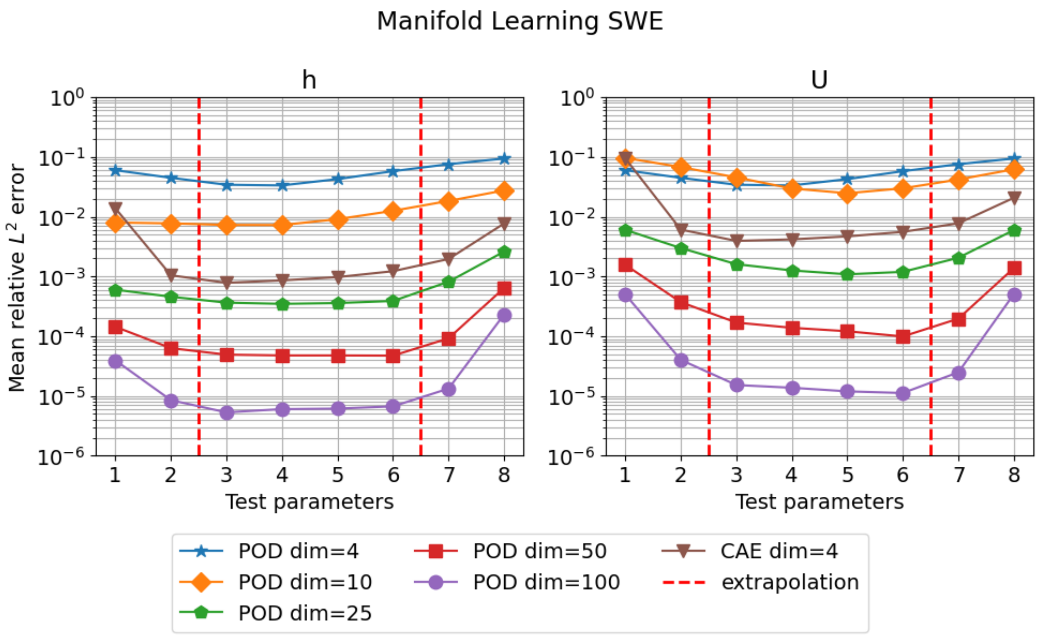


Fig. 6.12 **SWE.** Comparison between the CAE's and POD's projection errors of the discrete solution manifold represented by the $N^{\text{train}} = N^{\mu}_{\text{train}} \cdot N^{t}_{\text{train}} = 10 \cdot 501 = 5010$ training snapshots. The error is evaluated on the 8 test parameters, each associated with a time series of $N^{t}_{\text{test}} = 475$, for a total of $N_{\text{test}} = 3800$ test snapshots. The mean is performed over the time scale.

Hyper-reduction and teacher-student training

Mimicking the structure of the CAE, the compressed decoder is split in two, one for the velocity U and one for the free surface height h. The architecture for both the decoders is a feed-forward NN with a single hidden layer; they are reported in the Table 6.8. The compressed decoders are trained

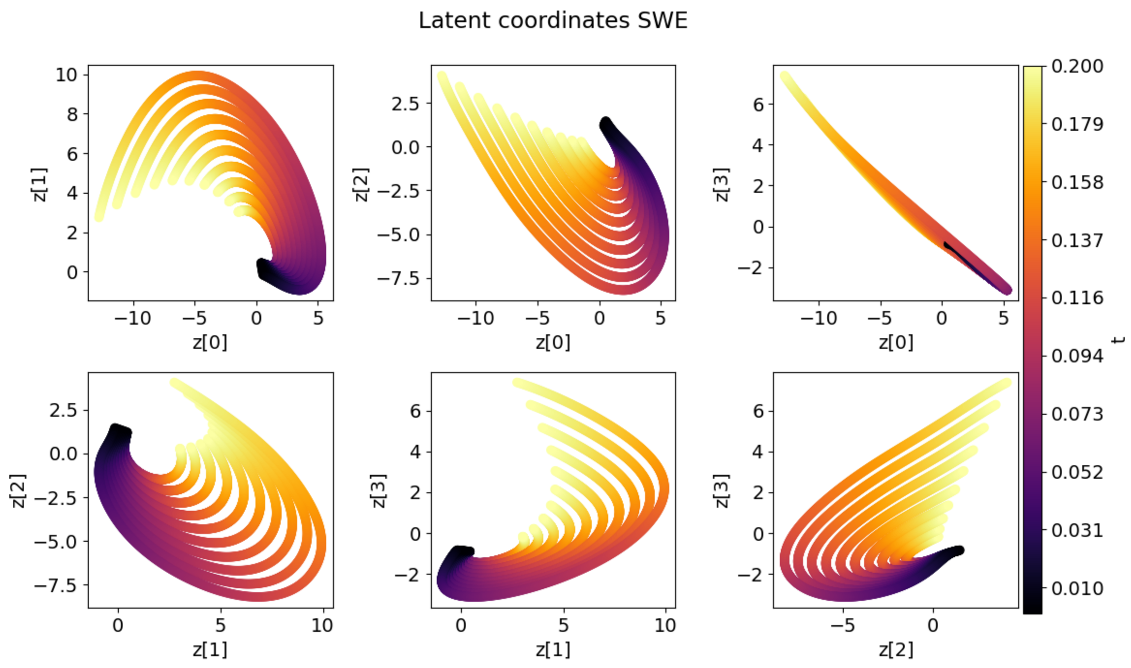


Fig. 6.13 **SWE.** Correlations among the 4 latent coordinates of the train set, 10 in total. They are obtained projecting the train snapshots into the latent space of the CAE with the encoder. The coloring corresponds to the time instants from 0. to 0.2, with a time step of $4e-4$ seconds. The initial time steps employed in the ROMs is 0.01 seconds.

for 1500 epochs, with a batch size of 20, and an initial learning rate of $1e-4$ that halves after 100 epochs if the loss does not decrease, with a minimum value of $1e-6$. As for the previous test case, the number of magic points, hidden layer sizes, training times, average training epoch computational cost are reported in Table 6.3. This time for each magic point correspond 3 degrees of freedom, so the actual output dimension of the compressed decoders is three times the submesh sizes.

The relative L^{∞}-error of the compressed decoder is shown in Figure 6.14. This time the extrapolation error is sensibly higher as already seen in the reconstruction error of the CAE.

Table 6.3 **SWE.** In this table are reported for the SWE test case: the submesh size and the hidden layer (HL) number of nodes of the compressed decoder; the Teacher-Student training (TS) duration in seconds, the Teacher-Student training average epoch duration in seconds; the average time step for NM-LSPG-GNAT with compressed decoder (GNAT-TS) in milliseconds, and the average time step for NM-LSPG-GNAT with the hyper-reduced residuals but full CAE decoder (GNAT-no-TS) in milliseconds. In red the average computational cost with 13 maximum residual evaluations due to the instability in the evolution of the dynamics of the test parameter 4.

MP	Submesh size	HL size	TS total epochs
25	238	600	582 [s]
50	345	600	1234 [s]
100	590	900	6428 [s]
150	654	900	6746 [s]

MP	TS avg epoch	avg GNAT-TS	avg GNAT-no-TS
25	0.388432 [s]	3.227 [ms]	14.782 [ms]
50	0.823084 [s]	4.062 [ms]	13.742 [ms]
100	4.285968 [s]	4.387 [ms]	15.238/22.688 [ms]
150	4.497579 [s]	4.307 [ms]	14.307 [ms]

As in the previous case, both the NM-LSPG-ROC and NM-LSPG-ROC-TS ROMs are considered. This time it's the NM-LSPG-ROC's dynamics to be less stable with 7 maximum residual evaluations of the LM algorithm. In this respect, for parameter test 4 and $\text{mp} = 100$ the maximum number of residual evaluations is increased to 13; in the error plots in Figures 6.15 and 6.16, only the value for parameter 4, $\text{mp} = 100$ is substituted. It is relevant to notice that the extrapolation error is lower for higher numbers of magic points and for the NM-LSPG-ROC method.

Comparison with data-driven predictions based on LSTM

The same LSTM architecture of the NCL test case is trained with the same training procedure, only that now there are 5010 training parameters-latent coordinates pairs. For the architecture's specifics see Table 6.9. The computational costs introduced in the previous test case are reported also for the SWE model in Table 6.4.

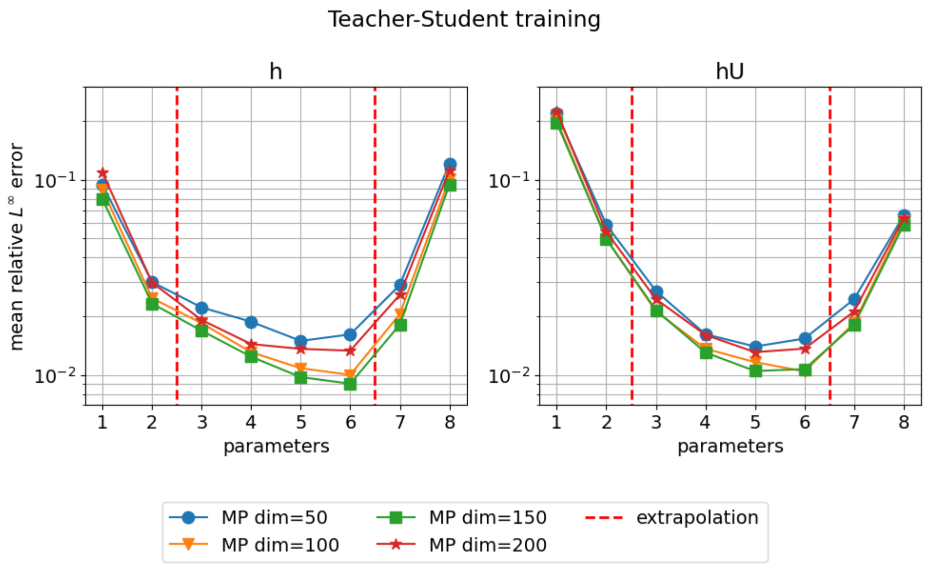


Fig. 6.14 **SWE.** Prediction accuracy on the test snapshots restricted to the magic points. The accuracy is measured with the relative L^∞-error averaged over the time trajectories associated to each one of the $N^{\mu}_{\text{test}} = 8$ test samples.

With the architecture and training procedure employed, the LSTM could not achieve a good accuracy at the initial time instants after $t_0 = 0.01$ seconds: for this reason in the plot of the errors in Figure 6.17 is reported both the mean over the whole test time series of 475 elements and over the time series after the 40-th element of the 475, that corresponds to the time instant 0.017 seconds. This issue should be ascribed at what we discussed in subsection 6.4.2, about the latent dynamics overlappings. Further remarks are provided in the Discussion section 6.5.

A part from this, the LSTM predictions are for almost every test parameter above only the NM-LSPG and CAE's reconstruction errors. Even in the extrapolation regimes, the predictions are more accurate than the NM-LSPG-ROC and NM-LSPG-ROC-TS reduced-order models. Regarding the computational costs, this time not only the LSTM model but also the NM-LSPG-ROC-TS ROMs achieve a little speedup w.r.t. the FOM. The choice of doubling the decoders has repercussions in the online costs, but it was made only in an effort to reach a good reconstruction error of the CAE; maybe more light architectures could be employed for the restricted time interval $[0.01, 0.2]$.

6.4.3 Neural networks' architectures

The architectures of the CAEs are shown in Tables 6.5, 6.6, of the compressed decoders in Tables 6.7 and 6.8, and of the LSTMs in Table 6.9. We mainly employ the Exponential Linear Unit (ELU) and Rectified Linear Unit (ReLU) activation functions. The padding is symmetric. The labels *Conv2d*, *ConvTr2d* and *ConvTr2dRec*, stand for 2d convolutions, transposed 2d convolutions [196], and 2d

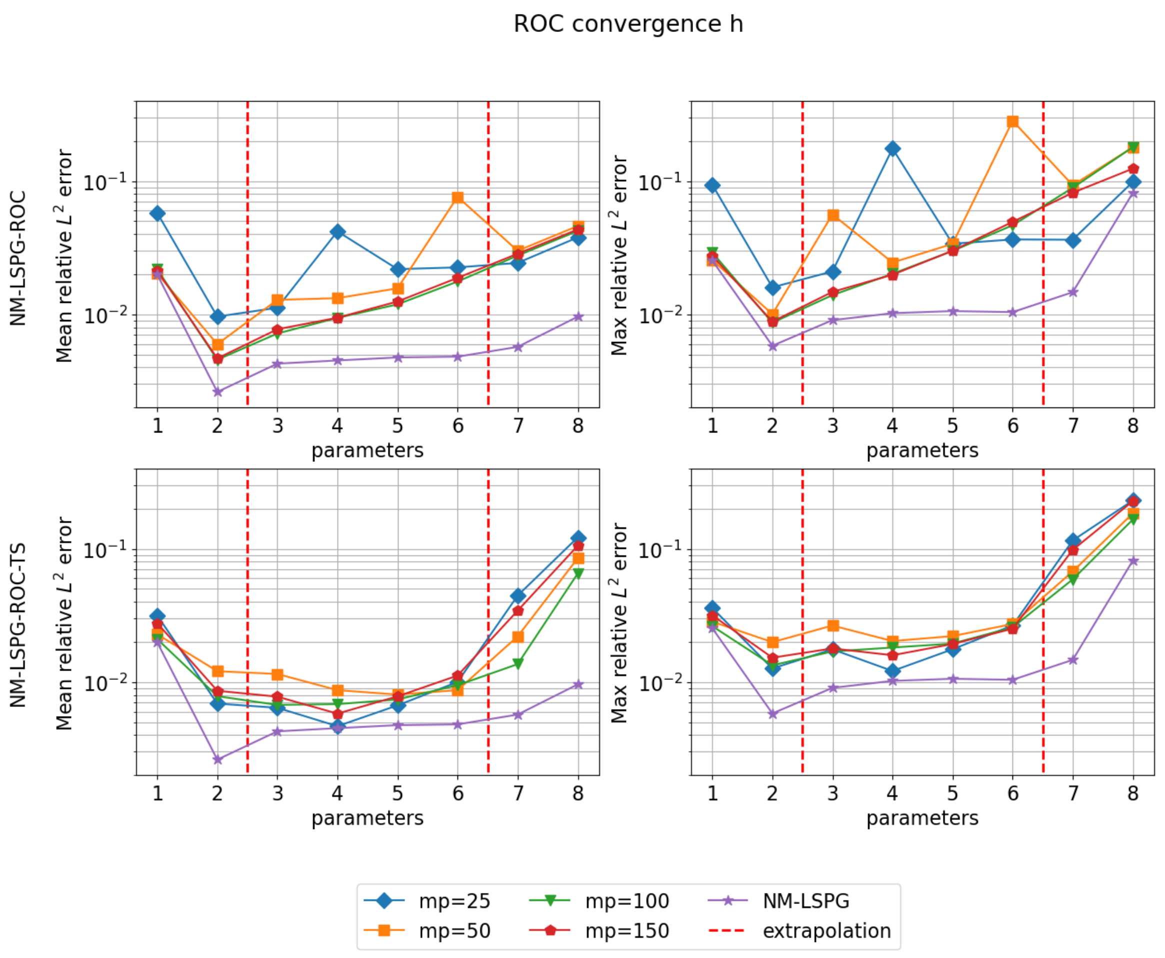


Fig. 6.15 **SWE.** First row: NM-LSPG-ROC mean and max relative L^2-error. Second row: NM-LSPG-ROC-TS mean and max relative L^2-error. In violet also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the 8 test parameters.

Table 6.4 **SWE.** Offline stage: full-order model (FOM) computation of the 5010 snapshots, average cost of a single epoch for the training of the CAE with a batch size of 20, and total cost for 500 epochs; average epoch's cost for the LSTM training with a batch size of 100, and total cost for 10000 epochs. Online stage: for the FOM, NM-LSPG and LSTM-NN models it is reported the average of a single time step cost over all the 8 test parameters and time series, and the average cost of the full dynamics over the 8 test parameters. The LSTM-NN full-dynamics is evaluated with a single forward.

Offline stage	Time
FOM snapshots evaluation	137.079881 [s]
CAE training single epoch avg	94.700484 [s]
CAE training	51431 [s]
LSTM-NN training single epoch avg	0.107722 [s]
LSTM-NN training	1076.150 [s]

Online stage	Time
avg FOM time step	7.251 [ms]
avg FOM full dynamics	116.024578 [s]
avg NM-LSPG time step	22.126 [ms]
avg NM-LSPG full-dynamics	336.171916 [s]
avg LSTM-NN time step	1.802 [ms]
avg LSTM-NN full-dynamics	6.834247 [s]

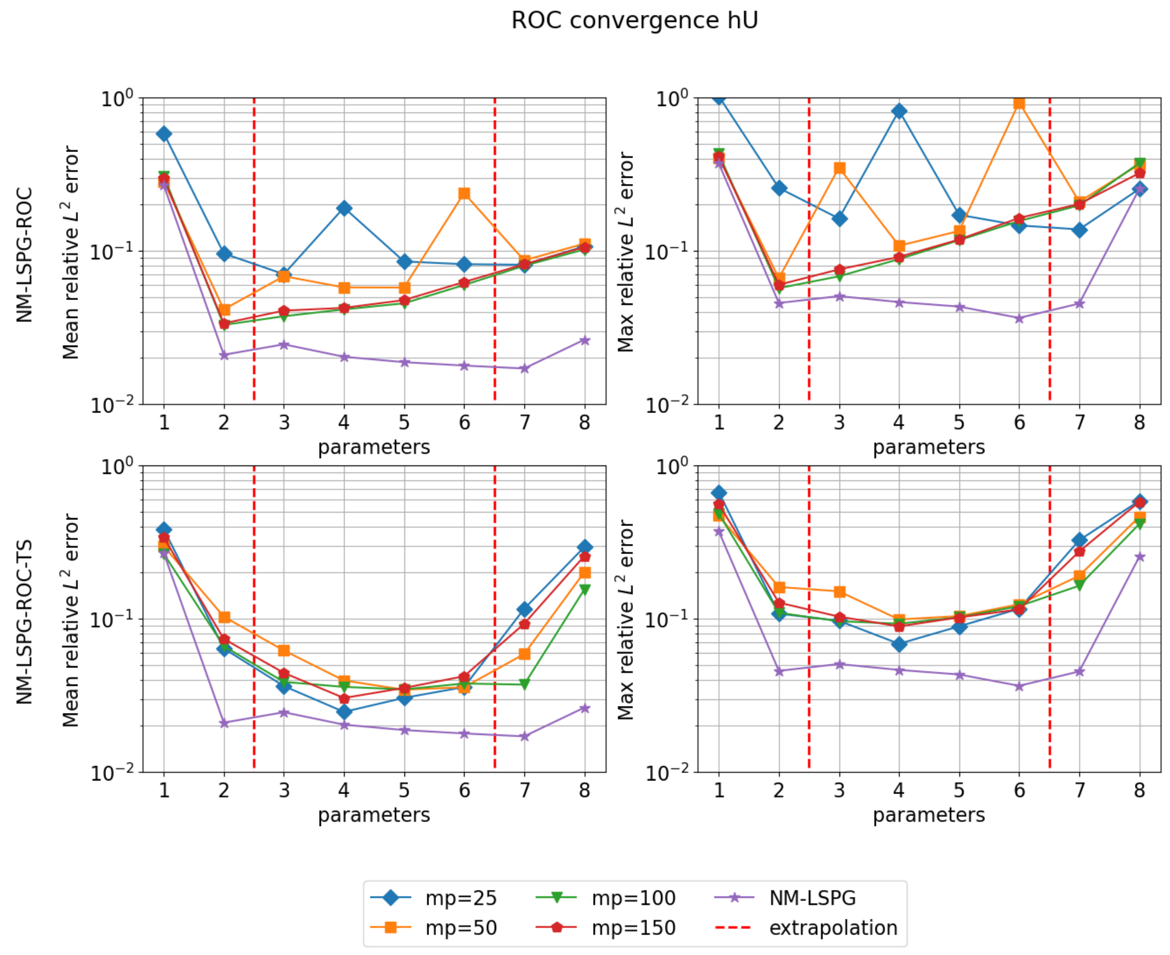


Fig. 6.16 **SWE.** First row: NM-LSPG-ROC mean and max relative L^2-error. Second row: NM-LSPG-ROC-TS mean and max relative L^2-error. In violet also the NM-LSPG accuracy is reported. The mean and max values are evaluated with respect to the time series of intermediate solutions associated to each one of the 8 test parameters.

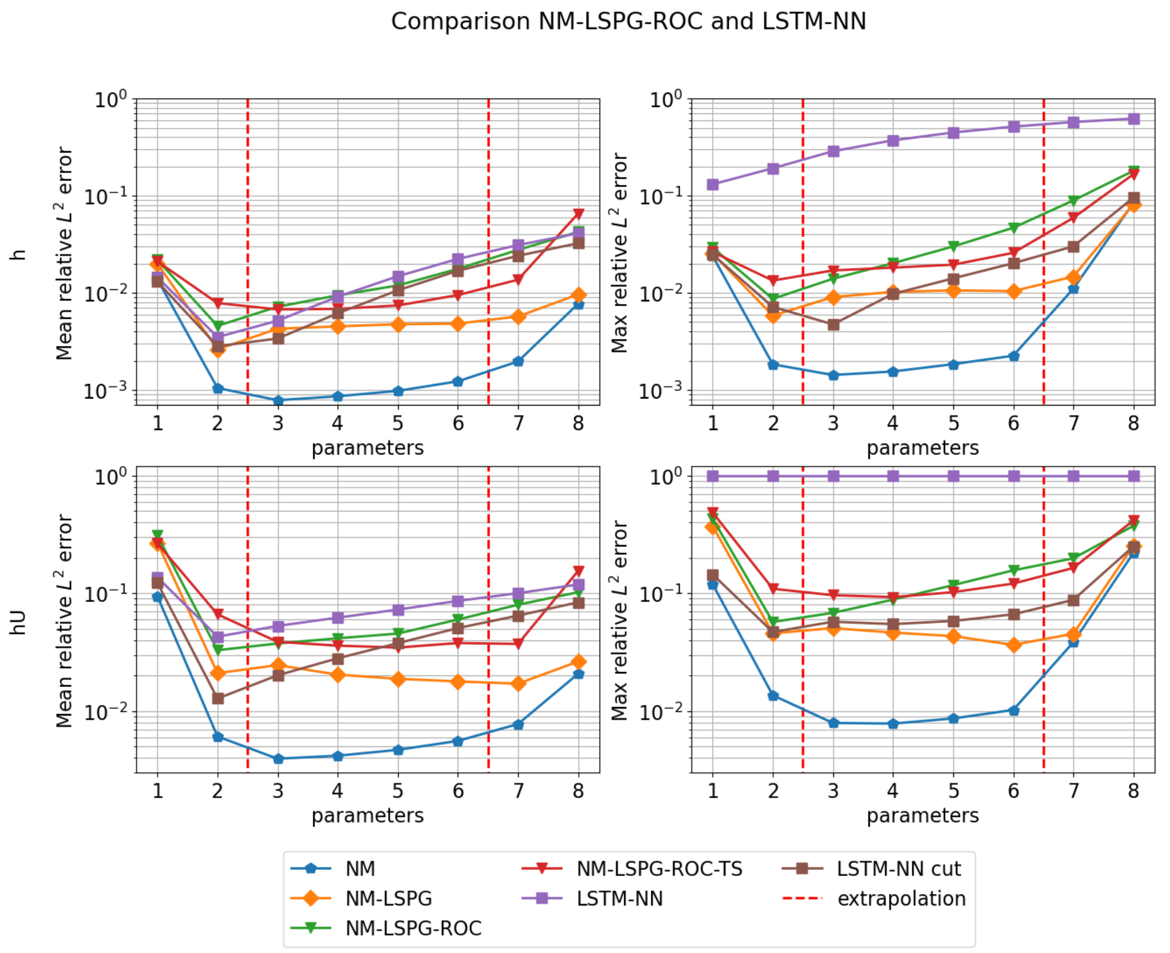


Fig. 6.17 **SWE.** Comparison of the accuracies between all the ROMs presented on the test set of 8 parameters, each associated to a time series of 475 intermediate solutions. The number of magic points employed for NM-LSPG-ROC and NM-LSPG-ROC-TS is 100. The mean and maximum relative L^2-errors are taken with respect to the time scale. Since the LSTM is inaccurate between the time instants 0.01 and 0.017 seconds we report also the mean over the time interval $[0.017, 0.2]$ witch label 'LSTM-NN cut', to establish a fair comparison.

transposed convolution associated to a recurrent layer, i.e. they are summed to the previous layer and then passed to an ELU activation function.

Table 6.5 **NCL.** Nonlinear Conservation Law model's Convolutional Autoencoder.

Encoder	Activation	Weights	Padding
Conv2d	ELU	[2, 8, 5, 5]	0
Conv2d	ELU	[8, 16, 3, 3]	1
Conv2d	ELU	[16, 32, 3, 3]	1
Conv2d	ELU	[32, 64, 3, 3]	1
Conv2d	ELU	[64, 128, 2, 2]	1
Linear	ELU	[1152, 4]	-

Decoder	Activation	Weights	Padding
Linear	ELU	[4, 1152]	-
ConvTr2d	ELU	[128, 64, 2, 2]	1
ConvTr2d	ELU	[64, 32, 3, 3]	1
ConvTr2d	ELU	[32, 16, 4, 4]	1
ConvTr2d	ELU	[16, 8, 4, 4]	0
ConvTr2d	ReLU	[8, 2, 4, 4]	1

Table 6.6 **SWE.** Shallow Water Equations model's Convolutional Autoencoder. The decoder for U has two recurrent layers.

Encoder	Activation	Weights	Padding
Conv2d	ELU	[2, 8, 5, 5]	0
Conv2d	ELU	[8, 16, 3, 3]	1
Conv2d	ELU	[16, 32, 3, 3]	1
Conv2d	ELU	[32, 64, 3, 3]	1
Conv2d	ELU	[64, 128, 2, 2]	1
Linear	ELU	[1152, 4]	-

Decoder h	Activation	Weights	Padding
Linear	ELU	[4, 1152]	-
ConvTr2d	ELU	[240, 120, 2, 2]	1
ConvTr2d	ELU	[120, 60, 3, 3]	1
ConvTr2d	ELU	[60, 30, 4, 4]	1
ConvTr2d	ELU	[30, 15, 4, 4]	0
ConvTr2d	ReLU	[15, 1, 4, 4]	1

Decoder U	Activation	Weights	Padding
Linear	ELU	[4, 2700]	-
ConvTr2d	ELU	[300, 75, 2, 2]	1
ConvTr2d	ELU	[150, 75, 3, 3]	1
ConvTr2dRec	-	[150, 75, 3, 3]	1
ConvTr2d	ELU	[75, 35, 4, 4]	1
ConvTr2d	ELU	[35, 20, 4, 4]	0
ConvTr2dRec	-	[35, 20, 4, 4]	0
ConvTr2d	-	[20, 2, 4, 4]	1

Table 6.7 **NCL.** Nonlinear Conservation Law model's Compressed Decoders.

Magic Points	1^{st} layer weight	1^{st} layer activation	2^{nd} layer weight	2^{nd} layer activation
50	[4, 300]	ELU	[300, 278]	ReLU
100	[4, 350]	ELU	[350, 492]	ReLU
150	[4, 400]	ELU	[400, 670]	ReLU

Table 6.8 **SWE.** Shallow Water Equations model's Compressed Decoders: height h and velocity U in order.

Magic Points	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
25	[4, 200]	ELU	[200, 238]	ReLU
50	[4, 200]	ELU	[200, 345]	ReLU
100	[4, 300]	ELU	[300, 590]	ReLU
150	[4, 300]	ELU	[300, 654]	ReLU

Magic Points	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
25	[4, 400]	ELU	[400, 478]	-
50	[4, 400]	ELU	[400, 960]	-
100	[4, 600]	ELU	[600, 1180]	-
150	[4, 600]	ELU	[600, 1308]	-

Table 6.9 Nonlinear Conservation Law model's and Shallow Water Equations model's Long-Shot Term Memory Neural Network.

	input dim	output dim	number of LSTM layers
LSTM layer	2	100	2

	1st layer weight	1st layer activation	2nd layer weight	2nd layer activation
Linear encoding layer	[100, 50]	ELU	[50, 4]	-

6.5 Discussion

We comment the numerical results obtained:

- Computational cost of the CAEs training. It is evident from Tables 6.2 and 6.7 that the offline stage's computational cost is dominated by the CAEs trainings. We have to remark that the architectures were not optimized to be the most parsimonious ones in order to achieve the desired reconstruction error. Moreover, libtorch training took almost twice more time than the same architecture's training in PyTorch, due to implementation inconsistencies. The cost of the forward evaluations of the decoder are comparable instead, not changing much the online costs. The training of the CAEs could be further reduced with transfer learning [106] or preprocessing steps that enlight some features of the dynamics that are more easily learnable, as was done in [95]. Also, the number of training snapshots could be optimized further for the test cases presented. The same observations apply also for the compressed decoders. Moreover, a parallel implementation of the training procedures on more than one GPU is mandatory to achieve competitive computational costs.

- Simultaneous CAE and compressed decoder/LSTM training. The additional costs of the LSTM and compressed decoder training could be cut with a unified training of the CAE and compressed decoder: after some epochs, the training of the LSTM or compressed decoder could be switched on and performed at the same time of the CAE's since the only additional information, apart from the restriction of the snapshots into the magic points, is the latent dynamics coordinates learned anyway during the CAE's optimization.

- Increasing the speedup of the nonlinear manifold ROMs. The bottleneck for the efficiency of the NM-LSPG-ROC-TS and NM-LSPG-ROC ROMs in the online stage is the cost of the compressed decoder or CAE's decoder forward. Regarding the NCL model, our results for the average time step of the NMLSPG-GNAT-TS and NM-LSPG-GNAT ROMs from Table 6.1 are comparable if not lower than the approximate time step of $7-8$ milliseconds for the NM-LSPG-GNAT with shallow autoencoders ROM presented in [146]. The difference is that now the FOM implemented with the FVM in `OpenFoam` [264] takes 2.42 seconds for 2000 time steps instead of 140.67 seconds for 1500 time steps in [146] with finite differences for a similar test case, i.e. 2d burgers equation instead of our nonlinear conservation law. To show a more consistent speedup w.r.t. the FOM, as for the SWE test case, the number of degrees of freedom could be increased without influencing the NM-LSPG-ROC-TS ROM since it is not dependent on the FOM's dimension. In a weaker sense also the NM-LSPG-ROC ROM is also independent on the number of degrees of freedom of the FOM, apart from the weights of the CAE's decoder that concur in increasing the cost of a single forward in the online stage.

- Generalization error of LSTM and CAE and additional inductive biases. The generalization error of the NN employed depends on a lot of factors, from the number of training samples to the regularization term in the loss, the architectures, etc. It is thus difficult to predict how much the accuracy of the predictions will decay outside the training range. Adding a physics-informed term in the loss of the CAE does not provide an improvement of the reconstruction error if enough training data are employed as in our test cases. Some regularization properties could be imposed in the latent dynamics to facilitate the evolution of the NMLSPG-ROC ROMs, for example imposing a linear latent dynamics could be beneficial

- Purely data-driven LSTM ROM vs NM-LSPG-ROC and NM-LSPG-ROC-TS ROMs. One crucial difference is interpretability: in the first case, the latent dynamics is obtained training the LSTM to approximate the latent coordinates, in the second case, the latent dynamics is evolved in time minimizing the hyper-reduced residual based on the physical model's equations. Regarding the LSTM predictions, we have seen in the test cases that despite the higher accuracy in the training range and the low computational online cost, there might be other issues in the extrapolation regime: in the NCL test case, the accuracy was sensitively lower in the extrapolation regime, even if it could be improved in theory increasing the layers and nodes of the LSTM in exchange for a higher training cost; in the SWE test case the initial time step from 0.01 to 0.017 seconds could not be well-approximated due to different scales and overlappings in the latent dynamics. The NM-LSPG-ROC and NM-LSPG-ROC-TS mitigated in some sense these issues, delegating less effort in the tuning of the hyperparameters of the LSTM and increasing the interpretability of the results. Moreover, the LSTM-NN, approximate the dynamics only every time step imposed by the training input-output pairs, while the nonlinear manifold ROMs, can in principle approximate the latent dynamics with an arbitrary small time step, since the decoder provides a continuous approximation of the solution manifold and the dynamics is evolved based on a numerical scheme with changeable time step. With respect to purely data-driven methods though, nonlinear manifold methods require a lower reconstruction error of the CAE since every successive ROMs' dynamics evolution depends on how well the intermediate solutions are reconstructed through the decoder.

- Learn the solution manifolds with autoencoders in unstructured meshes. The natural question of how to extend the CAE architecture to 3D or unstructured meshes, is being currently studied. In the literature there are already interesting results that employ graph neural networks and their variants to find latent representations of 3d simulations [201].

6.6 Conclusions and perspectives

We have developed two new hyper-reduced nonlinear manifold ROMs: NM-LSPG-ROC and NM-LSPG-ROC-TS, that can be converted in NM-LSPG-GNAT and NM-LSPG-GNAT-TS. In NMLSPG-ROC-TS the residuals of the NM-LSPG ROM are hyper-reduced with over-collocation, while the decoder of the CAE is approximated with teacher-student training into a compressed decoder. In NM-LSPG-ROC only the residuals' hyper-reduction is carried out. The methods perform similarly in accuracy and computational cost w.r.t. the NM-LSPG-GNAT with shallow autoencoders ROM introduced in [146], for a similar test case. The flexibility of our method permits to change the CAE architecture depending on the problem at hand, without imposing too many constraints on its structure, in order to reach the convergence faster and achieve a lower reconstruction error.

With respect to purely data-driven ROMs built on the CAE's solution manifold, NM-LSPG-ROC and NM-LSPG-ROC-TS provide more interpretable and, in the extrapolation regimes, sometimes more accurate predictions, and they need less hyperparameters tuning, once the CAE is trained. Moreover, the methods developed are equations-based rather than fully-intrusive, and exploit the physics of the model to evolve the latent dynamics. It is crucial, though, that the CAE's reconstruction error is sufficiently low, since the latent dynamics needs to be computed with numerical schemes that rely on the accuracy of the reconstructed solutions through the decoder of the CAE or the compressed decoder. We base these observations on the results obtained for two parametric nonlinear time-dependent benchmarks presented in the numerical results section 6.4, that is a 2d nonlinear conservation law model (NCL) and a 2d shallow water equations model. Despite the speedup is not achieved or not significant with respect to the FOMs, we reached satisfactory results in terms of the accuracy and the latent or reduced dimension of the ROMs.

Future directions of research involve the implementation of the developed ROMs in more complex applications with higher computational costs and degrees of freedom, such that a more evident speedup is reached. This may involve the development of more parsimonious architectures and training procedures to reduce the offline cost. The research in geometric deep learning will be crucial for the possible extensions of the present methodology to mesh-based 2d and 3d simulations. More has to be done also to further improve the interpretability of the results possibly taking into account a probabilistic approach, for example using Bayesian neural networks, and adhering more tightly to the physics of the model with additional inductive biases.

Chapter 7

Hyper-reduced nonlinear manifold method: adaptive strategies

A slow decaying Kolmogorov n-width of the solution manifold of a parametric partial differential equation precludes the realization of efficient linear projection-based reduced-order models. This is due to the high dimensionality of the reduced space needed to approximate with sufficient accuracy the solution manifold. To solve this problem, neural networks, in the form of different architectures, have been employed to build accurate nonlinear regressions of the solution manifolds. However, the majority of the implementations are non-intrusive black-box surrogate models, and only a part of them perform dimension reduction from the number of degrees of freedom of the discretized parametric models to a latent dimension. It is presented a new intrusive and explicable methodology for reduced-order modelling that employs neural networks for solution manifold approximation but that does not discard the physical and numerical models underneath in the predictive/online stage. The focus is on autoencoders used to compress further the dimensionality of linear approximants of solution manifolds, achieving in the end a nonlinear dimension reduction. After having obtained an accurate nonlinear approximant, the solutions on the latent manifold are sought with the residual-based nonlinear least-squares Petrov-Galerkin method, opportunely hyper-reduced in order to be independent of the number of degrees of freedom. New adaptive hyper-reduction strategies are developed along with the employment of local nonlinear approximants. The methodology is tested on two nonlinear time dependent parametric benchmarks involving a supersonic flow past a NACA airfoil and an incompressible turbulent flow around the Ahmed body.

Contents

7.1 Literature review

Real-world numerical models, coming from systems of partial differential equations (PDEs), usually study a physical phenomenon under the influence of different parameters. For each parametric instance, a single numerical simulation could take from hours to weeks to complete. Such is the case for complex fluid dynamics models or large-scale geophysical simulations. Fortunately, in some cases, the outputs of these models show evident correlations among them, partially because they follow the same physical laws embedded in the same numerical models, and partially because the parameters' dependency affects the solutions only as relatively small perturbations. Reduced-order modelling (ROM) leverages these correlations among snapshots, i.e. single solutions corresponding to different parametric instances, to reduce the computational time. The most successful model order reduction (MOR) methods combine the knowledge from the physical and the numerical models with the information coming from a database of solutions. One of the most employed methods is the reduced basis method [126, 228]. As most numerical models search for the solutions on discrete finite dimensional vector spaces like the finite volumes method (FVM), the finite element method (FEM), the spectral element method (SEM) and the discontinuous Galerkin method (DGM), model order reduction exploits the prior information coming from a dataset of training snapshots to update these ansatz spaces. The results are very low-dimensional linear vector spaces for which seeking the solutions associated with new parametric instances is more efficient if these solutions are expected to be correlated with the training dataset. Fundamentally, ROMs amortize the cost of computing an initial training database of solutions and low-dimensional adapted ansatz spaces in the offline stage, through subsequent efficient evaluations of unseen solutions in the online stage. It is important to remark that the numerical models employed in the offline stage are still employed also in the online stage so that the reduced solutions are discerned in the ansatz spaces through the satisfaction of the physical principles and mathematical constraints underneath the original numerical models.

Some difficulties arise when the solution manifold, that is the space of parameter dependent solutions, cannot be approximated with a satisfactory accuracy by linear low-dimensional spaces. If we consider a parameter space $\mathcal{P} \subset \mathbb{R}^p$, $p > 0$, and a solution map $\mathbf{U} : \mathcal{P} \subset \mathbb{R}^p \to X_h \sim \mathbb{R}^d$ that associates to each parameter $\boldsymbol{\mu} \in \mathcal{P}$ the corresponding solution $\mathbf{U}(\boldsymbol{\mu}) \in X_h \sim \mathbb{R}^d$ in the discretization space of choice X_h, where $d > 0$ is the number of degrees of freedom, we can quantify the linear approximability of the solution manifold $X_h \supseteq \mathcal{M} = \mathbf{U}(\mathcal{P})$ with the Kolmogorov n-width (KnW):

$$d_n(\mathcal{M}, X_h) = \inf_{\substack{W \subset \mathbb{R}^d \\ \dim W = n}} \sup_{\boldsymbol{\mu} \in \mathcal{P}} \inf_{\mathbf{V} \in X_h} \|\mathbf{V} - \mathbf{U}(\boldsymbol{\mu})\|_2. \tag{7.1}$$

A slow decaying Kolmogorov n-width with respect to the dimension of the linear approximant precludes the realization of efficient ROMs. One of the most prominent defects of linear ROMs is that even simple physical models, like linear advection, suffer from a slow Kolmogorov n-width decay. These are cases for which the snapshots are poorly correlated, sometimes almost orthogonal in X_h.

Recently, with the diffusion of scientific machine learning, black-box surrogate models have tackled slow-decaying KnW solution manifolds thanks to nonlinear approximants represented by neural networks (NNs), in the form of different combined architectures. The majority of these surrogate models, being non-intrusive, do not even perform dimension reduction from the space of degrees of freedom $\mathbb{R}^d$ to a reduced or latent space $\mathbb{R}^r$, $r \ll d$. While a low-dimensional space is needed for linear projection-based ROMs to seek the solutions efficiently in the online stage, surrogate models built with NNs rely on the fast evaluation of the nonlinear approximants for different inputs in the prediction phase. Apart from the imposition of additional inductive biases, the predicted solutions do not consider the physical and mathematical knowledge of the models under study: in fact, they are not obtained from the satisfaction of first principles like classical ROMs. Moreover, when dimension reduction is performed, it is essentially needed for feature extraction rather than to increase the efficiency of the surrogate models. Nonetheless, when architectures like autoencoders (AE) are employed, the approximation error of the solution manifolds decays more rapidly with respect to the latent dimension when compared to linear subspaces. This can be quantified with an extension of the definition of KnW δ_n for continuous maps:

$$\delta_n(\mathcal{M}, X_h) = \inf_{\substack{\psi \in \mathcal{C}(X_h, \mathbb{R}^r) \\ \phi \in \mathcal{C}(\mathbb{R}^r, X_h)}} \sup_{\boldsymbol{\mu} \in \mathcal{P}} \|\mathbf{U}(\boldsymbol{\mu}) - (\phi \circ \psi)(\mathbf{U}(\boldsymbol{\mu}))\|_2, \tag{7.2}$$

where $\psi \in \mathcal{C}(X_h, \mathbb{R}^r)$ and $\phi \in \mathcal{C}(\mathbb{R}^r, X_h)$ are continuous maps represented in our case by NNs. This enables the design of efficient intrusive ROMs even for models with a slow KnW decay.

The first employment of convolutional autoencoders for intrusive ROMs, namely Galerkin and least-squares Petrov-Galerkin nonlinear manifold methods appears in [161]. The major evident drawback is that both the architecture and the numerical schemes employed in the online predictive phase depend on the number of degrees of freedom (dofs), so the procedure itself is even slower than the full-order models. Typically, when performing MOR of nonlinear parametric PDEs hyper-reduction is employed to achieve independence with respect to the number of dofs. In this case, another ingredient complicates the matter since the nonlinearity coming from the decoder map needs also to be treated and made independent of the number of dofs. One of the first approaches in this direction is introduced in [146]. The architecture employed is a shallow masked autoencoder: the sparsity pattern imposed on the last decoder layers reduces the computational costs of the forward and Jacobian evaluations, while Gauss-Newton with approximated tensors (GNAT) is employed to hyper-reduce the residual. One problem that arises is that shallow autoencoders are sometimes not enough to accurately approximate complex solution manifolds and the methodology itself constrains the choice of architecture. The methodology was tested on the 2d Burgers equations solved with finite differences. Another strategy [223] uses teacher-student training to compress a generic architecture that performs dimension reduction, in this case a convolutional AE, onto a small feedforward NN. Possible combinations of hyper-reduction with reduced over-collocation [51] only on the residual or for both the decoder and the residuals were taken into consideration. The methodology was tested on a 2d nonlinear conservation law test case and a 2d shallow water equations benchmark solved with `OpenFoam` [177]. Afterward, it is introduced a new implementation [19] that considers as nonlinear approximant of the solution manifold the sum of a linear subspace and a linear closure term whose coefficients are the output of a feedforward NN. The hyper-reduction method is the energy-conserving sampling and weighting method (ECSW) and it was tested on a 2d Burgers' equations model solved with finite differences. Another approach [57], directly employs a relatively small decoder from the latent space to the submesh identified by the reduced over-collocation hyper-reduction method as nonlinear approximant of the solution manifold. In this way, the training phase is more efficient and the solutions are finally reconstructed with the hyper-reduction linear basis from the collocation nodes. To increase the accuracy of shallow masked AE, in [77] they implement domain decomposition and build a local shallow masked AE for each subdomain. The procedure is tested on a 2d Burgers' equation model.

In this work, we introduce a new methodology and we test it on more challenging benchmarks. For a moderately slow KnW decay, classical ROMs fail, but linear subspaces can still be employed as good approximants of the solution manifold. This is the rationale behind employing singular value decomposition modes (SVD) [95], or other linear transforms or filters, as a preprocessing step to dimension reduction with AE. In section 7.5, we will also show a case for which this assumption is not valid anymore and more deep NN architectures should be employed and reduced with teacher-student training following [223]. The novelties of our new approach are the following

- a new collocated hyper-reduction procedure specific for our nonlinear manifold approximant that combines randomized singular value decomposition modes with 1d convolutional autoencoders, in section 7.3
- an adaptive gradient-based hyper-reduction strategy, in section 7.3.2. Similar concepts of adaptation strategies are present in the literature [198, 267].

- the implementation of an efficient way to integrate local nonlinear manifolds with intrusive nonlinear least-squares Petrov-Galerkin through a local change of basis, in section 7.4
- the validation of our methodology on challenging test cases with moderately slow Kolmogorov n-width, in section 7.5

In summary, in section 7.2 the nonlinear least-squares Petrov-Galerkin method is presented, our nonlinear manifold approximant is introduced and some specifications regarding the normalization of the datasets and the evaluation of the randomized singular value decomposition (rSVD) modes are made. Then, in section 7.3 the employed hyper-reduction methods are introduced. Particular attention is focused on the reduced over-collocation method and its new adaptive formulation in subsection 7.3. A brief section 7.4 introduces a straightforward way to include local nonlinear manifolds in the methodology through a linear change of rSVD basis. Finally, two benchmarks are introduced in section 7.5. A 2d nonlinear parametric time-dependent supersonic compressible Navier-Stokes equations model is studied on a coarse and a finer mesh. A special focus is given to the comparison of different hyper-reduction techniques in subsection 7 and to the implementation of local nonlinear manifolds in subsection 7. The last subsections involve the study of a 3d nonlinear time-dependent geometrically parametrized turbulent incompressible Navier-Stokes equations model.

7.2 Residual-based ROMs on nonlinear manifolds

The starting point is a parametric time-dependent partial differential equation (PDE) on a computational domain $\Omega \subset \mathbb{R}^D$, $D = \{1,2,3\}$, with time interval $[0, T_{\boldsymbol{\mu}}]$ and parameter space $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$:

$$G(\mathbf{U}(\mathbf{x},t), \partial_t \mathbf{U}(\mathbf{x},t), \nabla \mathbf{U}(\mathbf{x},t), \dots; \boldsymbol{\mu}) = 0, \qquad (\mathbf{x},t) \in \Omega_{\boldsymbol{\mu}} \times [0, T_{\boldsymbol{\mu}}], \tag{7.3a}$$

$$B(\mathbf{U}(\mathbf{x},t); \boldsymbol{\mu}) = 0, \qquad (\mathbf{x},t) \in \partial\Omega_{\boldsymbol{\mu}} \times [0, T_{\boldsymbol{\mu}}], \tag{7.3b}$$

$$\mathbf{U}(\mathbf{x},t) = \mathbf{U}_{0,\boldsymbol{\mu}}(\mathbf{x}), \qquad (\mathbf{x},t) \in \Omega_{\boldsymbol{\mu}} \times \{0\}, \tag{7.3c}$$

where the state function $\mathbf{U} : \Omega_{\boldsymbol{\mu}} \times [0, T_{\boldsymbol{\mu}}] \subset \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}^s$, $s \geq 1$ belongs to a Banach space $X(\mathbb{R}^D \times \mathbb{R}; \mathbb{R}^s)$ for all $\boldsymbol{\mu} \in \mathcal{P}^q$, $q \geq 1$, of vector-valued time-dependent functions. The function $\mathbf{U}_{0,\boldsymbol{\mu}}(\mathbf{x})$ represents the possibly parametric dependent initial condition. The state function $\mathbf{U}$ is a synthetic notation that can include at the same time more than one physical field, like velocity, pressure, internal energy, and density for example. The function G represents the PDE itself and has as arguments the state function and its partial derivatives with respect to time and space. We do not restrict only to first order, higher derivatives are omitted. The boundary conditions are expressed through the operator B and we possibly parametric dependent. These definitions are introduced only to define the discretized systems we will work with. We have included also geometric parametrizations $\Omega_{\boldsymbol{\mu}} \subset \mathbb{R}^D$ through $\boldsymbol{\mu}$.

We will consider nonlinear time-dependent PDEs, but in general, the framework we are going to introduce can be applied also to stationary PDEs and linear PDEs. The only requirement is the slow

Kolmogorov n-width decay of the solution manifold, otherwise, it would be sufficient to apply the well-developed theory of linear projection-based ROMS. In fact, employing a nonlinear approximation of the solution manifold reduces the efficiency of linearly approximable solution manifolds, in general.

To be as general as possible, we will consider a generic discretization in space, with the constraints that it is supported on a computational mesh $\Omega_{\boldsymbol{\mu},h} \subset \mathbb{R}^D$ and the discretized differential operators have local stencils in order to implement efficiently hyper-reduction schemes later. So, in our framework, we include the Finite Volume Method (FVM), that we are employing, but also the Finite Element Method (FEM) and the Discontinuous Galerkin method (DGM), for example. Applying the method of lines, we discretize in space to obtain the following ordinary differential equation

$$G_h(\mathbf{U}_h(t), \partial_t \mathbf{U}_h(t), \nabla \mathbf{U}_h(t), \dots; \boldsymbol{\mu}) = \mathbf{0}, \quad t \in [0, T_{\boldsymbol{\mu}}], \tag{7.4a}$$

$$B_h(P_{\partial\Omega_{\boldsymbol{\mu}}}(\mathbf{U}_h(t)); \boldsymbol{\mu}) = 0, \quad t \in [0, T_{\boldsymbol{\mu}}], \tag{7.4b}$$

$$\mathbf{U}_h(0) = \mathbf{U}_{h,0}, \tag{7.4c}$$

where in this case the discrete state function $\mathbf{U}_h(t) \in X_h(\Omega_{\boldsymbol{\mu},h})$ belongs for all $t \in [0, T_{\boldsymbol{\mu}}]$ to a discretization space $X_h(\Omega_{\boldsymbol{\mu},h}) \sim \mathbb{R}^d$ where $d > 0$ is the number of degrees of freedom, with its norm $\|\cdot\|_{X_h}$. The map $P_{\partial\Omega_{\boldsymbol{\mu}}} : X_h(\Omega_{\boldsymbol{\mu},h}) \to X_h(\partial\Omega_{\boldsymbol{\mu},h})$ is the projection onto the discrete boundary $\partial\Omega_{\boldsymbol{\mu},h}$ of the computational domain $\Omega_{\boldsymbol{\mu},h}$.

Finally, we apply a discretization in time to obtain the discrete residual $G_{h,\delta t} : \mathcal{P} \times X_h \times X_h^{|I_t|} \to X_h$ at time $t \in \{t_0, \dots, t_{N_{\boldsymbol{\mu}}}\} = V_{\boldsymbol{\mu}}$

$$G_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s \in I_t}) = \mathbf{0}, \tag{7.5a}$$

$$B_{h,\delta t}(P_{\partial\Omega_{\boldsymbol{\mu}}}(\mathbf{U}_h^t); \boldsymbol{\mu}) = 0, \tag{7.5b}$$

$$\mathbf{U}_h^0 = \mathbf{U}_{h,0}, \tag{7.5c}$$

where the time instances $t_i \in [0, T_{\boldsymbol{\mu}}]$, $\forall i \in \{0, 1, \dots, N_{\boldsymbol{\mu}}\}$ and I_t is the set of previous time instances of the state variable $U_h^t \in X_h$ at time t, needed for the numerical time scheme of choice. For most cases of model order reduction, it is crucial that the discretization space $X_h \sim \mathbb{R}^d$ is not time or parametric dependent. With adaptive collocated hyper-reduction 7.3.2 these constraints can be relaxed.

7.2.1 Nonlinear least-squares Petrov-Galerkin method

We will introduce the nonlinear manifold least-squares Petrov-Galerkin method (NM-LSPG) [161] from its linear manifold version (LM-LSPG), see Figure 7.1. To perform model order reduction with LM-LSPG, we need to define a linear projection map $P_r : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$ from the reduced space $\mathbb{R}^r$, $r \ll d$ to the full-state space $X_h \sim \mathbb{R}^d$. We require that, fixed a tolerance $\varepsilon \ll 1$, the relative reconstruction error of the linear solution manifold $\mathcal{M} = \{\mathbf{U}_h \in X_h | \exists \boldsymbol{\mu} \in \mathcal{P} : \ g(\boldsymbol{\mu}) = \mathbf{U}_h\}$ is small:

$$\frac{\|\mathbf{U}_h - P_r P_r^T \mathbf{U}_h\|_2}{\|\mathbf{U}_h\|_2} < \varepsilon, \quad \forall \mathbf{U}_h \in \mathcal{M}. \tag{7.6}$$

Typically, this is achieved by sampling from the parameter space $\mathcal{P}$ a set of independent training parameters $\mathcal{P}_{\text{train}} \subset \mathcal{P}$. The corresponding training snapshots $\mathcal{U}_{\text{train}} = \{\mathbf{U}_{h,\boldsymbol{\mu}}\}_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}}}$ are employed to evaluate P_r. For every new parametric instance $\mu \in \mathcal{P}\backslash\mathcal{P}_{\text{train}}$, we can evaluate the corresponding solution solving the following nonlinear least-squares problem at each time step $t \in \{t_0,\dots,t_{N_{\boldsymbol{\mu}}}\} = V_{\boldsymbol{\mu}}$ in the reduced variables $\mathbf{z} \in \mathbb{R}^r$ and with initial condition $\mathbf{z}^0$

$$.\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|G_{h,\delta t}(\boldsymbol{\mu}, P_r(\mathbf{z}), \{P_r(\mathbf{z}^s)\}_{s\in I_t})\|^2_{X_h}, \tag{7.7a}$$

$$\mathbf{z}^0 = P_r^T(\mathbf{U}_{h,0}), \tag{7.7b}$$

where $\mathbf{z}^s \in \mathbb{R}^r$, $\forall s \in I_t$ are the previous reduced coordinates needed at time t by the numerical scheme. The nonlinear least-squares problem can be solved with optimization methods like Gauss-Newton with line-search [161], Levenberg-Marquardt [223] and derivative-free Pounders [270] implemented in PETSc [16], that we will employ.

If $G_{h,\delta t}$ is linear, then we can solve for (7.7a) without reconstructing the solution onto the full-state space $\mathbb{R}^d$. If $G_{h,\delta t}$ is nonlinear, hyper-reduction techniques must be introduced to recover the independence of the number of degrees of freedom.

The evolution of the reduced trajectory in the latent space $\mathbb{R}^r$ in Figure 7.1 is computed without reconstructing the full-states $U_h^t(\boldsymbol{\mu}) = P_r(\mathbf{z}(\boldsymbol{\mu}))$. The reconstruction from $\mathbb{R}^r$ to the ambient space $\mathbb{R}^d$ is performed only at the end with the projection map P_r.

The nonlinear counterpart of LM-LSPG, poses the approximability of the solution manifold $\mathcal{M}$ with a nonlinear manifold. We will define this approximating nonlinear manifold as the image of a nonlinear parametrization map $\phi : U \subset \mathbb{R}^r \to \mathbb{R}^d$, that is an atlas with a single chart, with U an open subset of $\mathbb{R}^r$. For our purposes, mainly linked to the definition of the initial conditions, we also need an approximation of the right-inverse of ϕ, that is $\psi : \phi(U) \to \mathbb{R}^r$ such that $\phi \circ \psi \approx I_d$.

There are many definitions that extend the notion of Kolmogorov n-width to nonlinear approximating spaces [75]. Our requirement is that the relative reconstruction error is below a fixed tolerance $\varepsilon \ll 1$:

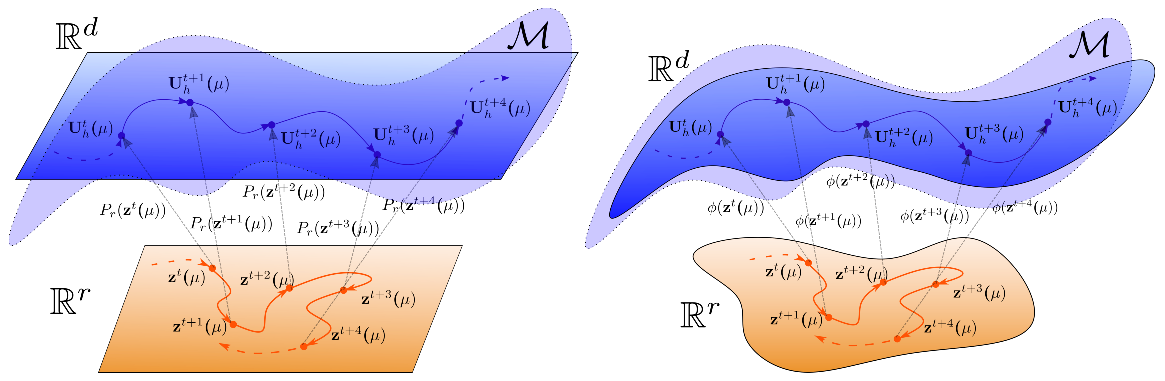
$$\frac{\|\mathbf{U}_h - (\phi\circ\psi)(\mathbf{U}_h)\|_2}{\|\mathbf{U}_h\|_2} < \varepsilon, \quad \forall \mathbf{U}_h \in \mathcal{M}. \tag{7.8}$$

The nonliear least squares problem solved for each time instance $t \in \{t_0,\dots,t_{N_{\boldsymbol{\mu}}}\} = V_{\boldsymbol{\mu}}$ is similar to the linear case with the substitution of the linear projection map P_r with ϕ and ψ:

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|G_{h,\delta t}(\boldsymbol{\mu}, \phi(\mathbf{z}), \{\phi(\mathbf{z}^s)\}_{s\in I_t})\|^2_{X_h}, \tag{7.9a}$$

$$\mathbf{z}^0 = \psi(\mathbf{U}_{h,0}). \tag{7.9b}$$

In this case, the sources of nonlinearity are the parametrization ϕ of the nonlinear approximation manifold and possibly also the residual $G_{h,\delta t}$. So, even if the residual $G_{h,\delta t}$ is linear we obtain a nonlinear least-squares problem to solve, due to the additional nonlinearity introduced with ϕ. As mentioned in the introduction, this is often a necessary step to overcome the problem of a slow


$\mathbb{R}^d$
$\mathcal{M}$
$\mathbf{U}_h^t(\mu)$
$\mathbf{U}_h^{t+1}(\mu)$
$\mathbf{U}_h^{t+2}(\mu)$
$\mathbf{U}_h^{t+3}(\mu)$
$\mathbf{U}_h^{t+4}(\mu)$
$P_r(\mathbf{z}^t(\mu))$
$P_r(\mathbf{z}^{t+1}(\mu))$
$P_r(\mathbf{z}^{t+2}(\mu))$
$P_r(\mathbf{z}^{t+3}(\mu))$
$P_r(\mathbf{z}^{t+4}(\mu))$
$\mathbb{R}^r$
$\mathbf{z}^t(\mu)$
$\mathbf{z}^{t+1}(\mu)$
$\mathbf{z}^{t+2}(\mu)$
$\mathbf{z}^{t+3}(\mu)$
$\mathbf{z}^{t+4}(\mu)$
$\phi(\mathbf{z}^t(\mu))$
$\phi(\mathbf{z}^{t+1}(\mu))$
$\phi(\mathbf{z}^{t+2}(\mu))$
$\phi(\mathbf{z}^{t+3}(\mu))$
$\phi(\mathbf{z}^{t+4}(\mu))$
Fig. 7.1 **Left:** evolution trajectories on the r-dimensional linear latent space and solution manifold embedded in the ambient space $\mathbb{R}^d$. The map P_r is a linear projection. **Right:** evolution trajectories on the r-dimensional latent space and nonlinear solution manifold in the ambient space $\mathbb{R}^d$. The map ϕ is a single chart nonlinear parametrization of the approximating solution manifold.

Kolmogorov n-width decay with a nonlinear approximating manifold that achieves a satisfactory accuracy with a lower latent/reduced dimension with respect to linear approximations.

Due to the nonlinearity, in general, solving (7.9a) is inefficient since the dependence on the number of degrees of freedom cannot be overcome. There are two factors that contribute to making the formulation (7.9a) not feasible as it was introduced in [161]. As for the linear case, the first is the nonlinearity of the residual $G_{h,\delta t}$, for which hyper-reduction techniques must be implemented. The second is the possibly expensive evaluation of ϕ and its dependence on the whole number of degrees of freedom d since the image of ϕ is contained in $\mathbb{R}^d$. So, hyper-reduction or similar techniques must be implemented also for the map ϕ, that in our case will be a neural network. See section 7.3 for more details on hyper-reduction and the next 7.2.2 for the definition of our nonlinear approximating solution manifold through the parametrization map ϕ.

7.2.2 Convolutional autoencoders: encodings and inductive biases

Applying SVD or principal component analysis (PCA) in machine learning jargon, to extract small dimensional and meaningful features from data is a technique largely employed in the data science community. After PCA, the new features can be used to train a neural network architecture more efficiently or for other purposes like clustering. This reasoning is applied also to data representing physical fields in model order reduction. One of the first examples is introduced by Ghattas et al. [186] in the context of inverse problems and model order reduction: the parameter-to-observable map is trained as a deep neural network (DNN) from the inputs reduced with active subspaces [60] to the outputs reduced with proper orthogonal decomposition (POD). In the context of model order reduction with autoencoders this technique is applied in [95].

We remark that an autoencoder with linear activation functions can reproduce the accuracy of truncated SVD if not also the principal modes. So, employing the SVD to extract meaningful features instead of adding a few linear neural network layers does not make the difference in terms of accuracy. What is crucial, especially for problems with a huge number of degrees of freedom, is the efficiency of SVD and its randomized version (rSVD) compared to the training of neural networks layers with linear activations.

Since we are considering physical fields supported on meshes we should not be limited to SVD: other compression algorithms possibly extracting meaningful features from spatial and temporal correlations are the Fourier and Wavelet transforms. In this context, model order reduction in frequency space is an active field of research [112, 184, 123]. Recently, also the Radon-Cumulative-Distribution (RCD) transform was applied to advection-dominated problems in model order reduction [165]. We will represent such generic transforms with $f_{\text{filter}} : X_h \sim \mathbb{R}^d \to \mathbb{R}^p$ and their approximate or true left inverse $f_{\text{filter}}^{-1} : \mathbb{R}^p \to X_h \sim \mathbb{R}^d$, such that $f_{\text{filter}}^{-1} \circ f_{\text{filter}} \approx I_d$.

In this work we will only consider rSVD to define the filtering maps f_{filter} and f_{filter}^{-1}, but the framework can be easily extended to other compression algorithms. The definitions of f_{filter} and f_{filter}^{-1} are reported in equation 7.18 of the next section. We will show some test cases where the number of rSVD modes needed to achieve a satisfactory accuracy reach 300, underlying truly slow Kolmogorov n-width applications, while in the literature only a moderate number of modes has been employed.

The neural network architectures we are going to use to define the maps $\phi : U \subset \mathbb{R}^r \to \mathbb{R}^d$ and $\psi : \phi(U) \to \mathbb{R}^r$ are reported in Tables 7.4 and 7.5 and shown in Figure 7.2. They are composed of standard 1d-convolutional layers since the filtered states $\tilde{U}_h = f_{\text{filter}}(\mathbf{U}_h)$ are not supported on a possible unstructured mesh anymore, but belong to the space of frequencies. So, in general, this approach is a viable alternative to graph neural networks [34] or other techniques to approximate physical fields supported on unstructured meshes.

To separate the application of the filtering/transforms maps f_{filter} and f_{filter}^{-1} from the convolutional neural networks layers, we define $\tilde{\psi} : \mathbb{R}^p \to \mathbb{R}^r$ and $\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^p$ through the relations $\phi = f_{\text{filter}}^{-1} \circ \tilde{\phi}$ and $\psi = \tilde{\psi} \circ f_{\text{filter}}$. So, the CNN layers are encapsulated in $\tilde{\phi}$ and $\tilde{\psi}$.

If the state vector $\mathbf{U}_h \in X_h$ includes more than one physical field, we have decided not to extract the frequencies or SVD reduced variables $\tilde{\mathbf{U}}_h$ for each field, but to do so all together in a monolithic fashion. This results in single-channel input of the CNN $\tilde{\psi}$ and a single-channel output of the CNN $\tilde{\phi}$, instead of having as many channels as the number of physical fields included in $\mathbf{U}_h$.

We remark that our CAE is trained only in the space of frequencies as input-output spaces achieving a relevant speedup thanks to this, as pointed out in [95]. The nonlinearity of the autoencoder is only exploited to further reduce the dimensionality from the frequency spaces and to directly approximate the solution manifold, which is effectively linear approximated in our case. For truly nonlinear solution manifold approximants we refer to [223].

7.2.3 Parallel Randomized Singular Value Decomposition

We recap in brief the procedure of randomized singular value decomposition (rSVD) [117], necessary to evaluate the modes when the snapshots matrix cannot be assembled altogether due to memory and

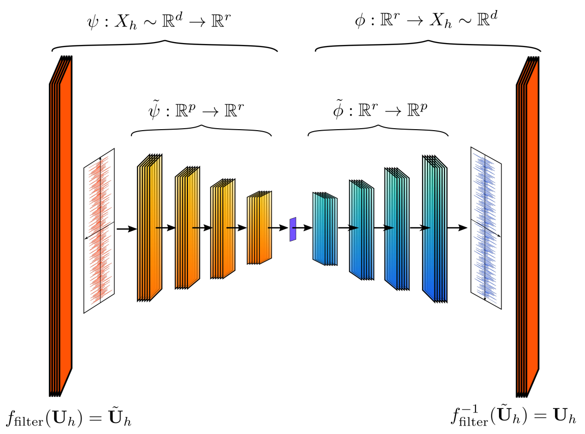

$\psi : X_h \sim \mathbb{R}^d \to \mathbb{R}^r$
$\phi : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$
$\tilde{\psi} : \mathbb{R}^p \to \mathbb{R}^r$
$\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^p$
$f_{\text{filter}}(\mathbf{U}_h) = \tilde{\mathbf{U}}_h$
$f^{-1}_{\text{filter}}(\tilde{\mathbf{U}}_h) = \mathbf{U}_h$

Fig. 7.2 The image shows schematically our implementation of the nonlinear maps $\phi = f^{-1}_{\text{filter}} \circ \tilde{\phi}$ and $\psi = \tilde{\psi} \circ f_{\text{filter}}$ that define the approximate nonlinear solution manifold. The functions $\tilde{\phi}$ and $\tilde{\psi}$ represent neural networks composed by subsequent 1d-convolutional and transposed 1d-convolutional layers, their specifics are reported in Table 7.4 and 7.5. The maps f_{filter} and f^{-1}_{filter} are the linear projection onto the first p rSVD modes and its transpose, their definition is reported in equation (7.18).

computational constraints. An alternative is represented by the frequent directions algorithm [100]. We remarkrSVDo that rSVD requires only matrix-vector evaluations and therefore can be applied also in a matrix-free fashion [137].

The only ingredient needed is the column-wise ordered matrix $A_{\text{train}} \in \mathbb{R}^{d \times n_{\text{train}}}$ of the training snapshots collection $\mathcal{U}_{\text{train}} := \{\mathbf{U}_{\boldsymbol{\mu},t}\}_{\boldsymbol{\mu} \in \mathcal{P}_{\text{train}},\, t \in V_{\boldsymbol{\mu}}}$, with $V_{\boldsymbol{\mu}} = \{t_1, \dots, t_{N_{\boldsymbol{\mu}}}\}$,

$$A_{\text{train}} = \left[\left(\begin{array}{cccc} | & | & | & | \\ \mathbf{U}_{\boldsymbol{\mu}_1,t_1} & \mathbf{U}_{\boldsymbol{\mu}_1,t_2} & \dots & \mathbf{U}_{\boldsymbol{\mu}_1,t_{N_{\boldsymbol{\mu}_1}}} \\ | & | & | & | \end{array}\right), \dots, \left(\begin{array}{cccc} | & | & | & | \\ \mathbf{U}_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_1} & \mathbf{U}_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_2} & \dots & \mathbf{U}_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_{N_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|}}}} \\ | & | & | & | \end{array}\right)\right] \in \mathbb{R}^{d \times n_{\text{train}}} \tag{7.10}$$

with $|\mathcal{U}_{\text{train}}| = n_{\text{train}}$ and $\mathbf{U}_{\boldsymbol{\mu},t} \in \mathbb{R}^d$ for all $\boldsymbol{\mu} \in \mathcal{P}_{\text{train}},\ t \in V_{\boldsymbol{\mu}}$. In our case, we do not assemble A_{train} since the computational domain can be partitioned and assigned to different processors during the evaluation of the full-order training solutions $\mathcal{U}_{\text{train}}$.

We represent with $M \in \mathbb{N}$ the number of cells $T \subset \Omega_h$ of the mesh $\{T_i\}_{i=1}^M \subset \mathcal{T}$ representing our discretized computational domain, and with $c \in \mathbb{N}$ the number of physical fields we are approximating: for the CNS test case $c = 4$ for the INS $c = 3$. The total number of degrees of freedom (dofs) is $d = c \cdot M$.

Since we will reduce with rSVD all the physical fields altogether in a monolithic fashion, we need to normalize the training snapshots with respect to the cell-wise measure $V \in \mathbb{R}^d$ (length, area or volume of each cell depending on the dimensionality of the mesh) and the different order of magnitudes and unit of measurement of the different physical fields considered.

For example, for the CNS test case $c=4$ we consider velocity $\mathbf{u}_{\boldsymbol{\mu},t}\in\mathbb{R}^M$, density $\rho_{\boldsymbol{\mu},t}\in\mathbb{R}^M$, internal energy $e_{\boldsymbol{\mu},t}\in\mathbb{R}^M$ and pressure $p_{\boldsymbol{\mu},t}\in\mathbb{R}^M$, so that

$$U_{\boldsymbol{\mu},t}=(\mathbf{u}_{\boldsymbol{\mu},t},\rho_{\boldsymbol{\mu},t},e_{\boldsymbol{\mu},t},p_{\boldsymbol{\mu},t})\in\mathbb{R}^{M\times c}=\mathbb{R}^d.$$

Similarly, for the INS test case $c=3$ we consider velocity $\mathbf{u}_{\boldsymbol{\mu},t}\in\mathbb{R}^M$, pressure $p_{\boldsymbol{\mu},t}\in\mathbb{R}^M$ and the turbulence viscosity $\nu_{\boldsymbol{\mu},t}\in\mathbb{R}^M$, so that

$$U_{\boldsymbol{\mu},t}=(\mathbf{u}_{\boldsymbol{\mu},t},p_{\boldsymbol{\mu},t},\nu_{\boldsymbol{\mu},t})\in\mathbb{R}^{M\times c}=\mathbb{R}^d. \tag{7.11}$$

The vector of cell-wise measures $\mathbf{V}\in\mathbb{R}^d$ is assembled from the vector $\mathbf{v}=\{\mathcal{L}_{\text{Lebesgue}}(T_i)\}_{i=1}^M$ where $\mathcal{L}_{\text{Lebesgue}}$ is the Lebesgue measure in $R^D\supset\Omega_h$. So that $\mathbf{V}=(\mathbf{v})_{i=1}^c\in\mathbb{R}^d$ is formed stacking $\mathbf{v}$ c-times. The pyhsical normalization field $\mathbf{N}\in\mathbb{R}^d$ is obtained from the maximum L^2-norm of each field: for the CNS test case we have

$$u_{\max}=\max_{\boldsymbol{\mu},t}\|\mathbf{u}_{\boldsymbol{\mu},t}\|_2,\quad \rho_{\max}=\max_{\boldsymbol{\mu},t}\|\rho_{\boldsymbol{\mu},t}\|_2,\quad e_{\max}=\max_{\boldsymbol{\mu},t}\|e_{\boldsymbol{\mu},t}\|_2,\quad p_{\max}=\max_{\boldsymbol{\mu},t}\|p_{\boldsymbol{\mu},t}\|_2, \tag{7.12}$$

so that $\mathbf{N}=(\mathbf{1}_{3M}\cdot u_{\max},\mathbf{1}_M\cdot\rho_{\max},\mathbf{1}_M\cdot e_{\max},\mathbf{1}_M\cdot p_{\max})\in\mathbb{R}^d$, with $\mathbf{1}_M$ the M-dimensional vector of ones. Similarly, for the INS test case we consider

$$u_{\max}=\max_{\boldsymbol{\mu},t}\|\mathbf{u}_{\boldsymbol{\mu},t}\|_2,\quad \rho_{\max}=\max_{\boldsymbol{\mu},t}\|\rho_{\boldsymbol{\mu},t}\|_2,\quad e_{\max}=\max_{\boldsymbol{\mu},t}\|e_{\boldsymbol{\mu},t}\|_2,\quad p_{\max}=\max_{\boldsymbol{\mu},t}\|p_{\boldsymbol{\mu},t}\|_2, \tag{7.13}$$

so that $\mathbf{N}=(\mathbf{1}_{3M}\cdot u_{\max},\mathbf{1}_M\cdot p_{\max},\mathbf{1}_M\cdot \nu_{\max})\in\mathbb{R}^d$.

So the columns $\{A^{\text{train}}_{\boldsymbol{\mu},t}\}_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}},\,t\in V_{\boldsymbol{\mu}}}$ of A_{train} are actually defined as

$$\mathbb{R}^d\ni A^{\text{train}}_{\boldsymbol{\mu},t}=\mathbf{U}_{\boldsymbol{\mu},t}\oslash\mathbf{W},\qquad \mathbf{W}=\mathbf{N}\oslash\mathbf{V}, \tag{7.14}$$

where we have considered only element-wise divisions $\bullet\oslash\bullet$ between vectors in $\mathbb{R}^d$. Notice that in this way we obtain unit-less states $\mathbf{U}_{\boldsymbol{\mu},t}/\mathbf{W}\in\mathbb{R}^d$. For the impact of physical normalization in model order reduction, see [193].

The reduced train and test rSVD coordinates are obtained with the following linear projection, employing the rSVD modes $U\in\mathbb{R}^{d\times r_{\text{rSVD}}}$ from Algorithm 7:

$$\mathbb{R}^{r_{\text{rSVD}}\times n_{\text{train}}}\ni Y_{\text{train}}=U^TA_{\text{train}},\qquad \text{(reduced train rSVD coordinates)} \tag{7.15a}$$

$$\mathbb{R}^{r_{\text{rSVD}}\times n_{\text{test}}}\ni Y_{\text{test}}=U^TA_{\text{test}},\qquad \text{(reduced test rSVD coordinates)} \tag{7.15b}$$

Algorithm 7: Parallelized Randomized Singular Value Decomposition.

```
input  : training column-wise ordered snapshots matrix $A_{\text{train}} \in \mathbb{R}^{d\times n_{\text{train}}}$,
         $r_{\text{rSVD}}$ reduced rSVD dimension, $p$ oversampling parameter
output : $U \in \mathbb{R}^{d\times r_{\text{rSVD}}}$ rSVD modes
1 Define the sketch dimension $l = r_{\text{rSVD}} + p$.
2 Draw the sketch matrix $\Omega \in \mathbb{R}^{n_{\text{train}}\times l}$ as a Gaussian random matrix.
3 Assemble **in parallel** $Y = A_{\text{train}}\Omega$, with $Y \in \mathbb{R}^{d\times l}$.
4 Evaluate the orthonormal basis $Q \in \mathcal{R}^{n_{\text{train}}}$ using $QR$-factorization $\mathbb{R}Y = QR$, with
    $R \in \mathbb{R}^{n_{\text{train}}\times n_{\text{train}}}$.
5 Project the snapshots into a random lower $l$-dimensional space **in parallel**: $S = Q^T A_{\text{train}}$,
    with $S \in \mathbb{R}^{l\times n_{\text{train}}}$.
6 Compute the thin SVD $S = \tilde{U}\tilde{\Sigma}\tilde{V}$, with $\tilde{U} \in \mathbb{R}^{l\times r_{\text{rSVD}}}$, $\Sigma \in \mathbb{R}^{r_{\text{rSVD}}\times r_{\text{rSVD}}}$, $\tilde{V} \in \mathbb{R}^{r_{\text{rSVD}}\times n_{\text{train}}}$.
7 Evaluate the rSVD modes $U = Q\tilde{U}$, with $U \in \mathbb{R}^{d\times r_{\text{rSVD}}}$.
```

and the reconstructed train and test fields are obtained employing the normalizing vector $\mathbf{W} = \mathbf{N} \oslash \mathbf{V}$:

$$\mathbb{R}^{d\times n_{\text{train}}} \ni A^{\text{rec}}_{\text{train}} = \mathbf{W} \odot UY_{\text{train}} = \mathbf{W} \odot UU^T A_{\text{train}}, \qquad \text{(reconstructed train snapshots)} \tag{7.16a}$$

$$\mathbb{R}^{d\times n_{\text{test}}} \ni A^{\text{rec}}_{\text{test}} = \mathbf{W} \odot UY_{\text{test}} = \mathbf{W} \odot UU^T A_{\text{test}}, \qquad \text{(reconstructed test snapshots)} \tag{7.16b}$$

where $\odot$ is the Hadamard columns-wise product, inverse operation of the normalization applied in (7.14).

To decide if the number of rSVD modes is sufficient to achieve the desired accuracy for the problem at hand we consider the mean and max relative L^2 reconstruction error on the training and test sets:

$$||A^{\text{rec}}_{\text{train}}||_{2,mean} = \frac{1}{n_{\text{train}}} \sum_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}},\, t\in V_{\boldsymbol{\mu}}} \frac{||\mathbf{U}^{\text{rec, train}}_{\boldsymbol{\mu},t} - \mathbf{U}^{\text{train}}_{\boldsymbol{\mu},t}||_2}{||\mathbf{U}^{\text{train}}_{\boldsymbol{\mu},t}||_2}, \qquad ||A^{\text{rec}}_{\text{train}}||_{2,max} = \max_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}},\, t\in V_{\boldsymbol{\mu}}} \frac{||\mathbf{U}^{\text{rec, train}}_{\boldsymbol{\mu},t} - \mathbf{U}^{\text{train}}_{\boldsymbol{\mu},t}||_2}{||\mathbf{U}^{\text{train}}_{\boldsymbol{\mu},t}||_2}, \tag{7.17a}$$

$$||A^{\text{rec}}_{\text{test}}||_{2,mean} = \frac{1}{n_{\text{test}}} \sum_{\boldsymbol{\mu}\in\mathcal{P}_{\text{test}},\, t\in V_{\boldsymbol{\mu}}} \frac{||\mathbf{U}^{\text{rec, test}}_{\boldsymbol{\mu},t} - \mathbf{U}^{\text{test}}_{\boldsymbol{\mu},t}||_2}{||\mathbf{U}^{\text{test}}_{\boldsymbol{\mu},t}||_2}, \qquad ||A^{\text{rec}}_{\text{test}}||_{2,max} = \max_{\boldsymbol{\mu}\in\mathcal{P}_{\text{test}},\, t\in V_{\boldsymbol{\mu}}} \frac{||\mathbf{U}^{\text{rec, test}}_{\boldsymbol{\mu},t} - \mathbf{U}^{\text{test}}_{\boldsymbol{\mu},t}||_2}{||\mathbf{U}^{\text{test}}_{\boldsymbol{\mu},t}||_2}, \tag{7.17b}$$

where $A^{\text{rec}}_{\text{train}} = (\mathbf{U}^{\text{rec, train}}_{\boldsymbol{\mu},t})_{\boldsymbol{\mu}\in\mathcal{P}_{\text{train}},\, t\in V_{\boldsymbol{\mu}}} \in \mathbb{R}^{d\times n_{\text{train}}}$ and $A^{\text{rec}}_{\text{test}} = (\mathbf{U}^{\text{rec, test}}_{\boldsymbol{\mu},t})_{\boldsymbol{\mu}\in\mathcal{P}_{\text{test}},\, t\in V_{\boldsymbol{\mu}}} \in \mathbb{R}^{d\times n_{\text{test}}}$.

Finally, we want to explicitly define the filtering/transform map $f_{\text{filter}} : X_h \sim \mathbb{R}^d \to \mathbb{R}^p$ and its approximate left inverse $f^{-1}_{\text{filter}} : \mathbb{R}^p \to X_h \sim \mathbb{R}^d$ with $p = r_{\text{rSVD}}$:

$$f_{\text{filter}}(\mathbf{U}_h) = U^T (\mathbf{U}_h \oslash \mathbf{W}), \quad f^{-1}_{\text{filter}}(\tilde{\mathbf{U}}_h) = (\mathbf{W} \odot U)\, \tilde{\mathbf{U}}_h. \tag{7.18}$$

7.3 Hyper-reduction

As introduced in section 7.2.1, there are two main problems that affect the efficient resolution of the nonlinear least squares problem in equation (7.9a) at each time instance $t \in V_{\boldsymbol{\mu}}$ and for each intermediate optimization step $i \in \{1,\dots,N_{t,\boldsymbol{\mu}}\}$ required by the nonlinear least-squares method, and they are both linked to the evaluation of the residual

$$G_{h,\delta t}(\boldsymbol{\mu},\phi(\mathbf{z}),\{\phi(\mathbf{z}^s)\}_{s\in I_t}). \tag{7.19}$$

We recall that we employ the derivative-free Pounders solver [270] implemented in PETSc [16]. Also, nonlinear least-squares problem optimizers that employ an approximation or the true Jacobian of the residual $G_{h,\delta t}$ can be employed.

The main problems to efficiently evaluate $G_{h,\delta t}$ are the following:

1. in general, the nonlinearity of $G_{h,\delta t}$ makes its evaluation dependent on the number of dofs d,
2. the map $\phi : U \subset \mathbb{R}^r \to \mathbb{R}^d$ might be computationally heavy to evaluate for each r-dimensional input and depends on the number of dofs since its output is d-dimensional.

The reason why we need this independence on the number of dofs is for our model order reduction procedure to be efficient even when d increases. Our two test cases *CNS* and *INS* have approximately $M = 30000$ and $M = 200000$ cells, and $d = 180000$ and $d = 1000000$ dofs respectively, which are still a moderate number of dofs compared to real applications. If we want to extend the methodology to larger meshes, we have to guarantee the independence of the number of dofs of our procedure.

We will address first the nonlinearity coming from $G_{h,\delta t}$. Typically, in the case of LM-LSPG, if the residual has a nonlinear term directly coming from the parametric PDE model, a class of methods under the name of hyper-reduction can be applied to ameliorate the situation. The idea is to reconstruct the residual only from its evaluations on a subset of degrees of freedom, in general. To do so, from the physical fields of the model considered taken as inputs, the values of those fields on the stencil needed by the numerical discretization have to be computed.

The simplest approach consists in collocating the residual on a subset of cells of the mesh, from now on called nodes or magic points. So we introduce two projection maps: the projection onto the magic points

$$P_{r_h} : \mathbb{R}^d \to \mathbb{R}^{r_h}, \quad P_{r_h}(\mathbf{U}_h) = \begin{pmatrix} | & | & | & | \\ \mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \dots & \mathbf{e}_{i_{r_h}} \\ | & | & | & | \end{pmatrix}^T \in \mathbb{R}^{r_h\times d}, \quad 0 < r_h \ll d \tag{7.20}$$

where $S_{r_h} = \{\mathbf{e}_{i_j}\}_{j=1}^{r_h}$ is a subset of the standard basis of $\mathbb{R}^d$ and the projection onto the submesh needed to evaluate the residual on the magic points

$$P^s_{r_h} : \mathbb{R}^d \to \mathbb{R}^s, \quad P^s_{r_h}(\mathbf{U}_h) = \begin{pmatrix} | & | & | & | \\ \mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \dots & \mathbf{e}_{i_s} \\ | & | & | & | \end{pmatrix}^T \in \mathbb{R}^{s\times d}, \quad 0 < r_h < s \ll d \tag{7.21}$$

where $S^s_{r_h} = \{\mathbf{e}_{i_j}\}_{j=1}^{s}$ is a subset of the standard basis of $\mathbb{R}^d$ containing $S_{r_h} \subset S^s_{r_h}$.

With these definitions, the residual from equation (7.9a) can be hyper-reduced as

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, P^s_{r_h}(\phi(\mathbf{z})), \{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}} \tag{7.22a}$$

$$= \underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}} \, \|P_{r_h} G_{h,\delta t}(\boldsymbol{\mu}, (P^s_{r_h} \circ f^{-1}_{\text{filter}})(\tilde{\phi}(\mathbf{z})), \{(P^s_{r_h} \circ f^{-1}_{\text{filter}})(\tilde{\phi}(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \tag{7.22b}$$

where we have employed the definitions of $\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^p$ and $f^{-1}_{\text{filter}} : \mathbb{R}^p \to \mathbb{R}^d$ from section 7.2.2.

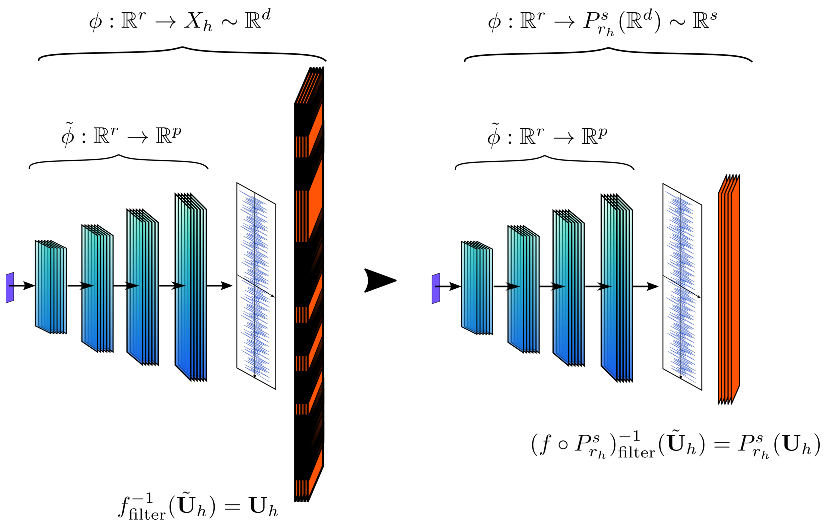
In this way, we have addressed the problem coming from the nonlinearity of the residual $G_{h,\delta t}$. At the same time, thanks to the choice of $\phi = f^{-1}_{\text{filter}} \circ \tilde{\phi}$ as composition of a linear projection depending on the dofs f^{-1}_{filter} and a small nonlinear neural network $\tilde{\phi}$ independent on the number of dofs, we have also tackled the second problem. In fact, also the parametrization map $(P^s_{r_h} \circ f^{-1}_{\text{filter}})(\tilde{\phi})$ restricted to the submesh is now independent on the number of dofs.

To be more specific, since we will be employing only rSVD as linear projections we have

$$f^{-1}_{\text{filter}}(\tilde{\mathbf{U}}_h) = P^s_{r_h} \circ \left((\mathbf{W} \odot U)\,\tilde{\mathbf{U}}_h\right) = \left(P^s_{r_h}(\mathbf{W}) \odot P^s_{r_h}(U)\right)\tilde{\mathbf{U}}_h. \tag{7.23}$$

where we have employed definition 7.18. So the hyper-reduction affects only the rSVD modes $U \in \mathbb{R}^{d\times p}$ and the normalization vector $\mathbf{W} \in \mathbb{R}^d$. A schematic representation of the hyper-reduced approximate nonlinear manifold parametrization map is shown in Figure 7.3.

We want to remark that the hyper-reduction procedure presented is effective thanks to the choice of implementation of the parametrization map ϕ through a combination of neural networks and rSVD modes. However, in some cases, the number of rSVD modes required $p = r_{\text{rSVD}}$ can become so large to guarantee a threshold accuracy in the relative L^2 reconstruction error that the methodology is no more efficient, since $p \gg 0$ and $\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^{r_{\text{rSVD}}}$. This computational burden affects both the offline stage for the training of the NN and the online residual evaluation. In these cases, one may want to employ heavier and deep generic neural network architectures like CNNs for structured meshes or GNNs for unstructured meshes that recover a good approximation of the solution manifold. The employment of these deep NNs brings up the problem of how to hyper-reduce them. The methodology introduced in this section cannot be applied, but a solution is represented by a strategy called teacher-student training for which in a following training phase a smaller fast *student* NN is tuned to replicate the results of the bigger slow *teacher* NN. This alternative approach is presented and studied in [223].



$\phi : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$
$\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^p$
$f_{\text{filter}}^{-1}(\tilde{\mathbf{U}}_h) = \mathbf{U}_h$
$\phi : \mathbb{R}^r \to P_{r_h}^s(\mathbb{R}^d) \sim \mathbb{R}^s$
$\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^p$
$(f \circ P_{r_h}^s)_{\text{filter}}^{-1}(\tilde{\mathbf{U}}_h) = P_{r_h}^s(\mathbf{U}_h)$

Fig. 7.3 **Left:** the decoder map $\tilde{\psi} : \mathbb{R}^r \to \mathbb{R}^p$ followed by the vector matrix multiplication with the rSVD modes $f_{\text{filter}}^{-1}(\tilde{\mathbf{U}}_h) : \mathbb{R}^p \to X_h \sim \mathbb{R}^d$. The latter are restricted to the submesh through the projection $P_{r_h}^s : X_h \sim \mathbb{R}^d \to \mathbb{R}^s$, that is the blackened dofs are discarded. **Right:** the map actually employed in the hyper-reduced nonlinear manifold least-squares Petrov-Galerkin method $\phi : \mathbb{R}^r \to P_{r_h}^s(X_h) \sim \mathbb{R}^s$. It is independent of the number of dofs and its evaluation is efficient thanks to the relatively small size of the decoder $\tilde{\psi}$.

7.3.1 Hyper-reduction methods and Magic Points Selection Algorithms

We are left with the task of defining the set of magic points S_{r_h} since the submesh $S_s \supset S_{r_h}$ is identified from the magic points and the choice of numerical scheme employed to discretize the parametric PDE. Until now, we have considered only the collocation of the residual on the magic points as hyper-reduction method. However, not only this is not the sole possible implementation but also not the most common one.

What we have actually described is part of the reduced over-collocation hyper-reduction method [51]. In this section, we will introduce also the gappy discrete empirical interpolation method (DEIM) [48], the energy-conserving sampling and weighting method (ECSW) [88], and DEIM with the quasi-optimal point selection algorithm (SOPT) from [156].

We have experimentally observed that for our test cases with slow Kolmogorov n-width and $r_{\text{rSVD}} = 150$ and $r_{\text{rSVD}} = 300$ rSVD modes, the reduced over-collocation method performs better. Further comments follow in section 7.6. Moreover, for test cases with a bigger number of dofs we have also developed a more successful adaptive magic points selection method introduced in section 7.3.2.

So, what we will be particularly focused on are the magic points selection algorithms from DEIM, ECSW and SOPT. The starting point for each one of them is the computation of a set of rSVD modes $U_{hr} \in \mathbb{R}^{d \times n_{hr}}$. Since we are hyper-reducing the residual $G_{h,\delta t}$ we need to collect a dataset of residual snapshots $\mathcal{G}_{\text{train}} = \{G_{h,\delta t}(\boldsymbol{\mu}, \mathbf{U}_h^t, \{\mathbf{U}_h^s\}_{s \in I_t})\}_{\boldsymbol{\mu} \in \mathcal{P}_{\text{train}},\, t \in V_{\boldsymbol{\mu}}} = \{G_{\boldsymbol{\mu},t}\}_{\boldsymbol{\mu} \in \mathcal{P}_{\text{train}},\, t \in V_{\boldsymbol{\mu}}}$, with $V_{\boldsymbol{\mu}} = \{t_1, \dots, t_{N_{\boldsymbol{\mu}}}\}$, in the training phase and compress them with rSVD:

$$G_{\text{train}} = \left[\begin{pmatrix} | & | & & | \\ G_{\boldsymbol{\mu}_1,t_1} & G_{\boldsymbol{\mu}_2,t_1} & \dots & G_{\boldsymbol{\mu}_1,t_{N_{\boldsymbol{\mu}_1}}} \\ | & | & | & | \end{pmatrix}, \dots, \begin{pmatrix} | & | & | & | \\ G_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_1} & G_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_1} & \dots & G_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|},t_{N_{\boldsymbol{\mu}_{|\mathcal{P}_{\text{train}}|}}}} \\ | & | & | & | \end{pmatrix}\right] \in \mathbb{R}^{d\times n_{\text{train}}}, \tag{7.24}$$

from this residual snapshots matrix G_{train} the rSVD modes are computed $U_G \in \mathbb{R}^{d\times r_G}$, $r_G > 0$ and utilzed for magic points sampling after setting $U_{hr} = U_G$. In this case r_G might possibly be different than r_{rSVD}.

However, employing the physical fields' rSVD modes $U_{hr} = U \in \mathbb{R}^{d\times r_{\text{rSVD}}}$ also to perform the magic points' sampling is a fast alternative [56], since no additional snapshots of the residual need to be collected, apart from those used to define the map ϕ, and the computational cost of an additional application of the rSVD algorithm, this time on the residual snapshots, is avoided.

Gappy discrete empirical interpolation

In the gappy DEIM algorithm, after the computation of the hyper-reduction basis $U_{hr} \in \mathbb{R}^{d\times r_{hr}}$ from the physical fields $A_{\text{train}} \in \mathbb{R}^{d\times n_{\text{train}}}$ or residual snapshots $G_{\text{train}} \in \mathbb{R}^{d\times n_{\text{train}}}$, U_{hr} is employed to find the magic points with a greedy algorithm: at each step, it is selected the cell of the mesh associated to the highest value of the hyper-reduction reconstruction error $\mathbf{r} \in \mathbb{R}^d$

$$\mathbf{r} = A_{\text{train}} - U(P_{r_h}U)^\dagger(P_{r_h}A_{\text{train}}), \qquad \text{or} \qquad \mathbf{r} = G_{\text{train}} - U_G(P_{r_h}U_G)^\dagger(P_{r_h}G_{\text{train}}) \tag{7.25}$$

where with the notation $(\bullet)^\dagger$ we represent the Moore-Penrose pseudo-inverse matrix and $P_{r_h} \in \mathbb{R}^{d\times\tilde{r}}$ is the intermediate projection matrix that evaluates a vector on the full-state space $\mathbb{R}^d$ on the magic points $S_{r_h} = \{\mathbf{e}_{i_j}\}_{j=1}^{\tilde{r}}$, $0 < \tilde{r} \leq r_h$, selected up to the considered step of the greedy algorithm. We remark that if vector-valued states $\mathbf{U} \in \mathbb{R}^d$ with $d = M\cdot c$, $c > 1$ are considered, it is selected the cell of the mesh that maximizes the sum over each of the c fields of the hyper-reduction reconstruction error.

For our implementation of the gappy DEIM greedy nodes selection algorithm we have chosen the one studied in [46] and applied to the Gauss-Newton tensor approximation (GNAT) hyper-reduction method that is more general than DEIM.

Once the magic points set $S_{r_h}=\{\mathbf{e}_{i_j}\}_{j=1}^{r_h}$ has been evaluated, the magic points and submesh projections P_{r_h}, $P^s_{r_h}$ are computed and the following variants of nonlinear hyper-reduced least-squares problem (7.9a) are solved for

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}}\ \|(P_{r_h}U)^{\dagger}\left(P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\oslash P_{r_h}\mathbf{W}\right)\|^2_{\mathbb{R}^{r_h}},\qquad \text{(FB-DEIM)} \tag{7.26a}$$

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}}\ \|(P_{r_h}U_G)^{\dagger}\left(P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\oslash P_{r_h}\mathbf{W}_G\right)\|^2_{\mathbb{R}^{r_h}},\qquad \text{(RB-DEIM)} \tag{7.26b}$$

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\operatorname{argmin}}\ \|P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}},\qquad \text{(C-DEIM)}, \tag{7.26c}$$

depending on the choice of hyper-reduction basis chosen $U_{hr}=U$ (FB-DEIM) or $U_{hr}=U_G$ (RB-DEIM) or if it is performed reduced over-collocation (C-DEIM). In the case of FB-DEIM and RB-DEIM, the residuals $G_{h,\delta t}$ are divided elment-wise by the normalization vectors $\mathbf{W}\in\mathbb{R}^d$ defined in (7.14) and $\mathbf{W}_G\in\mathbb{R}^d$ defined analogously from G_{train}.

A quasi-optimal nodes sampling method

As pointed out in [156], the matrix $P_{r_h}U_{hr}\in\mathbb{R}^{r_h\times r_{hr}}$ from the DEIM algorithm loses the orthogonality of its columns with respect to $U_{hr}\in\mathbb{R}^{d\times r_{hr}}$ that is crucial for the numerical stability of DEIM and to minimize the error in the L^2-norm of the hyper-reduction interpolation. In fact, the hyper-reduction L^2 error can be decomposed [48] as the sum of the best approximation error on the linear subspace in $\mathbb{R}^d$ spanned by the columns of $U_{hr}\in\mathbb{R}^{d\times r_{hr}}$ and the distance from the projection onto it

$$\|A_{\text{train}}-U_{hr}(P_{r_h}U_{r_h})^{\dagger}(P_{r_h}A_{\text{train}})\|_2^2=\|A_{\text{train}}-U_{hr}U_{hr}^TA_{\text{train}}\|_2^2+\|U_{hr}U_{r_h}^TA_{\text{train}}-U_{hr}(P_{r_h}U_{r_h})^{\dagger}(P_{r_h}A_{\text{train}})\|_2^2 \tag{7.27a}$$

$$=\|A_{\text{train}}-U_{hr}U_{hr}^TA_{\text{train}}\|_2^2+\|U_{hr}(P_{r_h}U_{r_h})^{\dagger}P_{r_h}(U_{r_h}U_{r_h}^T-I_d)A_{\text{train}}\|_2^2 \tag{7.27b}$$

$$=\|A_{\text{train}}-U_{hr}U_{hr}^TA_{\text{train}}\|_2^2+\|(P_{r_h}U_{r_h})^{\dagger}P_{r_h}(U_{r_h}U_{r_h}^T-I_d)A_{\text{train}}\|_2^2, \tag{7.27c}$$

where in the last step we have used the fact that $U_{hr}\in\mathbb{R}^{d\times r_{rh}}$ has orthonormal columns. Only the second term depends on the magic points selection, so the optimal strategy would be to minimize the solution of the least-squares problem

$$(P_{r_h}U_{r_h})^{\dagger}P_{r_h}(U_{r_h}U_{r_h}^T-I_d)A_{\text{train}}=\underset{X\in\mathbb{R}^{r_{hr}\times n_{\text{train}}}}{\operatorname{argmin}}\ \|P_{r_h}U_{r_h}X-P_{r_h}(U_{r_h}U_{r_h}^T-I_d)A_{\text{train}}\|_2^2, \tag{7.28}$$

that is to minimize the hyper-reduction L^2 error of the remaining part of $A_{\text{train}} \in \mathbb{R}^{d\times n_{\text{train}}}$ after the difference with its projection on the subspace spanned by the hyper-reduction basis $U_{hr} \in \mathbb{R}^{d\times r_{hr}}$.

A possible way, not necessarily optimal, to minimize (7.28) is to maximize the determinant of $P_{r_h}U_{r_h} \in \mathbb{R}^{r_h\times r_{hr}}$ and at the same time maximize its column orthogonality. In [231], they developed an efficient greedy algorithm to do so, maximizing

$$P^S_{r_h} = \underset{P\in\mathbb{R}^{r_h\times r_{hr}}}{\text{argmax}}\, \mathcal{S}(PU_{hr}), \qquad \mathcal{S}(PU_{hr}) = \left(\frac{\sqrt{\det\left((PU_{hr})^T PU_{hr}\right)}}{\prod_i^{r_{hr}} \|(PU_{hr})_i\|_2}\right)^{\frac{1}{r_{hr}}} \in [0,1], \tag{7.29}$$

where $\{(PU_{hr})_i\}_{i=1}^{r_{rh}}$ are the columns of $PU_{hr} \in \mathbb{R}^{r_h\times r_{hr}}$, assuming $\|(PU_{hr})_i\|_2 \neq 0, \forall i \in \{1,\dots,r_{hr}\}$. In particular, in [231] it is proved that $\mathcal{S}(PU_{hr}) = 1$ if and only if the columns of PU_{hr} are mutually orthonormal. The same quasi-optimal criterion is employed for hyper-reduction in [156], under the name of S-optimality (SOPT).

The DEIM algorithm with S-optimality magic points selection has the following formulation:

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\, \|(P^S_{r_h}U)^\dagger \left(P^S_{r_h}G_{h,\delta t}(\boldsymbol{\mu}, P^{S,s}_{r_h}(\phi(\mathbf{z})), \{P^{S,s}_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t}\right) \oslash P^S_{r_h}\mathbf{W}\big)\|^2_{\mathbb{R}^{r_h}}, \qquad \text{(FB-DEIM-SOPT)} \tag{7.30a}$$

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\, \|(P^S_{r_h}U_G)^\dagger \left(P^S_{r_h}G_{h,\delta t}(\boldsymbol{\mu}, P^{S,s}_{r_h}(\phi(\mathbf{z})), \{P^{S,s}_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t}\right) \oslash P^S_{r_h}\mathbf{W}_G\big)\|^2_{\mathbb{R}^{r_h}}, \qquad \text{(RB-DEIM-SOPT)} \tag{7.30b}$$

$$\mathbf{z}^t = \underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\, \|P^S_{r_h}G_{h,\delta t}(\boldsymbol{\mu}, P^{S,s}_{r_h}(\phi(\mathbf{z})), \{P^{S,s}_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}}, \qquad \text{(C-DEIM-SOPT)}, \tag{7.30c}$$

where $P^{S,s}_{r_h} \in \mathbb{R}^{s\times d}$ is the submesh projection corresponding to $P^S_{r_h} \in \mathbb{R}^{r_h\times d}$. Also in this case, depending on the choice of hyper-reduction basis chosen $U_{hr} = U$ (FB-DEIM-SOPT) or $U_{hr} = U_G$ (RB-DEIM-SOPT) or if it is performed reduced over-collocation (C-DEIM-SOPT), there are three different formulations.

Energy-conserving sampling and weighting method

Differently from DEIM, the ECSW hyper-reduction method finds a sparse integration formula to approximate the quantity of interest, if it is obtained through an integration on the computational domain $\Omega \subset \mathbb{R}^D$, like the residual $G_{h,\delta t}$ if it is calculated with the FVM, FEM or DGM.

The idea is to find a $\|\bullet\|_0$-sparse quadrature formula ($\|\mathbf{x}\|_0 = \#\{i \in \mathbb{N} | \mathbf{x}_i \neq 0\}$, $\forall \in \mathbb{R}^d$) such that the new weights $\mathbf{Q}^{hr} \in \mathbb{R}^d$ are sparse $\|\mathbf{Q}^{hr}\|_0 \ll 1$ and approximate up to a tolerance $0 < \tau \ll 1$ the sums of the training residual snapshots:

$$\mathbf{Q}^{hr} = \underset{\mathbf{Q}\in\mathbb{R}^d_+}{\text{argmin}} \|\mathbf{Q}\|_0 \quad \text{s.t.} \quad \|\mathbf{Q}\odot U_{hr} - \mathbf{c}\|^2_2 < \tau\|\mathbf{c}\|^2_2 \tag{7.31}$$

where $\mathbf{Q}\odot U_{hr}$ is the elment-wise multiplication of the quadrature weights vector $\mathbf{Q}\in\mathbb{R}^d$ with the columns of the training residual snapshots matrix $U_{hr}\in\mathbb{R}^{d\times r_{hr}}$, and $\mathbf{c}\in\mathbb{R}^{r_{hr}}$, $\mathbf{c}_j=\sum_{i=1}^d (U_{hr})_{ij},\ \forall j\in\{1,\dots,r_{hr}\}$ is the vector of integrals. With the notation $\mathbb{R}^d_+$ we consider the set of non-negative d-dimensional vectors. In practice, this NP-hard problem is relaxed to the following non-negative least-squares problem

$$\mathbf{Q}^{rh}=\underset{\mathbf{Q}\in\mathbb{R}^d_+}{\text{argmin}}\|\mathbf{Q}\odot U_{hr}-\mathbf{c}\|_2^2, \tag{7.32}$$

we solve it with the non-negative least-squares algorithm based on [157] and implemented in the Eigen library [111].

The nonlinear manifold least-squares problem (7.7a), is hyper-reduced with the ECSW method in the following formulations:

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\ \|\tilde{\mathbf{Q}}^{hr}\odot\left(P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\oslash P_{r_h}\mathbf{W}\right)\|^2_{\mathbb{R}^{r_h}},\qquad \text{(FB-ECSW)} \tag{7.33a}$$

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\ \|\tilde{\mathbf{Q}}^{hr}\odot\left(P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\oslash P_{r_h}\mathbf{W}_G\right)\|^2_{\mathbb{R}^{r_h}},\qquad \text{(RB-ECSW)} \tag{7.33b}$$

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\ \|P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}},\qquad \text{(C-ECSW),} \tag{7.33c}$$

where $\tilde{\mathbf{Q}}^{hr}\in\mathbb{R}^{r_h}$ is the quadrature weights vector obtained from the restriction of $\mathbf{Q}^{hr}\in\mathbb{R}^d$ to its non-zero entries. We also define $P_{r_h}\in\mathbb{R}^{d\times r_h}$ as the boolean matrix that selects the non-zero entries of $\mathbf{Q}^{hr}\in\mathbb{R}^d$ so that $\tilde{\mathbf{Q}}^{hr}=P_{r_h}\mathbf{Q}^{hr}\in\mathbb{R}^d$, and as a consequence the projection onto the submesh $P^s_{r_h}\in\mathbb{R}^{d\times s}$.

Also for this case, depending on the choice of hyper-reduction basis chosen $U_{hr}=U$ (FB-ECSW) or $U_{hr}=U_G$ (RB-ECSW) or if it is performed reduced over-collocation (C-ECSW), there are three different formulations.

7.3.2 Gradient-based adaptive hyper-reduction

The most successful hyper-reduction strategy for our implementation of the NM-LSPG method is the adaptive reduced over-collocation method (C-UP) that we introduce now. The method employs the standard formulation of the reduced over-collocation hyper-reduction method

$$\mathbf{z}^t=\underset{\mathbf{z}\in\mathbb{R}^r}{\text{argmin}}\ \|P_{r_h}G_{h,\delta t}(\boldsymbol{\mu},P^s_{r_h}(\phi(\mathbf{z})),\{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t})\|^2_{\mathbb{R}^{r_h}},\qquad \text{(C-UP)} \tag{7.34}$$

with the difference that the magic points are sampled adaptively during the time-evolution of the NM-LSPG trajectories. Its cost is amortized over the successive NM-LSPG time evaluations.

A heuristic approach relies on the positioning of the magic points where the sensitivities of the residual at time t have greater components in L^2-norm. If we define the residual map at time t

$$\mathbb{R}^r \ni \mathbf{z} \mapsto G_{h,\delta t}(\boldsymbol{\mu}, P^s_{r_h}(\phi(\mathbf{z})), \{P^s_{r_h}(\phi(\mathbf{z}^s))\}_{s\in I_t}) = G_{h,\delta t}(\phi(\mathbf{z})) \in \mathbb{R}^d, \qquad G_{h,\delta t} : \mathbb{R}^r \to \mathbb{R}^d, \quad (7.35)$$

losing for brevity the dependencies on the previous time steps, its sensitivities with respect to the latent coordinate $\mathbf{z}^t$ at time t are for all $i \in \{1,\dots,r\}$,

$$\mathbb{R}^d \ni \mathbf{J}_i = \left.\frac{\partial G_{h,\delta t}(\phi(\mathbf{z}))}{\partial \mathbf{z}_i}\right|_{\mathbf{z}=\mathbf{z}^t} = \left[\left.\frac{\partial G_{h,\delta t}(\phi(\mathbf{z}))}{\partial \boldsymbol{\phi}}\right|_{\phi(\mathbf{z})=\phi(\mathbf{z}^t)}\right]_{d\times s} \circ \left[\left.\frac{\partial \phi(\mathbf{z})}{\partial \mathbf{z}_i}\right|_{\mathbf{z}=\mathbf{z}^t}\right]_{s\times 1} \quad (7.36a)$$

$$= \left[\left.\frac{\partial G_{h,\delta t}(\phi(\mathbf{z}))}{\partial \boldsymbol{\phi}}\right|_{\phi(\mathbf{z})=\phi(\mathbf{z}^t)}\right]_{d\times s} \circ \left[P^s_{r_h} \circ f^{-1}_{\text{filter}}\right]_{s\times r_{\text{rSVD}}} \circ \left[\left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}_i}\right|_{\mathbf{z}=\mathbf{z}^t}\right]_{r_{\text{rSVD}}\times 1} \quad (7.36b)$$

$$= [\mathbf{G}]_{d\times s} [\mathbf{F}]_{s\times r_{\text{rSVD}}} [\boldsymbol{\Phi}_i]_{r_{\text{rSVD}}\times 1} \quad (7.36c)$$

with

$$\mathbf{G} = \left.\frac{\partial G_{h,\delta t}(\phi(\mathbf{z}))}{\partial \boldsymbol{\phi}}\right|_{\phi(\mathbf{z})=\phi(\mathbf{z}^t)} \in \mathbb{R}^{d\times s}, \quad \mathbf{F} = P^s_{r_h} \circ f^{-1}_{\text{filter}} \in \mathbb{R}^{s\times r_{\text{rSVD}}}, \quad \boldsymbol{\Phi} = \left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^t} \in \mathbb{R}^{r_{\text{rSVD}}\times r} \quad (7.37)$$

where with the chain rule we have highlighted the terms that compose the evaluation of the Jacobian matrix of the residual.

The exact evaluation of the Jacobian $\boldsymbol{\Phi} \in \mathbb{R}^{r_{\text{rSVD}}\times r}$ of the parametrization map $\tilde{\phi} : \mathbb{R}^r \to \mathbb{R}^{r_{\text{rSVD}}}$ is efficient enough to be employed by our adaptive hyper-reduction procedure. However, the matrix multiplication $\mathbf{G}\mathbf{F}\boldsymbol{\Phi}$ is inefficient since $\mathbf{G} \in \mathbb{R}^{d\times s}$ depends on the number of degrees of freedom and the submesh size s. If we wanted to apply DEIM to recover the residual on the full-space $\mathbb{R}^d$ from $P_{r_h} G_{h,\delta t}$, so considering the sensitivities of $U_{hr}(P_{r_h} U_{rh})^\dagger P_{r_h} G_{h,\delta t}$, we would need to compute the pseudo-inverse $(P_{r_h} U_{rh})^\dagger$ which could become a heavy task if repeated multiple times during the NM-LSPG method.

Our solution consists instead in evaluating $O : U \subset \mathbb{R}^r \times \mathcal{P} \to \mathbb{R}^{d\times r}$,

$$O(\mathbf{z}^t, \boldsymbol{\mu}) = f^{-1}_{\text{filter}} \circ \left(\left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^t} - \left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^{t-1}} \right) = [\mathbf{W} \odot U]_{d\times r_{\text{rSVD}}} \left[\left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^t} - \left.\frac{\partial \tilde{\phi}(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^{t-1}} \right]_{r_{\text{rSVD}}\times r}, \quad (7.38)$$

considering the current time-step t and the previous one $t-1$. In this way, we are employing the sensitivities of the neural network itself, rather than including also the information coming from the residual through the Jacobian $G \in \mathbb{R}^{d\times s}$.

We select only the first r_h cells of the computational domain that maximize

$$\underset{k=\{1,\dots,M\}}{\operatorname{argmax}} \left(\underset{j=1,\dots,r}{\operatorname{argmax}} \left(\sum_{i=1}^{c} O_i(\mathbf{z}^t, \boldsymbol{\mu})^2 \right)_{kj} \right) \quad (7.39)$$

where $O_i(\mathbf{z}^t, \boldsymbol{\mu}) \in \mathbb{R}^M$ for $i \in \{1,\dots,c\}$ are the degrees of freedom corresponding to the i-th physical field composing the full-state $\mathbf{U}_h \in X_h \sim \mathbb{R}^d$. When we update the magic points and consequently the submesh every n time-steps, we use the notation C-UPn, for example, C-UP50 for an adaption of the magic points every 50 time-steps.

The methodology presented derives from heuristic considerations and the necessity to keep the computational costs as low as possible. Experimentally it is shown that it is able to track the main moving features of our transient numerical simulations, thus adapting the magic points' position close to the most informative regions of the computational domain. See Figure 7.4. Additional collocation nodes are imposed on the boundaries to force the satisfaction of boundary conditions as can be noticed in Figure 7.4, at the inflow left boundary.

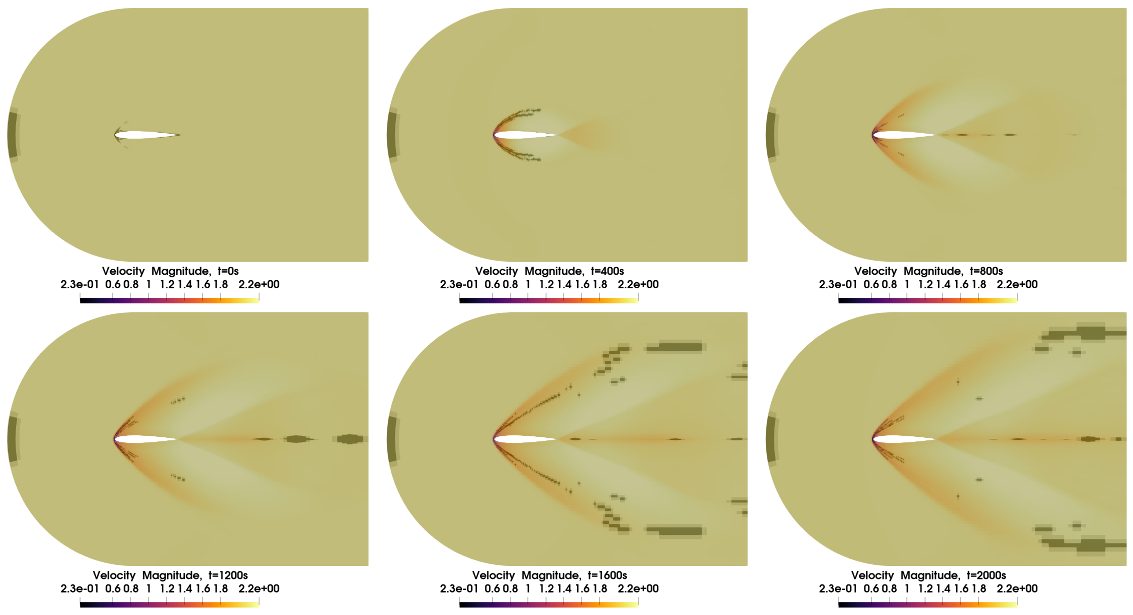


Fig. 7.4 Visual example of adaptive gradient-based reduced over-collocation. Predicted velocity magnitude time snapshots, for $t \in \{0, 400, 800, 1200, 1600, 2000\}$, of the 2d compressible Navier-Stokes equations' model described in subsection 7 for a value of the Mach number $\mathrm{Ma} = 2.12422656$, the fourth of five test parameters. It can be seen that the 700 collocation nodes in black, and the corresponding submesh in grey, updated with gradient-based adaptive hyper-reduction, through equation (7.39), every 50 time-steps, follow the transient dynamics. The time instants reported refer to the reference time interval $t \in [0, 2500]$, the real-time instants are obtained by applying the scaling reported in equation (7.43).

7.4 Local nonlinear manifold

Sometimes the L^2-norm relative reconstruction error (7.17a) cannot approximate with sufficient accuracy the nonlinear solution manifold in terms of reconstruction error, or the autoencoder architecture $\phi \circ \psi : \mathbb{R}^d \to \mathbb{R}^d$ or $\tilde{\phi} \circ \tilde{\psi} : \mathbb{R}^{r_{\mathrm{rSVD}}} \to \mathbb{R}^{r_{\mathrm{rSVD}}}$ does not meet the tolerance requirement (7.8). This happens when there are regions in the parameter space that correspond to less correlated solutions in the solution manifold.

A possible solution is to partition the parameter space in subdomains where the approximation properties of the rSVD modes and the autoencoder are satisfactory for the problem at hand. There are many implementations of local ROMs, often under the name of dictionary-based ROMs [69]. Generally, they also can be efficiently applied to our framework, thanks to the definition of our autoencoder through linear projections.

The setting is introduced only for two communicating local solution manifolds. For $i = 1, 2$, with the notations

$$\mathcal{P}^i_{n_{\text{train},i}} \subset \mathcal{P}_{n_{\text{train}}}, \qquad \text{(local parameter subset)} \tag{7.40a}$$

$$\phi^i : \mathbb{R}^r \to X_h \sim \mathbb{R}^d, \quad \tilde{\phi}^i : \mathbb{R}^r \to \mathbb{R}^p, \quad \psi^i : X_h \sim \mathbb{R}^d \to \mathbb{R}^r, \quad \tilde{\psi}^i : \mathbb{R}^p \to \mathbb{R}^r, \qquad \text{(local autoencoder)} \tag{7.40b}$$

$$f^i_{\text{filter}} : X_h \sim \mathbb{R}^d \to \mathbb{R}^p, \quad (f^i_{\text{filter}})^{-1} : \mathbb{R}^p \to X_h \sim \mathbb{R}^d, \qquad \text{(local linear projections)} \tag{7.40c}$$

$$A^i_{n_{\text{train},i}} \in \mathbb{R}^{d\times n_{\text{train},i}}, \quad \mathbf{W}^i \in \mathbb{R}^d, \quad U^i \in \mathbb{R}^{d\times r_{\text{rSVD}}} \qquad \text{(local rSVD quantities)} \tag{7.40d}$$

we denote the corresponding parametric subsets, the decoder maps, the encoder maps, the linear filter/transform maps, the training snapshots matrices, the normalizing vectors and the local rSVD basis of the two local solution manifolds. In principle, the local latent dimensions $r = r^1 = r^2$ with our notation, can be different $r^1 \neq r^2$ and the same is valid for $p = r_{\text{rSVD}} = p^1 = p^2$, that is $p^1 = r^1_{\text{rSVD}} \neq r^2_{\text{rSVD}} = p^2$. In fact, it is possible to adapt the latent and linear filter dimensions of the nonlinear approximating manifold parametrized by ϕ^i to the local Kolmogorov n-width decay of the subset of the parameter space under consideration. In a sense, the rationale is similar to the one behind hp-FEM or hp-DGM methods.

Gluing two local solution manifolds requires care especially because it may not be guaranteed that the corresponding full-states are close to each other with sufficient accuracy $((f^2_{\text{filter}})^{-1} \circ f^2_{\text{filter}} \circ (f^1_{\text{filter}})^{-1})(\tilde{\mathbf{U}}^1_h) \approx (f^1_{\text{filter}})^{-1}(\tilde{\mathbf{U}}^1_h)$. Many techniques have been developed to tackle this problem, but, for the moment, we will study only the most simple one. In fact, a possible way to avoid this consists in just overlapping the training datasets $A_{n_{\text{train},1}} \in \mathbb{R}^{d\times n_{\text{train},1}}$ and $A_{n_{\text{train},2}} \in \mathbb{R}^{d\times n_{\text{train},2}}$, that is considering a bigger intersection of their corresponding parameters subsets $\mathcal{P}^1_{n_{\text{train},1}} \subset \mathcal{P}_{n_{\text{train}}}$ and $\mathcal{P}^2_{n_{\text{train},2}} \subset \mathcal{P}_{n_{\text{train}}}$, $\mathcal{P}^1_{n_{\text{train},1}} \cap \mathcal{P}^2_{n_{\text{train},2}} \neq \varnothing$.

In our case, we have that the change of basis linear map $f^{12}_{\text{filter}} = f^2_{\text{filter}} \circ (f^1_{\text{filter}})^{-1} : \mathbb{R}^{r_{\text{rSVD}}} \to \mathbb{R}^{r_{\text{rSVD}}}$ is computed offline as

$$\tilde{\mathbf{U}}^2_h = f^{12}_{\text{filter}}(\tilde{\mathbf{U}}^1_h) = (f^2_{\text{filter}} \circ (f^1_{\text{filter}})^{-1})(\tilde{\mathbf{U}}^1_h) = (U^2)^T \left(\left(\left(\mathbf{W}^1 \odot U^1\right) \tilde{\mathbf{U}}^1_h\right) \oslash \mathbf{W}^2\right). \tag{7.41}$$

This change of basis between communicating local solution manifolds is represented schematically in Figure 7.5.

We will consider only two local solution manifolds such that only the time interval is partitioned, see section 7.

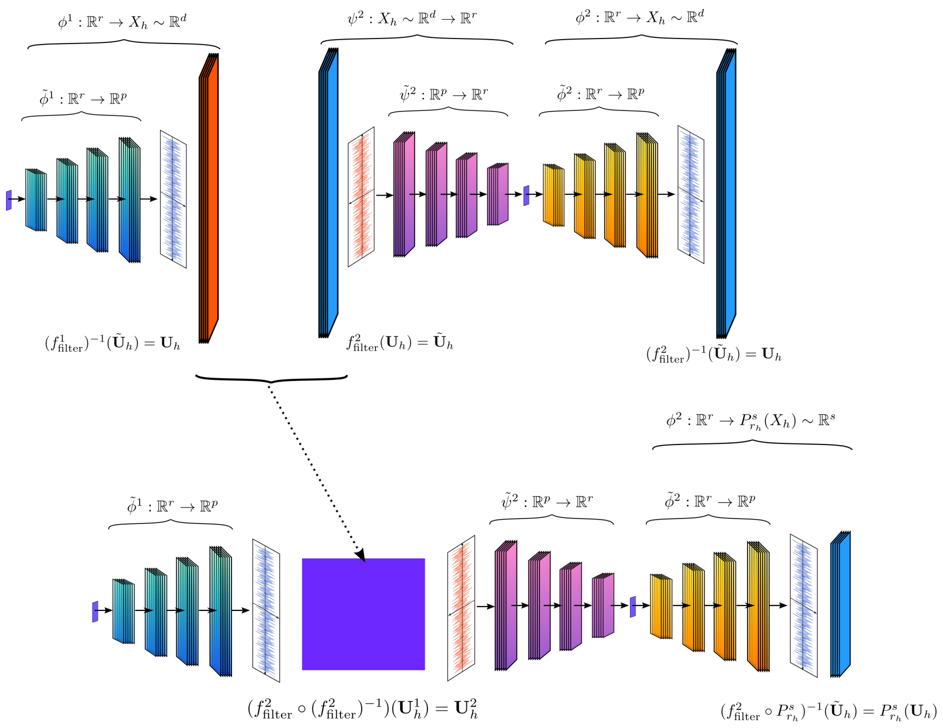

$\phi^1 : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$
$\tilde{\phi}^1 : \mathbb{R}^r \to \mathbb{R}^p$
$\psi^2 : X_h \sim \mathbb{R}^d \to \mathbb{R}^r$
$\tilde{\psi}^2 : \mathbb{R}^p \to \mathbb{R}^r$
$\phi^2 : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$
$\tilde{\phi}^2 : \mathbb{R}^r \to \mathbb{R}^p$
$(f^1_{\text{filter}})^{-1}(\tilde{\mathbf{U}}_h) = \mathbf{U}_h$
$f^2_{\text{filter}}(\mathbf{U}_h) = \tilde{\mathbf{U}}_h$
$(f^2_{\text{filter}})^{-1}(\tilde{\mathbf{U}}_h) = \mathbf{U}_h$
$\phi^2 : \mathbb{R}^r \to P^s_{r_h}(X_h) \sim \mathbb{R}^s$
$\tilde{\phi}^1 : \mathbb{R}^r \to \mathbb{R}^p$
$\tilde{\psi}^2 : \mathbb{R}^p \to \mathbb{R}^r$
$\tilde{\phi}^2 : \mathbb{R}^r \to \mathbb{R}^p$
$(f^2_{\text{filter}} \circ (f^2_{\text{filter}})^{-1})(\mathbf{U}^1_h) = \mathbf{U}^2_h$
$(f^2_{\text{filter}} \circ P^s_{r_h})^{-1}(\tilde{\mathbf{U}}_h) = P^s_{r_h}(\mathbf{U}_h)$

Fig. 7.5 **Above:** the decoder ϕ^1 of subdomain 1 on the left, and the encoder ψ^2 followed by the decoder ϕ^2 of subdomain 2. To pass from one subdomain whose solution manifold is spanned by the rSVD modes associated to $(f^1)^{-1}_{\text{filter}}$ to the one spanned by the rSVD modes associated to f^2_{filter} and $(f^2)^{-1}_{\text{filter}}$, an efficient change of basis needs to be performed during the online stage. The dependency on the number of dofs is avoided by multiplying in the offline stage the rSVD basis associated with subdomain 1 with the ones associated with subdomain 2. **Below:** the same maps as above, only now an efficient change of basis between subdomains $\tilde{\mathbf{U}}^2_h = f^{12}_{\text{filter}}(\tilde{\mathbf{U}}^1_h) = (f^2_{\text{filter}} \circ (f^1_{\text{filter}})^{-1})(\tilde{\mathbf{U}}^1_h)$ can be used online, since it is precomputed offline as a change of basis matrix of sizes $p^2 \times p^1$.

7.5 Numerical experiments

We will test the presented methodology on two challenging benchmarks from the point of view of model order reduction, as they evidently suffer from a slow Kolmogorov n-width decay. Employing classical linear projection-based ROMs would be unfeasible as they would need more than hundreds of rSVD modes. For the comparison of classical methods with intrusive ones exploiting nonlinear approximants see [146, 19].

The test cases we will present are developed with the open-source CFD software library `OpenFoam` [264]. Being rather small test cases, discretized with the finite volumes method on meshes with **4500**, **32160** and **198633** cells, the speedups obtained are relatively small. However, since our ROMs are independent of the number of dofs, as long as increasing the resolution does not bring a slower Kolmogorov n-width decay, more evident speedups can be achieved. Compared to the finite differences method employed in [146, 19], the FVM implementation in `OpenFoam` is highly optimized and it is therefore more challenging to achieve a speedup for small test cases. The new methodology is implemented on top of the open-source software library for model order reduction `ITHACA-FV` [238].

The first test case we consider involves the compressible Navier-Stokes equations (CNS) in the supersonic regime. We will consider as parameters time and the Mach number $\text{Ma} \in [2,5]$ imposed at the inflow boundary. In order to show, that the method scales well increasing the number of dofs, we employ two meshes: a coarse one with **4500** cells and a finer one with **32160** cells, see Figure 7.6. On the first mesh in subsection 7, we test different hyper-reduction methods. On the second mesh in subsection 7 we test the use of local nonlinear manifold approximants and the straight-forward change of basis we have presented in section 7.4. The `OpenFoam` solver we employ is the sonicFoam [177] solver.

The second test case we consider is the incompressible turbulent flow around the Ahmed body (INS). The parameters are time and the slant angle of Ahmed's body. The test case presented is introduced and studied in [282]. Steady-state solutions are obtained with the solver SIMPLE [138](Semi-Implicit Method for Pressure-Linked Equations), however, we employ PISO [197] (Pressure Implicit with Splitting of Operator) since we want to reduce the transient dynamics that has a slow KnW. Adding time as parameter brings a much more complex solution manifold to approximate.

The specifics of the convolutional autoencoders used are reported in section 7.5.3 along with the training costs and other hyperparameters. For all the CAE trainings we employed the ADAM [147] optimizer with initial learning rate of 0.001 and a scheduler that halves it every 200 epochs if the validation loss has not improved. Every architecture is trained for 3000 epochs on a single GPU NVIDIA Quadro RTX 4000. The wallclock time expended on training is between 1 hour and 1 hour and half for the test cases in 7, 7 and 4. Substantial computational savings depend on the choice of the architecture, especially on the fact that the training is independent with respect to the number of dofs since the inputs and outputs of the autoencoder belong to the lower dimensional space generated by the rSVD basis.

In order not to interrupt the presentation of the numerical results, we postponed to section 7.5.4 the showcase of some predicted solutions of each test case: Figure 7.23 refers to the **CNS1** test case in line 7, Figure 7.24 refers to the **CNS2** test case in line 7 and finally Figures 7.25 and 7.27 refers to the test case **INS1** in line 4.

7.5.1 Supersonic flow past a NACA airfoil (CNS)

The first nonlinear time-dependent parametric PDE model we consider are the compressible Navier-Stokes equations for a perfect diatomic gas with ratio of specific heat $\gamma = 7/5$. A speed of sound of $c = \sqrt{\gamma \frac{R}{M} T} = 1\text{m/s}$ is imposed through the choice of molar mass M at a temperature of $T = 1K$, where R is the universal gas constant. The following system of PDEs is solved on the 2d computational domain $\Omega_h \subset \mathbb{R}^2$ shown in Figure 7.6:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \qquad \text{(mass conservation)} \tag{7.42a}$$

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p - \nabla \cdot \left(\nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T + \tfrac{1}{3}(\nabla \cdot \mathbf{u}) I_d\right)\right) = 0, \qquad \text{(momentum conservation)} \tag{7.42b}$$

$$\partial_t \rho(e+K) + \nabla \cdot ((\rho(e+K)+p)\mathbf{u}) - \nabla \cdot \left(\nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T + \tfrac{1}{3}(\nabla \cdot \mathbf{u}) I_d\right) \mathbf{u}\right) = 0, \qquad \text{(energy conservation)} \tag{7.42c}$$

$$e = Tc_v, \quad pc_v = \rho Re, \quad K = \tfrac{1}{2}\rho \mathbf{u}^2, \qquad \text{(state equations),} \tag{7.42d}$$

the parameters we consider are time and the inlet Mach number $\text{Ma} = |\mathbf{u}|/c = |\mathbf{u}|$, $\mu = \text{Ma} \in [2,5]$, $t \in V_\mu = \{1, \dots, N_\mu\}$. The viscosity is fixed at $\nu = 1e-5$. The boundary conditions are imposed at the inflow Γ_{inflow}, outflow Γ_{outflow} and airfoil Γ_{airfoil} boundaries, see Figure 7.6. The initial and boundary conditions for the velocity $\mathbf{u}$, pressure p and temperature T fields are:

$$\begin{cases} \mathbf{u}(\mathbf{x},t) = \text{Ma}, & (\mathbf{x},t) \in (\mathring{\Omega}_h \times \{t=0\}) \cup (\Gamma_{\text{inflow}} \times [0, T_{\boldsymbol{\mu}}]) \\ p(\mathbf{x},t) = 1, & (\mathbf{x},t) \in (\mathring{\Omega}_h \times \{t=0\}) \cup (\Gamma_{\text{inflow}} \times [0, T_{\boldsymbol{\mu}}]) \\ T(\mathbf{x},t) = 1, & (\mathbf{x},t) \in (\mathring{\Omega}_h \times \{t=0\}) \cup (\Gamma_{\text{inflow}} \times [0, T_{\boldsymbol{\mu}}]) \end{cases},$$

$$\begin{cases} \mathbf{n} \cdot \nabla \mathbf{u}(\mathbf{x},t) = 0, & \mathbf{x} \in \Gamma_{\text{outflow}} \\ \text{(non-reflective)}, & \mathbf{x} \in \Gamma_{\text{outflow}} \\ \mathbf{n} \cdot \nabla T(\mathbf{x},t) = 0, & \mathbf{x} \in \Gamma_{\text{outflow}} \end{cases}, \qquad \begin{cases} \mathbf{u}(\mathbf{x},t) = 0, & \mathbf{x} \in \Gamma_{\text{airfoil}} \\ \mathbf{n} \cdot \nabla p(\mathbf{x},t) = 0, & \mathbf{x} \in \Gamma_{\text{airfoil}} \\ \mathbf{n} \cdot \nabla T(\mathbf{x},t) = 0, & \mathbf{x} \in \Gamma_{\text{airfoil}} \end{cases},$$

where non-reflective boundary conditions are imposed on the pressure field at the outflow boundaries.

The Mach number training and test instances are sampled from the parameter space $\mathcal{P} = [2,5] \times V_\mu$, such that the Mach angle α is sampled uniformly, and the time step Δt and final time T_μ are chosen depending on the Mach number from the reference time step $(\Delta t)_{\text{ref}} = 0.001$ and final time $(T_\mu)_{\text{ref}} = 2.5$:

$$\alpha = \arcsin \frac{1}{\text{Ma}}, \quad \Delta t = (\Delta t)_{\text{ref}} \frac{\text{Ma}_{\text{ref}}}{\text{Ma}}, \quad T_\mu = (T_\mu)_{\text{ref}} \frac{\text{Ma}_{\text{ref}}}{\text{Ma}} \tag{7.43}$$

in this way, the training and test time series have the same length even if the final times are different.

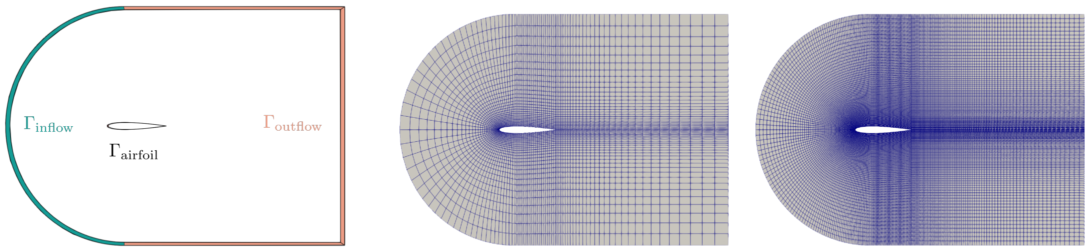


Fig. 7.6 **Left:** computational domain of the compressible flow past a NACA airfoil test case. **Center:** coarse mesh with 4500 cells. **Right:** finer mesh with 32160 cells.

We take into account two different meshes a coarse one with **4500** cells and **2700** dofs and a finer one with **32160** cells and **192960** dofs. We will refer to these two test cases with the notation CNS-1 for the coarse mesh and CNS-2 for the finer mesh. The employment of two meshes permits us to show that our methodologies achieve more significant speedups when the number of dofs is increased since they are independent of the number of dofs: the timings and relative speedups can be observed from Tables 7.1 and 7.2.

The solver we will employ is `OpenFoam`'s [264] pressure-based transonic/supersonic solver for compressible gases sonicFoam [177]. SonicFoam algorithm 8 solves for the solution at the n-th time instant follows PIMPLE predictor-corrector scheme, a combination of PISO [197] and SIMPLE [138]. In algorithm 8, we highlight the predictor-corrector scheme and the main steps. We employ the Euler scheme in time. Starting from the previous fields $\mathbf{u}^n$ velocity, ρ^n density, e^n internal energy, and p^n pressure, the solutions at time step $n+1$ are evaluated. After an intermediate density evaluation ρ^* in line 2 corresponding to the continuity equation (7.42a), begins the PIMPLE corrector loop from line 3: this outer loop comes from SIMPLE and relaxes the intermediate solution fields after every iteration. Then, at line 4, the intermediate velocity field $\mathbf{u}^*$ is evaluated implicitly solving the momentum equation (7.42b): the diagonal $A[\rho^*, \Delta t]$ and over-diagonal $H[\mathbf{u}^*, \rho^*, \Delta t]$ parts of the system are highlighted along with the pressure contribution ∇p^* since they will be employed later in the pressure-Poisson equation at line 10. The energy predictor step at line 5 is evaluated afterwards corresponding to the energy conservation (7.42c). Then, the thermodynamics properties corresponding to the state equations (7.42d) are corrected and the PISO pressure corrector loop begins at line 7. Inside the non-orthogonal corrector loop in case of non-orthogonal meshes, the pressure-Poisson equation is repeatedly solved and, afterward, the velocity is corrected to satisfy the continuity equation and the density is updated with the new pressure through the equations of state. Many steps have not been reported for simplicity, for a more detailed analysis see [177]. What we want to show is that in comparison the nonlinear least-squares Petrov-Galerkin scheme for the compressible Navier-Stokes (CNS) equations (NM-LSPG-CNS) is simplified, at each n-th time step and i-th intermediate optimization step in algorithm 9: having a solution manifold, trained on a previous database, as ansatz space, the corrector loops can be avoided and only the residual evaluation of the mass (7.42a), momentum (7.42b), energy (7.42c) and pressure-Poisson equations are needed.

For the same reasoning, the orthogonal corrector loops can be avoided as the solutions searched for on the approximant nonlinear manifold should be already corrected.

Along the lines of the previous considerations, we can employ larger time steps. In fact, it will be shown that using four times the reference time step, that is $(\Delta t)_{\text{ref}} = 0.004$, brings a speedup also in the case of a coarse mesh, see Table 7.1.

To get a grasp of the extension of the solution manifold for the test cases CNS-1 and CNS-2, we show 4 snapshots corresponding to the time instants and Mach numbers $t = 0.2s$, $\text{Ma} = 5.2$, $t = 2.5s$, $\text{Ma} = 5.2$, $t = 0.2s$, $\text{Ma} = 1.8$, and $t = 2.5s$, $\text{Ma} = 1.8$ in Figure 7.8.

The value of the L^2 relative error is low for the internal energy field because its absolute value is higher than the absolute errors as can be seen in section 7.5.4 for the test cases CNS1 and CNS2.

Interpolation and quadrature based hyper-reduction on a coarse mesh (CNS-1)

As anticipated, in this subsection we will consider a coarse mesh of **4500** cells, reported in Figure 7.6, for a total of $d = 27000$ dofs. The training interval is $\text{Ma} \in [2,5]$ and $t \in \{0, \Delta t, \dots, 2500 \cdot \Delta t\} = V_{\boldsymbol{\mu}}$ with $\text{Ma}_{\text{ref}} = 2$ and $(T_\mu)_{\text{ref}} = 2$, and the time steps opportunely scaled with respect the current Mach number through equation (7.43). We consider the following training and test Mach numbers

$$\mathcal{P}_{\text{train}} = \{5, 4.256, 3.709, 3.291, 2.960, 2.693, 2.473, 2.290, 2.134, 2\} \times V_{\boldsymbol{\mu}}, \quad |\mathcal{P}_{\text{train}}| = 10 \cdot 2500, \tag{7.44a}$$

$$\mathcal{P}_{\text{test}} = \{\mathbf{5.2}, 4.042, 3.314, 2.816, 2.455, 2.182, \mathbf{1.970}, \mathbf{1.8}\} \times V_{\boldsymbol{\mu}}, \quad |\mathcal{P}_{\text{test}}| = 8 \cdot 2500, \tag{7.44b}$$

it can be observed that the first and last two test parameters are in the extrapolation regime as they don't belong to the interval $[2,5]$. From now on, the test parameters will be numbered in the order in which they appear in equation (7.44b) and refer only to the Mach number: from test parameter 1 with $\text{Ma} = 5.2$ to test parameter 8 with $\text{Ma} = 1.8$, we will use this notation also in the following figures. A grasp of the solution manifold extension can be observed in Figure 7.8. So, the training dataset $A_{\text{train}} \in \mathbb{R}^{d \times n_{\text{train}}}$ is represented by $n_{\text{train}} = |\mathcal{P}_{\text{train}}| \cdot 1250 = 12500$ training snapshots, as only one every two time instants is saved. The test dataset $A_{\text{test}} \in \mathbb{R}^{d \times n_{\text{test}}}$ is composed by $n_{\text{test}} = |\mathcal{P}_{\text{test}}| \cdot 500 = 5000$ snapshots as only one every four time instants is saved. The number of rSVD modes considered is $r_{\text{rSVD}} = 150$ evaluated from the training dataset A_{train}. The CAE architecture employed is reported in Table 7.4.

As first study, we show in Figure 7.9 the accuracy of the different hyper-reduction methods introduced in section 7.3. For all the methods, it is employed a fixed number of $r_h = \mathbf{500}$ collocation nodes and $r_{\text{rSVD}} = \mathbf{150}$ rSVD modes used for both the definition of the nonlinear approximant map introduced in section 7.2.2 and the hyper-reduction basis. When residual bases **RB** are employed they

Fig. 7.7 **INS**. Comparison between the n-th time instant iterations of the full-order model numerical scheme sonicFoam (**Left**) and the nonlinear manifold least-squares Petrov Galerkin (NM-LSPG-CNS) method (**Right**).

Algorithm 8: sonicFoam n-th iteration

```
1  Start with an initial velocity field $\mathbf{u}^n$, density field $\rho^n$, internal energy field $e^n$, and pressure field $p^n$ at the $n$-th time step.
2  Density prediction step:
       $\rho^* - \rho^n + \Delta t \nabla \cdot (\rho^* \mathbf{u}^n) = 0$
3  while Pressure-velocity PIMPLE corrector loop do
4      Momentum predictor step:
           $A[\rho^*]\mathbf{u}^* = H[\mathbf{u}^*, \rho^*] - \nabla p^*.$
5      Energy predictor step:
6          $\partial_t(\rho^*(e^* + K^*)) + \nabla \cdot (\rho^* \mathbf{u}(e^* + K^*) + \mathbf{u}^* p^*) = 0$
7      Correct thermodynamics properties.
8      while Pressure corrector loop do
9          while Non-orthogonal corrector loop do
10             Evaluate the pressure-corrector $p^*$:
11                 $\nabla \cdot (A[\rho^*]^{-1} \nabla p^*) = \nabla \cdot (A[\rho^*]^{-1} H[\mathbf{u}^{n,i}, \nu_t^{n,i}])$
12         Correct density and velocity
               $\mathbf{u}^* \leftarrow \mathbf{u}^* - A^{-1} \nabla p^*$
```

Algorithm 9: NM-LSPG-CNS (i,n)-th step

```
1  Start with an initial velocity field $\mathbf{u}^n$, density field $\rho^n$, internal energy field $e^n$, and pressure field $p^n$ at the $n$-th time step.
2  Density residual evaluation:
       $r_\rho = \rho^* - \rho^n + \Delta t \nabla \cdot (\rho^* \mathbf{u}^n)$
   Momentum residual evaluation:
       $r_\mathbf{u} = A[\rho^*]\mathbf{u}^{n,i} - H[\mathbf{u}^{n,i}, \nu_t^{n,i}] + \nabla p^{n,i}.$
3  Energy residual evaluation:
4      $r_e = \partial_t(\rho^*(e^* + K^*)) + \nabla \cdot (\rho^* \mathbf{u}(e^* + K^*) + \mathbf{u}^* p^*)$
5  Pressure-Poisson residual evaluation:
       $r_p = \nabla \cdot (A[\rho^*]^{-1} \nabla p^{n,i}) - \nabla \cdot (A[\rho^*]^{-1} H[\mathbf{u}^{n,i}, \nu_t^{n,i}]).$
6  Normalization of the residuals:
       $r_\rho \leftarrow \frac{r_\rho}{\max_i \rho_i}, \quad r_\mathbf{u} \leftarrow \frac{r_\mathbf{u}}{\max_i \mathbf{u}_i}, \quad r_e \leftarrow \frac{r_e}{\max_i e_i},$
7      $r_p \leftarrow \frac{r_p}{\max_i p_i}$
```

are evaluated separately with respect to the ones used to define the nonlinear approximant $\phi : \mathbb{R}^r \to \mathbb{R}^d$. It can be seen that the reconstruction error (7.8) of the autoencoder represents the baseline accuracy that we want to reach in terms of mean L^2 relative error. It is also clear that with our implementation of hyper-reduction the most accurate but also performing methods are the collocated ones. The lower

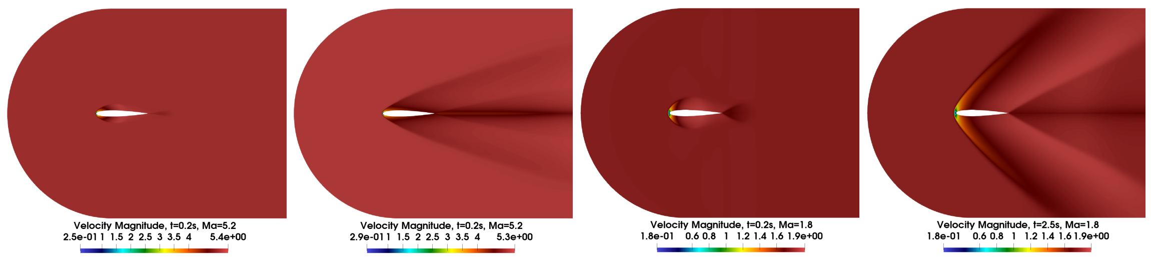


Fig. 7.8 **CNS-1 and CNS-2**. **From left to right**: test full-order snapshots representing the velocity field magnitude corresponding to the parametric values $(t = 0.2s,\ \mathrm{Ma} = 5.2)$, $(t = 2.5s,\ \mathrm{Ma} = 5.2)$, $(t = 0.2s,\ \mathrm{Ma} = 1.8)$, and $(t = 2.5s,\ \mathrm{Ma} = 1.8)$. The whole extension of the solution manifold includes the transient dynamics from the time instants $t = 0s$ to $t = 2.5s$ opportunely rescaled through equation (7.43) and the different test Mach number values $\mathrm{Ma} \in [1.8, 5.2]$. The presence of moving discontinuities at different Mach angles makes this test case difficult to reduce with classical linear projection based ROMs.

accuracy of the other methodologies may be attributed to our choice of considering the physical fields of interest altogether in a monolithic fashion, both for the evaluation of the rSVD bases employed in the HR offline stage and the computation of the normalization vectors from equation (7.14). See section 7.6 for further comments. Moreover, collocation methods are more efficient as the collocated residuals do not need to be multiplied further by a pseudo-inverse or a vector of weights as in DEIM and ECSW methods. Further comments are delivered in section 7.6.

Since we observed that collocated hyper-reduction reached a better accuracy, we show in the next study the decay of the mean L^2 relative error associated to the C-DEIM, C-DEIM-SOPT, and C-UP50 methods, varying the number of collocation nodes/magic points from 150 to 1200. The results are shown in Figure 7.10. The most performing method is C-UP50, the gradient-based adaptive one, which pays the additional cost of a submesh update every 50 time-steps.

The advantage of having a continuous nonlinear approximant for the solution manifold enables the possibility to choose a bigger reference time step in the online prediction stage. The results in terms of mean and max L^2 relative error are shown in Figure 7.11. Thanks to this choice of reference time step $(\Delta t)_{\mathrm{ref}} = 0.004$ four times bigger than the full-order one $(\Delta t)_{\mathrm{ref}} = 0.001$ and twice as the sampling step used to select the training snapshots $A_{\mathrm{train}} \in \mathbb{R}^{d \times n_{\mathrm{train}}}$, a speedup can be achieved also for this small test case.

The average computational times of the NM-LSPG method with gradient-based adaptive hyper-reduction C-UP50 are shown in Table 7.1. The average is performed considering all 8 test parameters for different Mach numbers. The average total time includes the cost of submesh updates introduced by the C-UP50 hyper-reduction. There is no evident speedup with respect to the full-order model. However, when a four times bigger reference time step $(\Delta t)_{\mathrm{ref}} = 0.004$ is imposed a speedup of almost 2 is achieved also for this small test case.

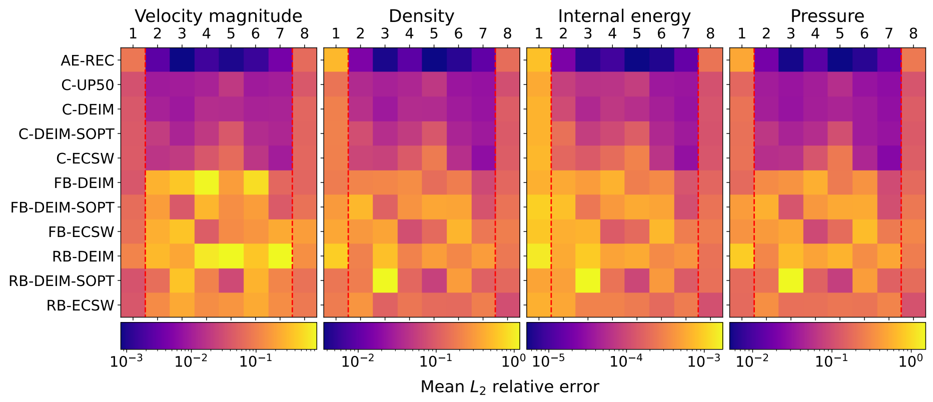


Fig. 7.9 **CNS-1**. Comparison of hyper-reduction methods based on the mean L^2 relative error evaluated on each physical field of interest for the CNS test case. The acronyms correspond to: **AE-REC** is the autoencoder reconstruction error from (7.8), **C-UP50** is the reduced collocation method with the gradient based-sampling strategy from subsection 7.3.2 applied every 50 time steps over 2000 time instants; the other notations are introduced in subsections 7.3.1, 7.3.1, and 7.3.1. For all the methods, it is employed a fixed number of $r_h = \mathbf{500}$ collocation nodes and $r_{\text{rSVD}} = \mathbf{150}$ rSVD modes used for both the definition of the nonlinear approximant map introduced in section 7.2.2 and the hyper-reduction basis. With the current monolithic hyper-reduction implementation, the best methods are the collocated ones, for further comments see section 7.6 on this matter.

Table 7.1 **CNS-1**. Timings of the CNS on a coarse mesh, subsection 7. The timings of the reduced over-collocation method **C-UP50** relative to the choice of $r_h = \{150, 300, 600, 1200\}$ collocation nodes is reported along the full-order model (FOM) timings of the sonicFoam solver. The same reference time step as the FOM one is employed $(\Delta t)_{\text{ref}} = 0.001$. Before the FOM results, the timings of **C-UP50** but with a larger reference time step of $(\Delta t)_{\text{ref}} = 0.004$ are reported in bold for a number of magic points $r_h = 500$. For the timings of the CNS on the finer mesh, see Table 7.2.

collocation nodes (r_h)	mean time-step	mean update every 50	average total time
$r_h = 150$, $(\Delta t)_{\text{ref}} = 0.001$	11.937 [ms]	56.830 [ms]	29.842 [s]
$r_h = 300$, $(\Delta t)_{\text{ref}} = 0.001$	25.294 [ms]	97.628 [ms]	63.235 [s]
$r_h = 600$, $(\Delta t)_{\text{ref}} = 0.001$	36.894 [ms]	110.110 [ms]	92.234 [s]
$r_h = 1200$, $(\Delta t)_{\text{ref}} = 0.001$	56.277 [ms]	119.707 [ms]	140.691 [s]
$r_h = \mathbf{500}$, $(\Delta t)_{\text{ref}} = 0.004$	**28.736** [ms]	**82.337** [ms]	**17.960** [s]
FOM, $(\Delta t)_{\text{ref}} = 0.001$	13.440 [ms]	-	33.614 [s]

Collocated hyper-reduction on a finer mesh (CNS-2)

To be sure to obtain a good approximation with rSVD modes, we increase the number of training snapshots for this CNS-2 test case on the finer mesh of **32160** cells shown in Figure 7.6. So we

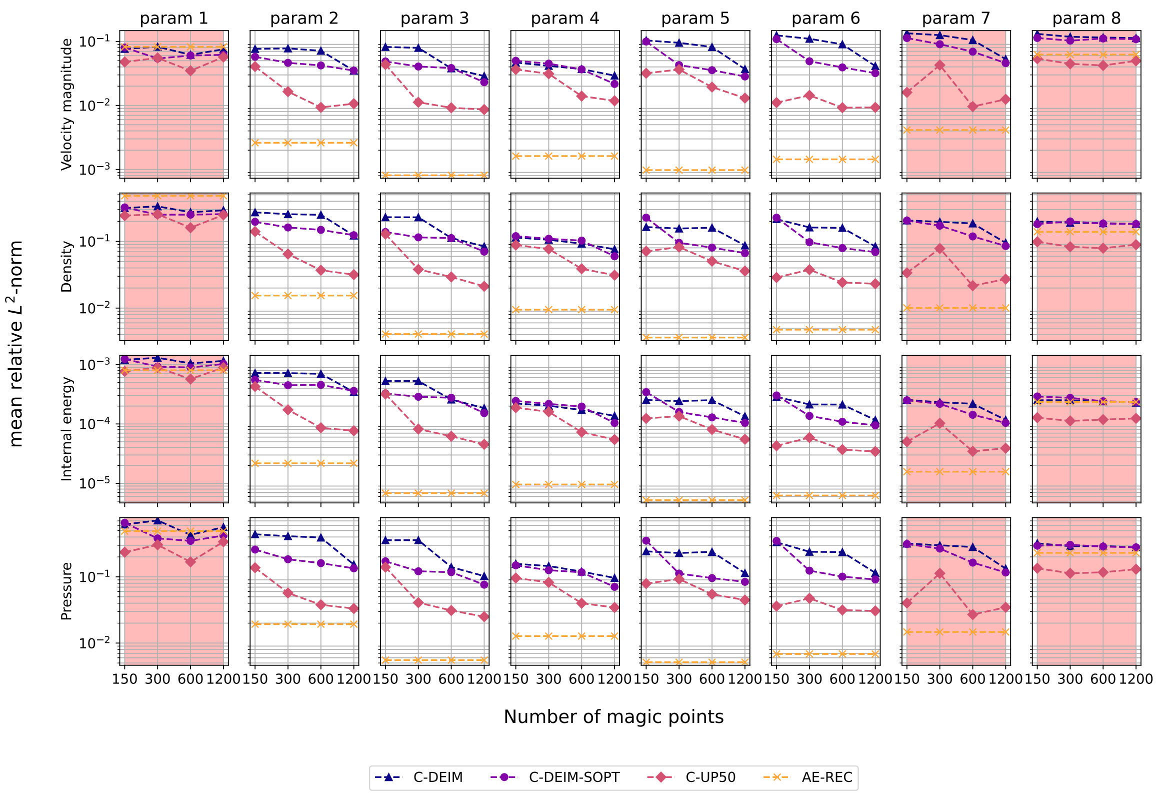


Fig. 7.10 **CNS-1**. The decay of the mean L^2 relative error is assessed for the **C-DEIM**, **C-DEIM-SOPT**, and **C-UP50** reduced over-collocation methods with sampling strategies corresponding to a greedy one, the quasi-optimal (SOPT) one and the gradient-based adaptive one every 50 time-steps. In this case, we employ the same reference time step as the full-order model $(\Delta t)_{\text{ref}} = 0.001$. The baseline represented by the autoencoder mean L^2 reconstruction error **AE-REC** is reported. The extrapolation parameters have a shaded red background.

consider the following 20 training and 5 test parameters:

$$\begin{aligned}\mathcal{P}_{\text{train}} = \{&5., 4.617, 4.290, 4.007, 3.760, 3.543, 3.350, 3.178, 3.024, 2.885,\\ &2.758, 2.644, 2.539, 2.442, 2.354, 2.272, 2.196, 2.126, 2.061, 2.\} \times V_{\boldsymbol{\mu}}, \quad |\mathcal{P}_{\text{train}}| = 20 \cdot 2500,\end{aligned} \tag{7.45a}$$

$$\mathcal{P}_{\text{test}} = \{\mathbf{5.2}, 3.469, 2.622, 2.124, \mathbf{1.8}\} \times V_{\boldsymbol{\mu}}, \quad |\mathcal{P}_{\text{test}}| = 5 \cdot 2500, \tag{7.45b}$$

the first and last test parameters in bold correspond to the extrapolation regime. We remark that we did not optimize the number of training snapshots: possibly, a smaller number of them is needed to obtain an accurate regression of the solution manifold. As for the previous test case we will refer to the test Mach numbers from Ma $= 5.2$ to Ma $= 1.8$ with the numbers from 1 to 5 in this order. A grasp of the extension of the solution manifold we want to approximate is shown in Figure 7.8. The training dataset $A_{\text{train}} \in \mathbb{R}^{d \times n_{\text{train}}}$ is represented by $n_{\text{train}} = |\mathcal{P}_{\text{train}}| \cdot 625 = 20 \cdot 625 = 12500$ training snapshots,

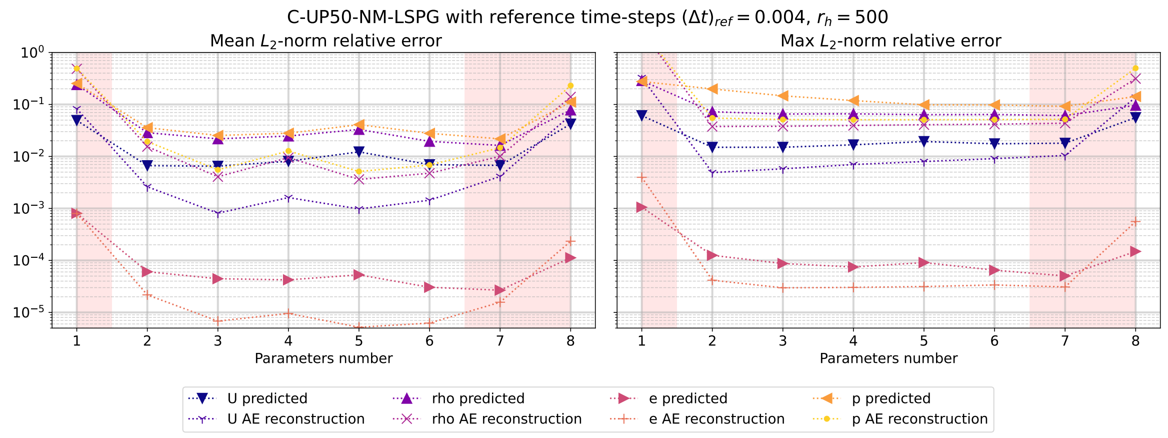


Fig. 7.11 **CNS-1**. Accuracy of the nonlinear manifold least-squares Petrov-Galerkin (NM-LSPG) method with adaptive gradient-based reduced over-collocation (C-UP50) every 50 time steps. The results are reported in terms of the mean L^2 relative error for the physical fields of interest with respect to the 8 test parameters. The extrapolation regimes are delimited by a red-shaded background. The number of magic points is $h_r = 500$. It is important to notice that the reference time step is $(\Delta t)_{\text{ref}} = 0.004$ and not $(\Delta t)_{\text{ref}} = 0.001$ equal to the full-order model one. This permits the methodology to reach a speedup even for these small test cases with only **4500** cells.

since only one every four time instants is saved. The test dataset $A_{\text{test}} \in \mathbb{R}^{d\times n_{\text{test}}}$ is composed by $n_{\text{test}} = |\mathcal{P}_{\text{test}}| \cdot 625 = 5 \cdot 625 = 3125$ snapshots as only one every four time instants is saved. At first, the number of rSVD modes considered on the whole parameter space is $r_{\text{rSVD}} = 300$ evaluated from the training dataset A_{train}. Later, we will extract rSVD on two separate time intervals to study the performance of local nonlinear solution manifolds. The decay of the mean and max L^2 reconstruction errors is shown for the fields of interest in Figures 7.12 and 7.13 for the training and test datasets. The rSVD basis evaluated from the training dataset as explained in section 7.2.3 are the same used to evaluate the test reconstruction error in Figure 7.13. The presence of moving discontinuities coming from the transient dynamics and the different Mach angles causes an evident degradation of the test reconstruction error with respect to the training reconstruction error.

For this test case, a moderately high number of rSVD modes equal to $r_{\text{rSVD}} = 300$ can be employed to approximate with sufficient accuracy the solution manifold. However, it can be observed that more complex parameter dependencies can exacerbate this behavior and make the approximation of the solution manifold with a linear rSVD basis unfeasible. In those cases, fully nonlinear NN architectures directly supported on the dofs of the physical fields of interest can be employed and hyper-reduced with the variant of the nonlinear manifold LSPG method introduced in [223].

We want to study the implementation of local nonlinear manifolds approximants and how to efficiently change from one subdomain to the other in the online stage. We consider only two parametric subdomains determined by the splitting of the reference time interval $[0s, 2.5s]$ into to two subintervals $[0s, 1.2s] \supset V^1_{\boldsymbol{\mu}}$ and $[0.6s, 2.5s] \supset V^2_{\boldsymbol{\mu}}$, with $|V^1_{\boldsymbol{\mu}}| = 1.2/(\Delta t)_{\text{ref}} = 300$ and $|V^2_{\boldsymbol{\mu}}| =$

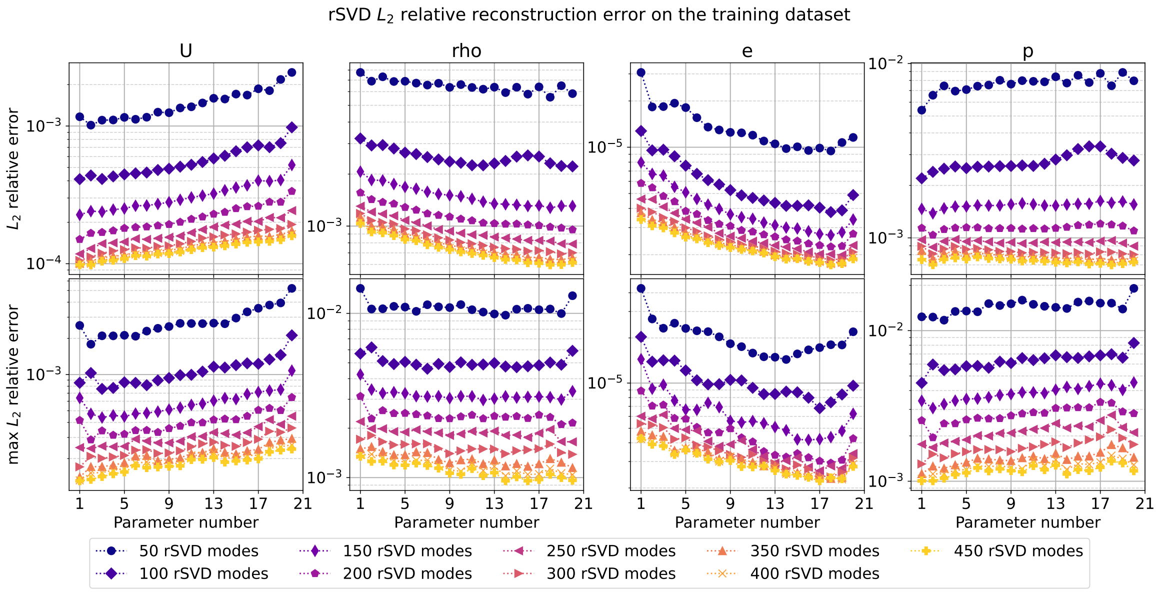


Fig. 7.12 **CNS-2**. The decay of the mean and max L^2 reconstruction errors defined through equations (7.17a) for the 20 train parameters introduced in equation (7.45a), is shown. The number of rSVD modes initially chosen on the whole parameter space for this test case is $r_{\text{rSVD}} = \mathbf{300}$. Even though the solution manifold includes moving discontinuities at different Mach angles as shown in Figure 7.8, a moderately high number of rSVD modes can still be employed to obtain a good reconstruction error but at the same time precludes the implementation of efficient linear projection-based ROMs. A heuristic understanding of the moderately slow Kolmogorov n-width decay for this test case can be observed in the degradation of the reconstruction error on the test set 7.13.

$(2.5-0.6)/(\Delta t)_{\text{ref}} = 625-150$ since one every 4 time instants in $[0,2.5]$ is saved with a reference time step of $(\Delta t)_{\text{ref}} = 0.001$. Notice that they overlap in order to achieve a good accuracy when the change of basis is performed. The change of basis between the domains is performed at the reference time instant $t = \mathbf{0.8s}$. More sophisticated techniques can be implemented [284]. The notation we employ to distinguish between the subdomains is introduced in section 7.4. So the parameter spaces we consider are:

$$\begin{aligned}\mathcal{P}^1_{n_{\text{train},1}} = \{&5., 4.617, 4.290, 4.007, 3.760, 3.543, 3.350, 3.178, 3.024, 2.885,\\ &2.758, 2.644, 2.539, 2.442, 2.354, 2.272, 2.196, 2.126, 2.061, 2.\} \times V^1_{\boldsymbol{\mu}}, \quad |\mathcal{P}^1_{n_{\text{train},1}}| = 20\cdot 300,\end{aligned} \tag{7.46a}$$

$$\begin{aligned}\mathcal{P}^2_{n_{\text{train},2}} = \{&5., 4.617, 4.290, 4.007, 3.760, 3.543, 3.350, 3.178, 3.024, 2.885,\\ &2.758, 2.644, 2.539, 2.442, 2.354, 2.272, 2.196, 2.126, 2.061, 2.\} \times V^2_{\boldsymbol{\mu}}, \quad |\mathcal{P}^2_{n_{\text{train},2}}| = 20\cdot (625-150),\end{aligned} \tag{7.46b}$$

and consequently the training snapshots matrices $A^i_{n_{\text{train},i}} \in \mathbb{R}^{d\times n_{\text{train},i}}$ $i=1,2$ are assembled and the two rSVD basis $U^i \in \mathbb{R}^{d\times r_{\text{rSVD}}}$ $i=1,2$ evaluated. Each one has $r_{\text{rSVD}} = 300$ modes, but in principle, a

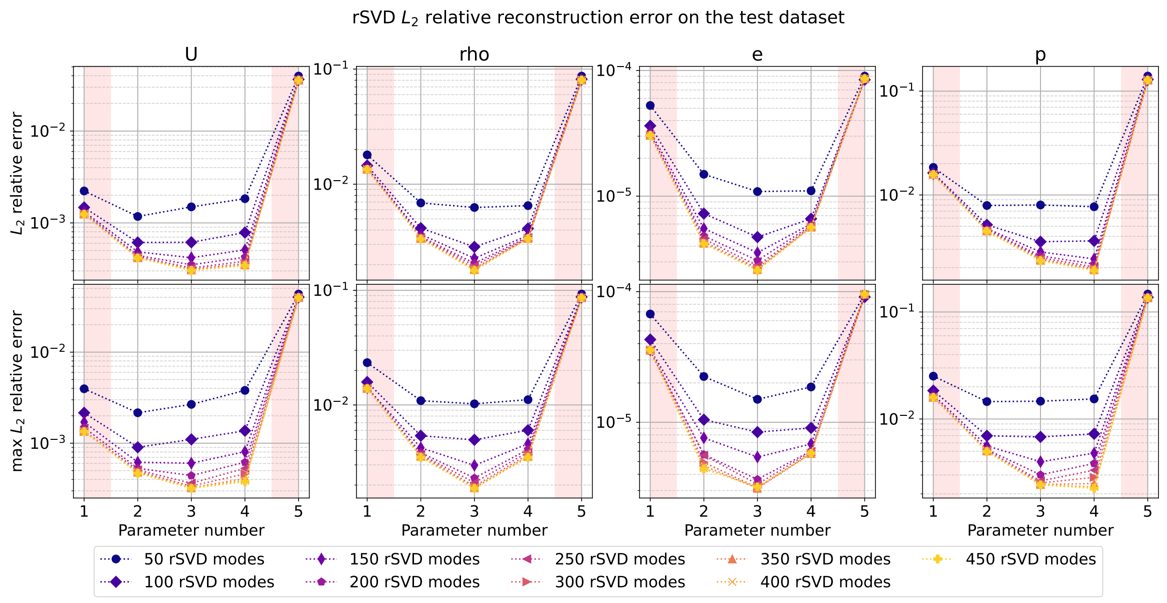


Fig. 7.13 **CNS-2**. The decay of the mean and max L^2 reconstruction errors defined through equations (7.17a) for the 5 test parameters introduced in equation (7.17b), is shown. The number of rSVD modes initially chosen on the whole parameter space for this test case is $r_{\text{rSVD}} = 300$. The presence of moving discontinuities at different Mach angles as shown in Figure 7.8 causes an evident degradation of the reconstruction error with respect to the training reconstruction error in Figure 7.12. A shaded red background identifies the extrapolation regime.

different number can be used. The number of training snapshots are $n_{\text{train},1} = 20 \cdot |V^1_{\boldsymbol{\mu}}| = 20 \cdot 300$ and $n_{\text{train},2} = 20 \cdot |V^2_{\boldsymbol{\mu}}| = 20 \cdot (625 - 150)$. The same CAE architecture for the two nonlinear approximants $\phi^i : \mathbb{R}^r \to X_h \sim \mathbb{R}^d$ $i = 1,2$ is employed and reported in Table 7.5. The latent dimension is 4 in both cases. We remind that the change of basis matrix from equation (7.41) is computed offline so the methodology maintains the independence with respect to the number of dofs.

The results for the choice of $r_h = 800$ collocation nodes, $(\Delta t)_{\text{ref}} = 0.001$ reference time step and $C-UP50$ hyper-reduction method, are reported in Figure 7.14. There, the whole trajectories corresponding to the 5 test parameters numbered in order are shown. In particular, the extrapolation regimes involving parameters 1 and 5 are shown, and the instant $t = 0.8s$ in reference time scale is highlighted for each test trajectory by an orange vertical line. It must be observed that not always the employment of local nonlinear submanifolds permits reaching smaller prediction errors: the only test parameter affected by an improvement is test parameter 4, which nonetheless corresponds to Mach number $\text{Ma} = 2.124$, and therefore it is more difficult to approximate since the Mach angle is wider. For this test case, we don't notice discontinuities from the passage of the solution from one local solution manifold to the other, as it can be seen also from the continuity of the errors in Figure 7.14.

Similarly to the previous test case CNS-1, the decay of the mean L^2 relative error is studied in Figure 7.15 for the C-UP20 and C-UP50 hyper-reduction methods with reference time step $(\Delta t)_{\text{ref}} = 0.004$, that is four times the reference time step of the FOM $(\Delta t)_{\text{ref}} = 0.001$. In this case, always two local subdomains are considered. Associated with these studies, the computational costs for mean time instant evaluation and mean total time expended for the whole trajectory evaluation with reference time step $(\Delta t)_{\text{ref}} = 0.004$, are shown in Table 7.2. It can be observed that the computational costs increase from the same number of collocation nodes (MP) in Table 7.2 with respect to Table 7.1: this is due mostly to the number of rSVD basis changed from 150 to 300.

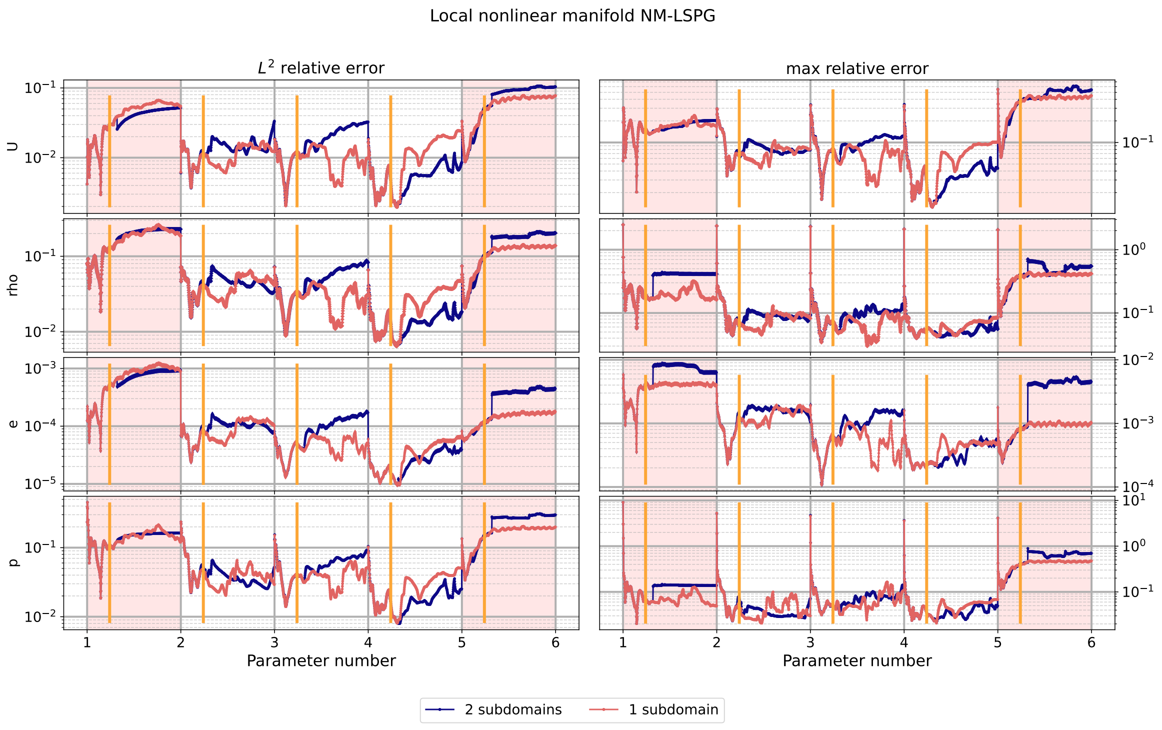


Fig. 7.14 **CNS-2**. Results corresponding to the employment of two local nonlinear manifolds corresponding to the reference time intervals $[0s, 1.2s] = V^1_{\boldsymbol{\mu}}$ and $[0.6s, 2.5s] = V^2_{\boldsymbol{\mu}}$, as introduced in section 7.4. The mean L^2 relative error and relative errors in max norm are shown for each of the **625** time instances associated with each one of the **5** test parameters corresponding to the test Mach numbers in equation (7.45b). The reference time instant in which the change of basis is performed is $t = 0.8s$ and it is highlighted by an orange vertical line for each of the 5 test parameters. The extrapolation parameters 1 and 5 have a shaded red background.

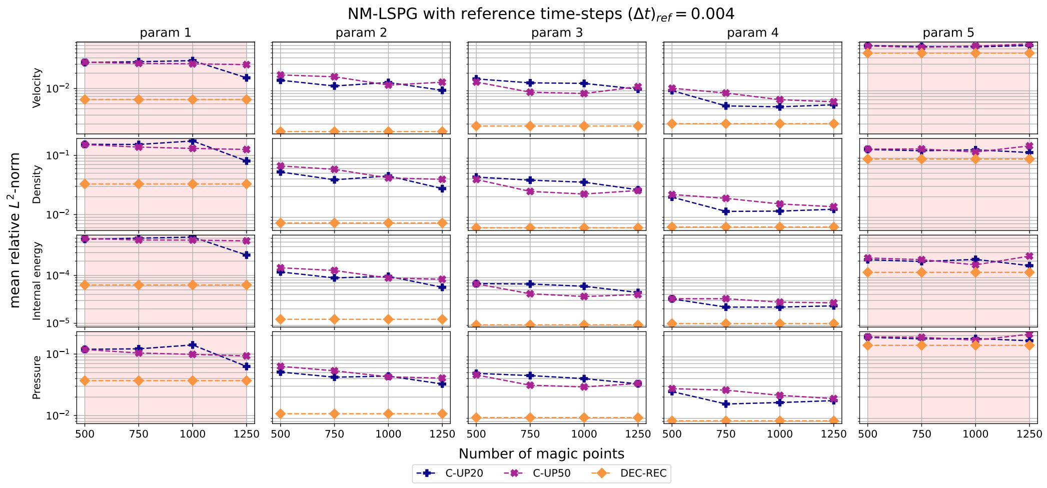


Fig. 7.15 **CNS-2**. The decay of the mean L^2 relative error is assessed for the **C-UP20**, and **C-UP50** reduced over-collocation methods with sampling strategies corresponding to the gradient-based adaptive one every 50 and 20 time-steps, respectively. In this case, we employ a reference time step of four times $(\Delta t)_{\text{ref}} = 0.004s$ the full-order model one $(\Delta t)_{\text{ref}} = 0.001s$. We always consider two local nonlinear manifolds. The baseline represented by the autoencoder mean L^2 reconstruction error **AE-REC** is reported. The extrapolation parameters 1 and 5 have a shaded red background.

Table 7.2 **CNS-2**. Timings of the CNS on a finer mesh, subsection 7. The timings of the reduced over-collocation method **C-UP50** relative to the choice of $r_h = \{150, 300, 600, 1200\}$ collocation nodes is reported along the full-order model (**FOM**) timings of the sonicFoam solver. The reference time step of **C-UP50** is $(\Delta t)_{\text{ref}} = 0.004$, four times bigger than the FOM. The different reference time steps affect the mean total time and not the mean time steps computational costs. For the timings of the CNS on the coarse mesh, see Table 7.1. The change of basis between the local nonlinear manifolds is irrelevant but nonetheless included in the mean total time. The label $m.$ stands for mean.

	m. time-step	m. update every 20	m. total time	m. time-step	m. update every 50	m. total time
150	38.615 [ms]	286.103 [ms]	34.215 [s]	35.297 [ms]	302.612 [ms]	25.890 [s]
300	52.209 [ms]	325.225 [ms]	42.774 [s]	48.940[ms]	316.992 [ms]	34.717 [s]
600	62.888 [ms]	330.269 [ms]	49.642 [s]	61.803 [ms]	358.956 [ms]	42.887 [s]
1200	74.759 [ms]	362.719 [ms]	58.302 [s]	75.128 [ms]	362.236 [ms]	51.510 [s]
FOM	110.334 [ms]	-	275.944 [s]	110.334 [ms]	-	275.944 [s]

7.5.2 Incompressible turbulent flow around the Ahmed body (INS)

The other benchmark (INS) we introduce to test our methodology involves the Reynolds-averaged Navier-Stokes equations (RANS) used to model an incompressible flow around the Ahmed body:

$$\partial_t \bar{\mathbf{u}} + \nabla\cdot(\bar{\mathbf{u}}\otimes\bar{\mathbf{u}}) + \nabla\bar{p} - \nabla\cdot\left((\nu+\nu_t)\left(\nabla\bar{\mathbf{u}}+\nabla\bar{\mathbf{u}}^T\right)\right) = 0, \qquad \text{(momentum conservation)} \qquad (7.47a)$$

$$\nabla\cdot\bar{\mathbf{u}} = 0, \qquad \text{(mass conservation)} \qquad (7.47b)$$

where $\bar{\mathbf{u}}$ and $\bar{p}$ are the time-averaged velocity and the kinematic pressure fields, and ν_t is the kinematic Eddy turbulent viscosity. The turbulence is modelled with the Spalart-Allmaras one-equation model [236, 269]. The Reynolds' number is in the order of $\mathrm{Re} = 2.8\cdot 10^6$. The parameters we consider are the slant angle θ of the Ahmed body and time $\mathcal{P} = [\theta_{\min},\theta_{\max}]\times V_{\boldsymbol{\mu}}$. We will consider two parameter ranges for the slant angle: one associated to small geometrical deformations in subsection 4 (INS-1) with $\theta_{\min} = 15^\circ$ and $\theta_{\max} = 18.265^\circ$ and one associated to large geometrical deformations in subsection 4 (INS-2) with $\theta_{\min} = 15^\circ$ and $\theta_{\max} = 35^\circ$.The computational domain is shown in Figure 7.16. The geometry of the Ahmed body and the mesh are deformed with radial basis function interpolation as described in [282].

The mesh, geometries, initial and boundary conditions are taken from the studies performed in [282] where a classical linear projection-based method is applied to reduce the SIMPLE [138] numerical scheme with the employment of a neural network to approximate the Eddy viscosity. In that case, steady-state solutions are predicted while we focus on the transient dynamics since it is more challenging from the point of view of solution manifold approximation. In fact, we employ the PISO [197] numerical scheme to achieve, from the initial conditions, convergence towards periodic cycles rather than stationary solutions.

The time interval is not dependent on the slant angle θ, that is the final time is $T_{\boldsymbol{\mu}} = T = 0.1s$, the time step is $(\Delta t)_{\boldsymbol{\mu}} = \Delta t = 0.0001s$ and the collection of time instants is $V_{\boldsymbol{\mu}} = V = \{0s, 0.0001s,\dots,0.1s\}$. The initial and boundary conditions for the velocity $\bar{\mathbf{u}}$ and pressure $\bar{p}$ averaged fields are:

$$\begin{cases}\mathbf{u}(\mathbf{x},t) = 40\mathrm{ms}^{-1}, & (\mathbf{x},t)\in\mathring{\Omega}_h\times\{t=0\}\\ p(\mathbf{x},t) = 0, & (\mathbf{x},t)\in\mathring{\Omega}_h\times\{t=0\}\end{cases}, \qquad \begin{cases}\mathbf{u}(\mathbf{x},t) = 40\mathrm{ms}^{-1}, & \mathbf{x}\in\Gamma_{\text{inflow}}\\ \mathbf{n}\cdot\nabla p(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{inflow}}\end{cases},$$

$$\begin{cases}\mathbf{n}\cdot\nabla\mathbf{u}(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{outflow}}\\ p(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{outflow}}\end{cases}, \qquad \begin{cases}\mathbf{u}(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{Ahmed}}\cup\Gamma_{\text{bottom}}\\ \mathbf{n}\cdot\nabla p(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{Ahmed}}\cup\Gamma_{\text{bottom}}\end{cases},$$

$$\begin{cases}\mathbf{n}\cdot\nabla\mathbf{u}(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{sides}}\\ \mathbf{n}\cdot\nabla p(\mathbf{x},t) = 0, & \mathbf{x}\in\Gamma_{\text{sides}}\end{cases}$$

where the computational domain and its boundaries $\Gamma_{\text{outflow}},\Gamma_{\text{inflow}},\Gamma_{\text{Ahmed}},\Gamma_{\text{bottom}}$ and the remaining faces Γ_{sides} are shown in Figure 7.16. The computational mesh considered has a fixed number of cells for each geometrical deformation equal to $M = \mathbf{198633}$ for a total of $d = \mathbf{993165}$ dofs considering the average velocity, pressure and Eddy viscosity fields.

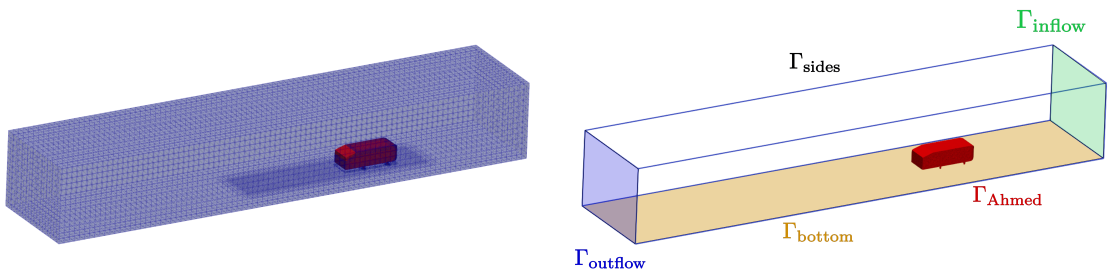


Fig. 7.16 **INS. Left:** computational domain of the incompressible turbulent flow past the Ahmed body test case. The mesh has **198633** cells, for a total of $d = 993165$ dofs for each geometrical deformation corresponding to a different choice of the slant angle θ. **Right:** description of the boundaries of the computational domain: a fixed velocity 40ms^{-1} and pressure $p = 0$ fields are imposed at the inflow boundary Γ_{inflow} for each geometrical deformation.

The `OpenFoam` solver we employ is the transient PISO numerical scheme. The predictor-corrector scheme is similar to the one described for the sonicFoam solver in subsection 7.5.1 and shown in Algorithm 10 in a simplified version for a comparison with the nonlinear manifold least-squares Petrov-Galerkin (NM-INS) method in Algorithm 11. As before, only we consider the n-th time instant and possible i-th intermediate optimization steps for the NM-INS method. First, the velocity field is obtained with an implicit predictor step in line 2, where the time discretization is hidden inside the diagonal A and over-diagonal $H[\mathbf{u}^*, \nu_t^n]$ parts of the finite volume discretization of the Reynolds averaged momentum equation (7.47a). Afterward, the velocity is corrected to satisfy the continuity equation (7.47b) through the kinematic pressure, obtained with a pressure-Poisson equation in line 5. Finally, the new Eddy viscosity is obtained.

The NM-INS method does not implement a predictor-correct strategy instead: only residual evaluations need to be computed and PISO and non-orthogonal corrector loops are omitted, as we search for the converged solutions corresponding to the last corrector steps. We remark that we don't consider the residual of the Eddy viscosity equation of the Spalart-Allmaras one-equation model in the NM-INS algorithm, but only the velocity and pressure ones. This choice and the difficulty of linearly approximating the Eddy viscosity, compared to the velocity and pressure fields, could be the reasons behind the worse accuracy in the predictions of the Eddy viscosity. For example, this can be observed in Figure 7.19.

As for the CNS test case, we consider a five times bigger time step for the nonlinear manifold method: instead of $\Delta t = 0.0001s$ employed for the full-order solutions, we set $\Delta t = 0.0005$ for the NM-INS method.

Fig. 7.17 **INS**. Comparison between the n-th time instant iterations of the full-order model numerical scheme PISO (**Left**) and the nonlinear manifold least-squares Petrov Galerkin (NM-INS)method (**Right**).

Algorithm 10: PISO n-th iteration

```
1 Start with an initial pressure field $p^n$, velocity field $\mathbf{u}^n$ and
  kinematic eddy viscosity $\nu_t^n$ at the $n$-th time step.
2 Momentum predictor step:
      $A\mathbf{u}^* = H[\mathbf{u}^*, \nu_t^n] - \nabla p^n.$
3 while PISO pressure-corrector loop do
4     while Non-orthogonal corrector loop do
5         Evaluate the pressure-corrector term $p^*$:
              $\nabla \cdot (A^{-1}\nabla p^*) = \nabla \cdot (A^{-1}H[\mathbf{u}^*, \nu_t^n]).$
6     Correct velocity
          $\mathbf{u}^* \leftarrow \mathbf{u}^* - A^{-1}\nabla p^*$
7 Solve for the kinematic Eddy viscosity.
```

Algorithm 11: NM-INS (i,n)-th residual

```
1 Start with an initial pressure field $p^{n,i}$, velocity field $\mathbf{u}^{n,i}$ and
  kinematic eddy viscosity $\nu_t^{n,i}$ at the $n$-th time step.
2 Momentum residual evaluation:
      $r_{\mathbf{u}} = A\mathbf{u}^{n,i} - H[\mathbf{u}^{n,i}, \nu_t^{n,i}] + \nabla p^{n,i}.$
3 Pressure-Poisson residual evaluation:
      $r_p = \nabla \cdot (A^{-1}\nabla p^{n,i}) - \nabla \cdot (A^{-1}H[\mathbf{u}^{n,i}, \nu_t^{n,i}]).$
4 Normalization of the residuals:
      $r_{\mathbf{u}} \leftarrow \frac{r_{\mathbf{u}}}{\max_i \mathbf{u}_i}, \quad r_p \leftarrow \frac{r_p}{\max_i p_i}$
```

Small geometrical deformations (INS-1)

As can be understood from the small geometrical deformations in Figure 7.18, the difficulty resides in the approximation of the transient dynamics rather than in the influence of the geometrical parameter. We consider **5** training slant angles and **4** test slant angles inside the training range (no extrapolation):

$$\mathcal{P}_{\text{train}} = \{15^\circ + i \cdot ((35^\circ - 15^\circ)/49) \,|\, i \in \{0,2,4,6,8\}\} \times V, \quad |\mathcal{P}_{\text{train}}| = 5, \tag{7.48a}$$

$$\mathcal{P}_{\text{test}} = \{15^\circ + i \cdot ((35^\circ - 15^\circ)/49) \,|\, i \in \{1,3,5,7\}\} \times V, \quad |\mathcal{P}_{\text{test}}| = 4,. \tag{7.48b}$$

In the order written in equations (7.48a) and (7.48b), we name the training and test parameters from 1 to 5 and from 1 to 4, respectively.

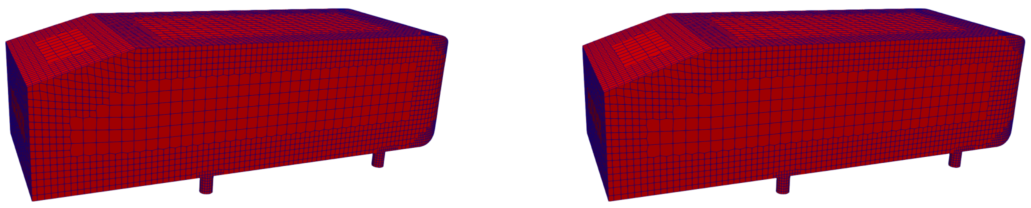


Fig. 7.18 **INS-1**. **Left:** Ahmed body with slant angle $\theta = 15.408^\circ$ from the training parameter set. **Right:** Ahmed body with slant angle $\theta = 18.265^\circ$ from the training parameter set.

As anticipated we employ a five times bigger time step $\Delta t = 0.0005$s with respect to the full-order one $\Delta t = 0.0001$s. We apply the C-UP-20 hyper-reduction with $r_h = 1500$ collocation nodes. For the test parameter 1, $\theta = 15.4^\circ$, the mean L_2 relative errors corresponding to the 1000 time instants from 0.0005s to 0.1s are shown in Figure 7.19. As can be seen, efficient and relatively accurate predictions can be obtained. The computational time spent is summarized in Table 7.3, reaching a speedup of around **26** with respect to the full-order model for this simple test case. Local nonlinear manifold approximations could be employed to achieve better predictions in the initial time instants, splitting the time interval.

The convergence with respect to the number of collocation nodes is shown in Figure 7.20 for the hyper-reduction methods C-UP-20 and C-UP-50 and collocation nodes $r_h \in \{500, 1000, 1500, 2000, 2500\}$. The corresponding timings are reported in Table 7.3 for a comparison with the full-order method. The computational cost of the submesh update can be reduced by restricting the collocation nodes that can be selected only to a neighborhood of the current submesh.

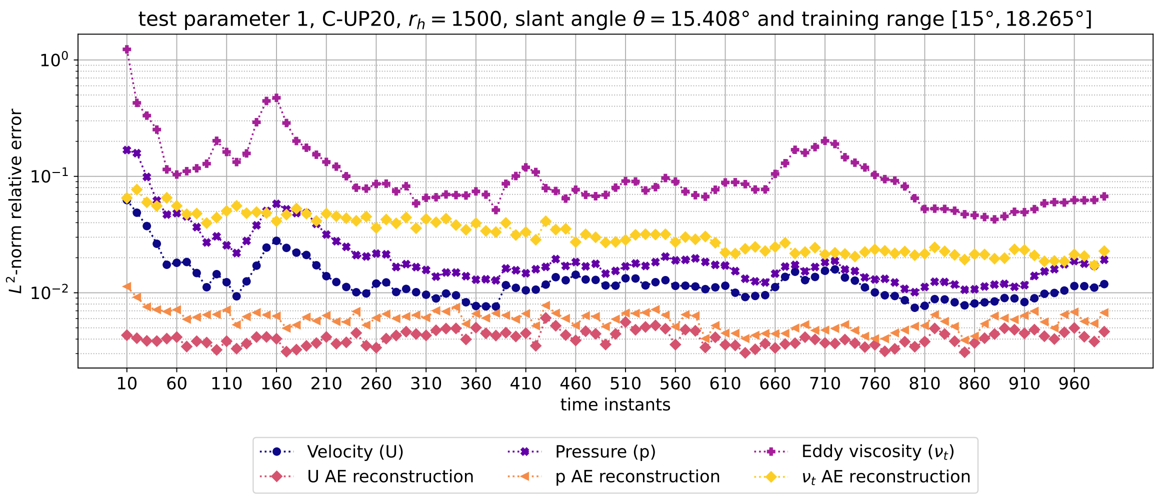


Fig. 7.19 **INS-1**. Every 10 time steps the L^2 relative error is reported for the predicted physical fields of interest associated with test parameter **1**, corresponding to the slant angle $\theta = 15.408°$ and the **C-UP20** method with $\mathbf{r_h} = \mathbf{1500}$ collocation nodes. The training parameter range is relatively small $[15°, 18.265°]$. The L^2 autoencoder relative reconstruction error represents an experimental lower bound to the prediction error. The accuracy is worse in the first time steps, a possible strategy to further reduce it is to employ local nonlinear manifold approximations as described in section 7.4.

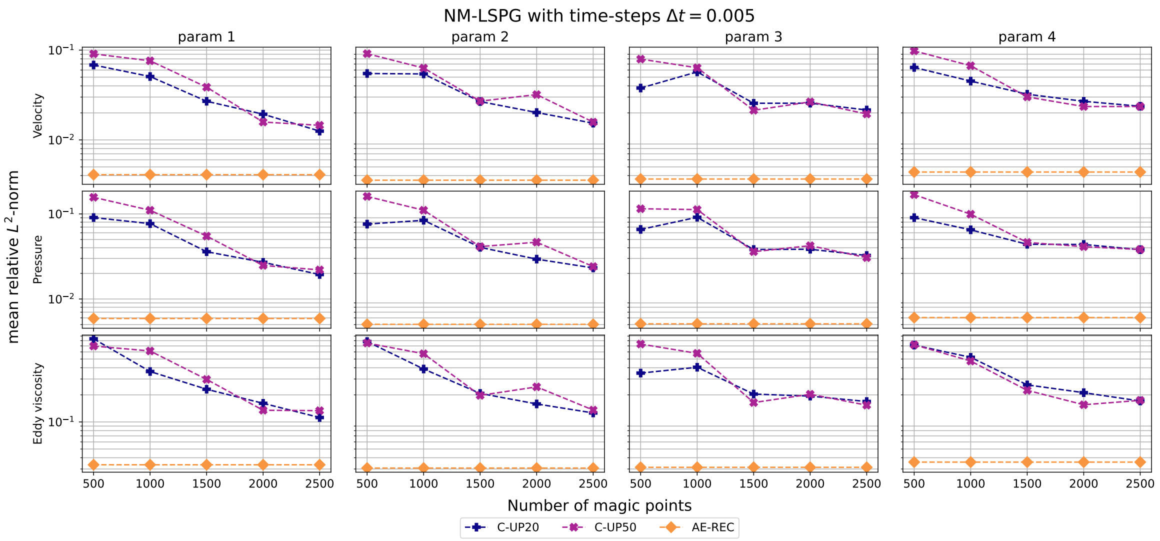


Fig. 7.20 **INS-1**. The decay of the mean L^2 relative error is assessed for the **C-UP20**, and **C-UP50** reduced over-collocation methods with sampling strategies corresponding to the gradient-based adaptive one every 50 and 20 time-steps, respectively. In this case, we employ a reference time step of four times $(\Delta t)_{\text{ref}} = 0.0005s$ the full-order model one $(\Delta t)_{\text{ref}} = 0.0001s$. The baseline represented by the autoencoder mean L^2 reconstruction error **AE-REC** is reported.

Table 7.3 **INS-1**. Timings of the INS for small geometrical deformations, subsection 4. The timings of the reduced over-collocation methods **C-UP20** (**left**) and **C-UP50** (**right**) relative to the choice of $r_h = \{500, 1000, 1500, 2000, 2500\}$ collocation nodes is reported along with the full-order model on one single CPU core (**FOM-1**) timings of the PISO solver. The time step of **C-UP20** and **C-UP50** is $(\Delta t)_{\text{ref}} = 0.0005$, five times bigger than the FOM with $(\Delta t)_{\text{ref}} = 0.0001$. The different reference time steps affect the average total time and not the mean time steps computational costs. The timings for the **FOM-8** run in parallel with 8 cores are wallclock times. The label $m.$ stands for mean and $m.UP$ stands for the mean computational time of the magic points and submesh updates.

MP	m. time-step	m. UP every 20	m. total time	m. time-step	m. UP every 50	m. total time
500	66.650 [ms]	660.136 [ms]	19.931 [s]	49.171 [ms]	572.604 [ms]	12.125 [s]
1000	116.432 [ms]	749.300 [ms]	30.779 [s]	136.320 [ms]	778.478 [ms]	30.378 [s]
1500	171.507 [ms]	809.385 [ms]	42.395 [s]	160.974 [ms]	842.653 [ms]	35.565 [s]
2000	212.034 [ms]	777.112 [ms]	50.178 [s]	194.183 [ms]	723.026 [ms]	41.729 [s]
2500	276.482 [ms]	933.621 [ms]	64.633 [s]	312.016 [ms]	973.697 [ms]	66.298 [s]
FOM-1	791.318 [ms]	-	13.189 [min]	791.318 [ms]	-	13.189 [min]
FOM-8	365.355 [ms]	-	3.589 [min]	365.355 [ms]	-	3.589 [min]

Large geometrical deformations (INS-2)

A situation where the methodology devised may fail is considered. One of the main problems of employing rSVD modes to linearly approximate solution manifolds with respect to nonlinear dimension reduction methods that employ neural networks, is a slow Kolmogorov n-width decay or, from a different point of view, the high generalization error on the test set. This is evident when increasing the dimension of the linear reduced space, the error on the training set decreases, but the error in the test set does not.

An example of this behavior is shown in Figure 7.22. This time we want to approximate the solution manifold corresponding to the parameter range for the slant angle $\theta \in [15°, 35°]$. We sample uniformly 50 slant angles:

$$\mathcal{P} = \{15° + i \cdot ((35° - 15°)/49) \,|\, i \in \{0, \dots, 49\}\} \times V, \quad |\mathcal{P}| = 50 \cdot 1000, \tag{7.49}$$

and, as usual, we number the parameters in order from 1 to 50. The training and test parameters of the previous section 4, correspond to the indices $\{1, 3, 5, 7, 9\}$ for the training slant angles and $\{2, 4, 6, 8\}$ for the test slant angles.

With reference to Figure 7.21, it is clear that a linear approximant of the whole solution manifold cannot be used, if the computational budget is limited to only **13** training time series. Increasing the computational budget, local linear reduced basis achieve substantial improvements. The results in terms of mean L^2 reconstruction error are reported in Figure 7.21 below. Each local linear solution manifold approximant is highlighted by shaded backgrounds of different colors. The leftmost corresponds to the parameters' range of the previous section 4.

Neural networks are known to overcome this problem with truly nonlinear dimension reduction algorithms, that is with respect to the methodology introduced in this work, without the direct involvement of linear rSVD basis. Possible hyper-reduction strategies that can be applied to a general truly nonlinear neural network architecture are studied in [223]. Nevertheless, the methodology presented in this work could effectively be applied for each of the four subdomains in Figure 7.22 with satisfactory accuracy in terms of test reconstruction error.

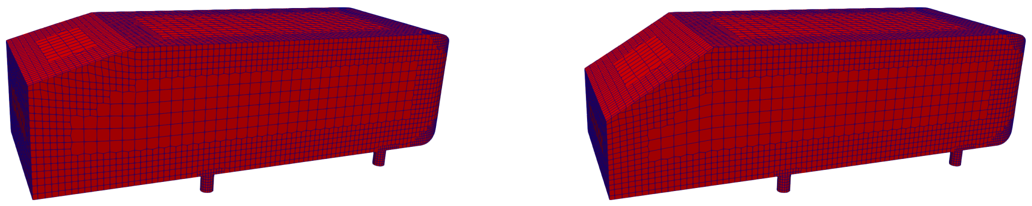


Fig. 7.21 **INS-2**. **Left:** Ahmed body with slant angle $\theta = 15^{\circ}$ from the training parameter set. **Right:** Ahmed body with slant angle $\theta = 35^{\circ}$ from the training parameter set.

7.5.3 Architectures

Table 7.4 The 1d Convolutional Autoencoder for the CNS1 on the coarse mesh in subsection 7. The label *Act* stands for Activation and *Pad*. for padding.

Encoder	Act	Weights	Pad.	Stride
Conv1d	ELU	[1, 8, 4, 4]	1	2
Conv1d	ELU	[8, 16, 4, 4]	1	2
Conv1d	ELU	[16, 32, 4, 4]	1	2
Conv1d	ELU	[32, 64, 4, 4]	1	2
Conv1d	ELU	[64, 128, 4, 4]	1	2
Linear	ELU	[512, 4]	-	

Decoder	Act	Weights	Pad.	Stride
Linear	ELU	[4, 512]	-	
ConvTr1d	ELU	[128, 64, 5, 5]	1	2
ConvTr1d	ELU	[64, 32, 4, 4]	1	2
ConvTr1d	ELU	[32, 16, 5, 5]	1	2
ConvTr1d	ELU	[16, 8, 5, 5]	1	2
ConvTr1d	ReLU	[8, 1, 4, 4]	1	2

Table 7.5 The 1d Convolutional Autoencoder for the CNS2 on the finer mesh in subsections 7 and for the INS1 in subsection 4. The label *Act* stands for Activation and *Pad*. for padding.

Encoder	Act	Weights	Pad.	Stride
Conv1d	ELU	[1, 8, 4, 4]	1	2
Conv1d	ELU	[8, 16, 4, 4]	1	2
Conv1d	ELU	[16, 32, 4, 4]	1	2
Conv1d	ELU	[32, 64, 4, 4]	1	2
Conv1d	ELU	[64, 128, 4, 4]	1	2
Linear	ELU	[512, 4]	-	

Decoder	Act	Weights	Pad.	Stride
Linear	ELU	[4, 512]	-	
ConvTr1d	ELU	[128, 64, 5, 5]	1	2
ConvTr1d	ELU	[64, 32, 4, 4]	1	2
ConvTr1d	ELU	[32, 16, 5, 5]	1	2
ConvTr1d	ELU	[16, 8, 5, 5]	1	2
ConvTr1d	ReLU	[8, 1, 4, 4]	1	2

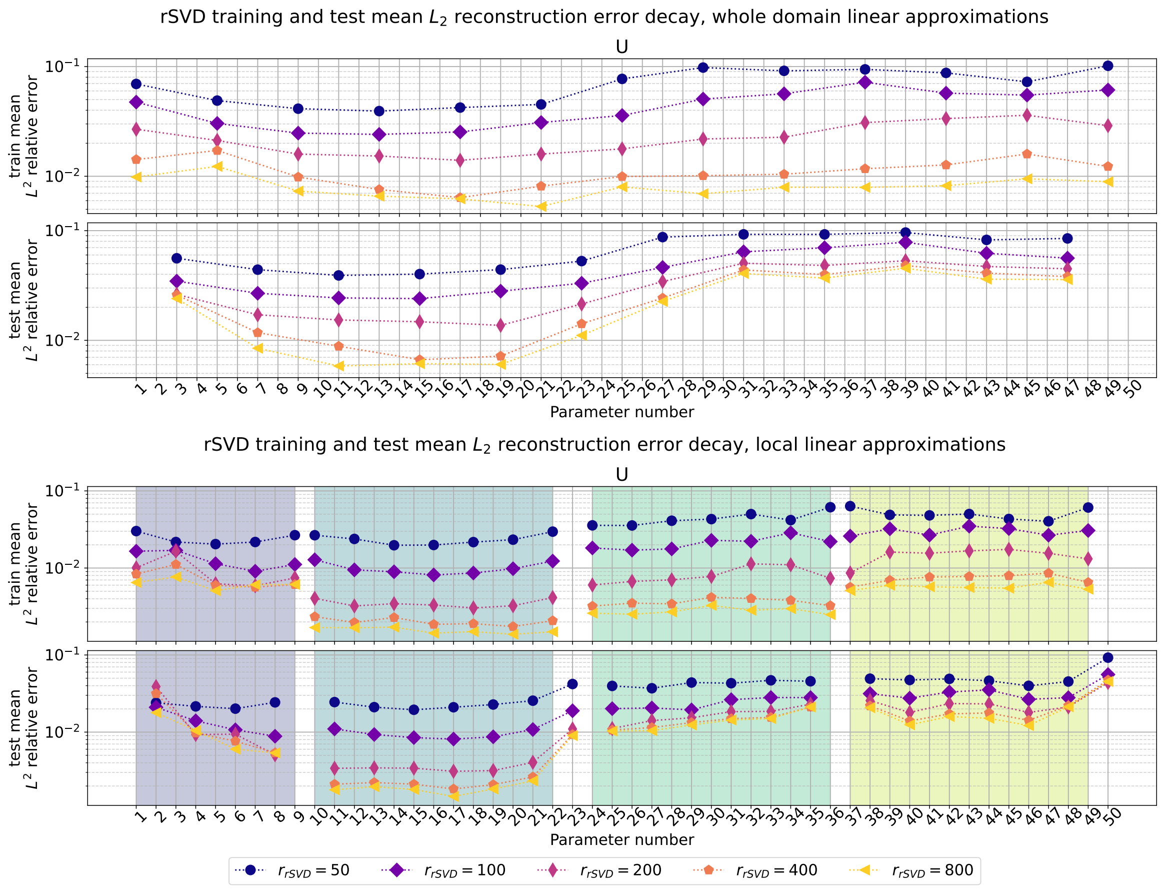


Fig. 7.22 **INS-2**. Studies of the decay of the mean L^2 reconstruction errors defined trough equations (7.17a) for the 50 geometrical deformations in (7.49). For brevity, we report only the errors on the velocity field. **Above:** approximation of the whole solution manifold with 13 training time series. It is evident the major drawback in employing linear approximants: the generalization error on the test set remains high even if the reduced dimension is increased from 50 to 800. **Below:** approximation of the solution manifold with 4 local linear approximants. They are highlighted by the shaded regions in background. The leftmost corresponds to the solution manifold studied in the previous section 4.

7.5.4 Predicted snapshots

Some snapshots associated to the test cases studied are reported: Figure 7.23 refers to the **CNS1** test case in line 7, Figure 7.24 refers to the **CNS2** test case in line 7 and finally Figures 7.25 and 7.27 refers to the test case **INS1** in line 4.

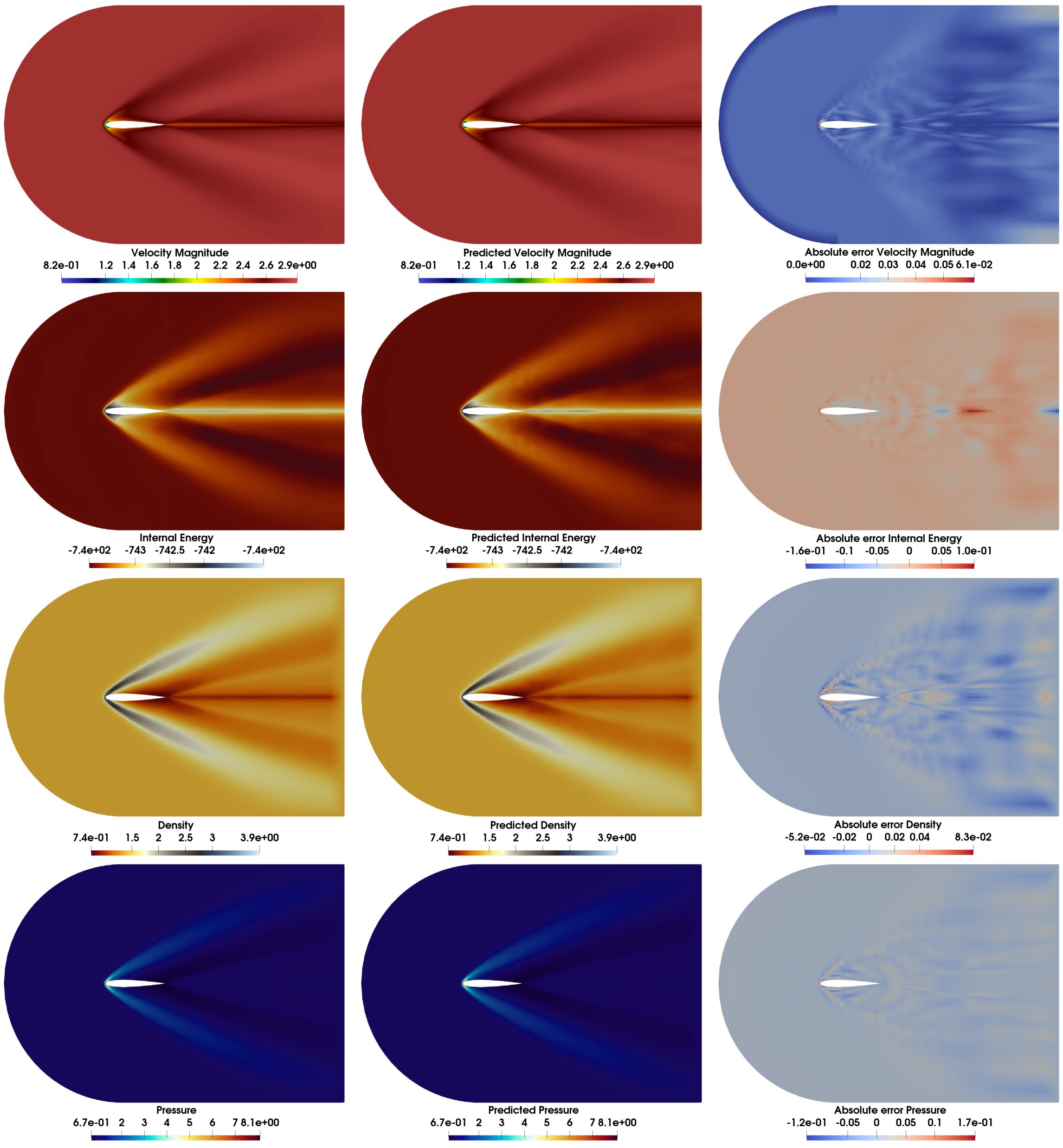


Fig. 7.23 **CNS1.** Predicted velocity, density, internal energy and pressure fields for the test case described in line 7 with $\mathrm{Ma} = 3.314$ corresponding to test parameter number **3** at the final time instant $t = 2.5$s. The number of cells is **4500** the total number of degrees of freedom is **27000**. The number of collocation nodes is $r_h = \mathbf{500}$

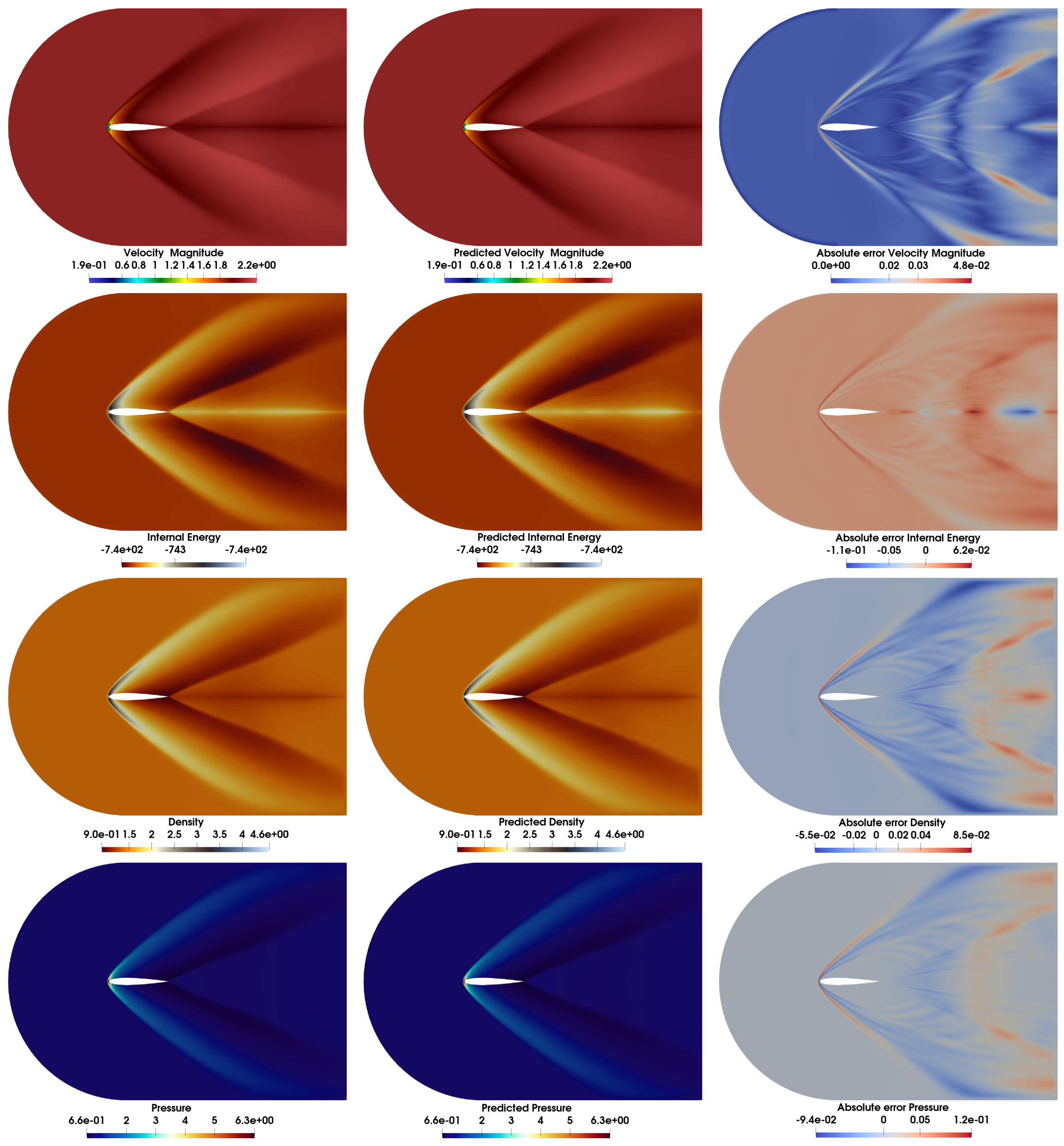


Fig. 7.24 **CNS2.** Predicted velocity, density, internal energy and pressure fields for the test case described in line 7 with $\mathrm{Ma} = 2.622$ corresponding to test parameter number **3** at the final time instant $t = 2.5\mathrm{s}$. The number of cells is **32160** the total number of degrees of freedom is **192960**. The adaptive magic points are shown in Figure 7.4. The number of collocation nodes is $r_h = \mathbf{700}$

7.6 Discussion

Some crucial considerations not yet underlined, are the subject of this section:

- choice of latent dimension. The latent dimensionality equal to $\mathbf{r} = \mathbf{4}$ and the number of rSVD modes equal to $\mathbf{r}_{\mathrm{rSVD}} = \mathbf{150}$ or $\mathbf{r}_{\mathrm{rSVD}} = \mathbf{300}$ are not chosen after parametric studies and therefore more optimal values for the problems at hand can be generally found. As long as we obtain a satisfactory approximation of the solution manifold for some values of the latent dimension of the autoencoder and of the reduced dimension of the linear projections, we did not change them, thus exploiting the possibility to employ the same neural networks architectures shown in Tables 7.4 and 7.5 for our numerical investigations. In fact, our focus is in the hyper-reduction methodologies applied after a relatively accurate nonlinear approximant of the solution manifold is obtained. Fixed a tolerance for the L^2 relative reconstruction error, the best value for r_{rSVD} can be efficiently found observing the decay of the reconstruction error as done in section 4 for the INS-2 test case. Also, the latent dimensionality of the autoencoder cannot be established *a priori*, for some theoretical bounds see [92]. Nevertheless, many empirical studies can be performed to determine the latent dimensionality [74].

- non-collocated hyper-reduction methods. Due to our choice of monolithic normalization and reduction of the physical fields of interest as described in section 7.2.3, the non-collocated versions of the hyper-reduction methods introduced in section 7.3 do not perform well in terms of accuracy as shown in Figure 7.9. The same gappy DEIM implementation performs well when applied in synergy with teacher-student training of a reduced decoder for a 2d nonlinear conservation law parametric model in [223], the only physical field considered is velocity, so normalization is not needed. Anyway, in terms of efficiency, collocation methods are faster since they do not perform additional matrix-vector multiplications (for DEIM and DEIM-SOPT) or scalar products (ECSW), after the evaluation of the residuals at the magic points. Moreover, employing adaptive hyper-reduction strategies in the online stage has higher computational costs for non-collocated approaches since pseudo-inverse matrices need to be evaluated (for DEIM).

- stability issues. Reduced over-collocation methods may bring unstable numerical schemes, especially if the solution manifold is not approximated with enough accuracy by the nonlinear approximant. Possible solutions involve the training of NN architectures with more smooth latent space and more regular maps such as those that should be guaranteed by variational autoencoders [148] or other machine learning architectures. On this line of thought, many additional inductive biases can be imposed. However, from the point of view of numerical analysis, stabilization strategies for ROMs have frequently been employed also for the classical projection-based methodologies as they often suffer from stability issues, especially when hyper-reduction is applied. Possible solutions include structure-preserving and/or regularized versions [149, 125] of hyper-reduction methods.

- inductive biases and autoencoder regularity. The training and the NN architecture itself can be enriched by inductive biases. Among them, there are first principles (e.g. conservation laws), geometrical simmetries (group equivariant filters [34]), latent operators/numerical schemes (e.g. operator inference [199, 258]) latent regularity (supposedly imposed by variational autoencoders and other architectures), structure-preservation [37], other numerical and mathematical properties (e.g. positiveness).

7.7 Conclusions

We developed and tested on challenging benchmarks a new method that exploits nonlinear solution manifold approximants. It relies on convolutional autoencoders and linear transforms/filter maps (specifically parallel randomized singular value decomposition) to approximate solution manifolds even when affected by a moderately slow Kolmogorov n-width decay. The main novelty resides in the implementation of new efficient collocated and adaptive gradient-based hyper-reduction strategies specifically tailored for our choice of nonlinear approximants. Local solution manifold approximations and efficient ways to perform the change of basis are also taken into consideration. We managed to achieve significant speedups while keeping a satisfactory accuracy even when we considered small benchmarks in terms of degrees of freedom, implemented in `OpenFoam`. Our test cases model complex physics such as the compressible and incompressible turbulent Navier-Stokes equations and suffer from a moderately slow Kolmogorov n-width decay that would need in the order of hundreds of linear basis to be well-approximated by classical projection-based ROMs.

The crucial objective that we want to reach with our methodology is the development of numerically and physically explicable model order reduction methods while exploiting machine learning architectures. The majority of scientific machine learning strategies employ neural networks in the predictive/online stage as black box surrogate models, without exploiting the underlying physics of the models embedded in the numerical schemes of the full-order models' solvers. Differently, our approach efficiently exploits the numerical schemes also in the predictive phase, with the possibility to characterize the latent solutions found as minimizers of the residuals of conservation laws. The interpretability of the results is evidently increased.

The main disadvantages regard the employment of linear basis: parametric models that suffer from a slow Kolmogorov n-width decay, such as the incompressible flow around the Ahmed body with geometrical deformations studied in section 4, may require too many computational resources to obtain local linear approximations of the solution manifold. The use of more generic truly nonlinear neural network architectures should bring lower generalization errors with less training data. Some implementations of hyper-reduction for generic NN architectures involve teacher-student training and are presented in [223].

Another aspect that can be substantially improved is stabilization issues: the development of stabilization mechanisms that would aid the nonlinear least-squares optimizers in the search for a physically accurate latent solution would greatly improve our methodology. On the same subject, structure-preserving and other regularizing frameworks, without mentioning additional useful inductive biases, would also help to achieve more accurate solutions.

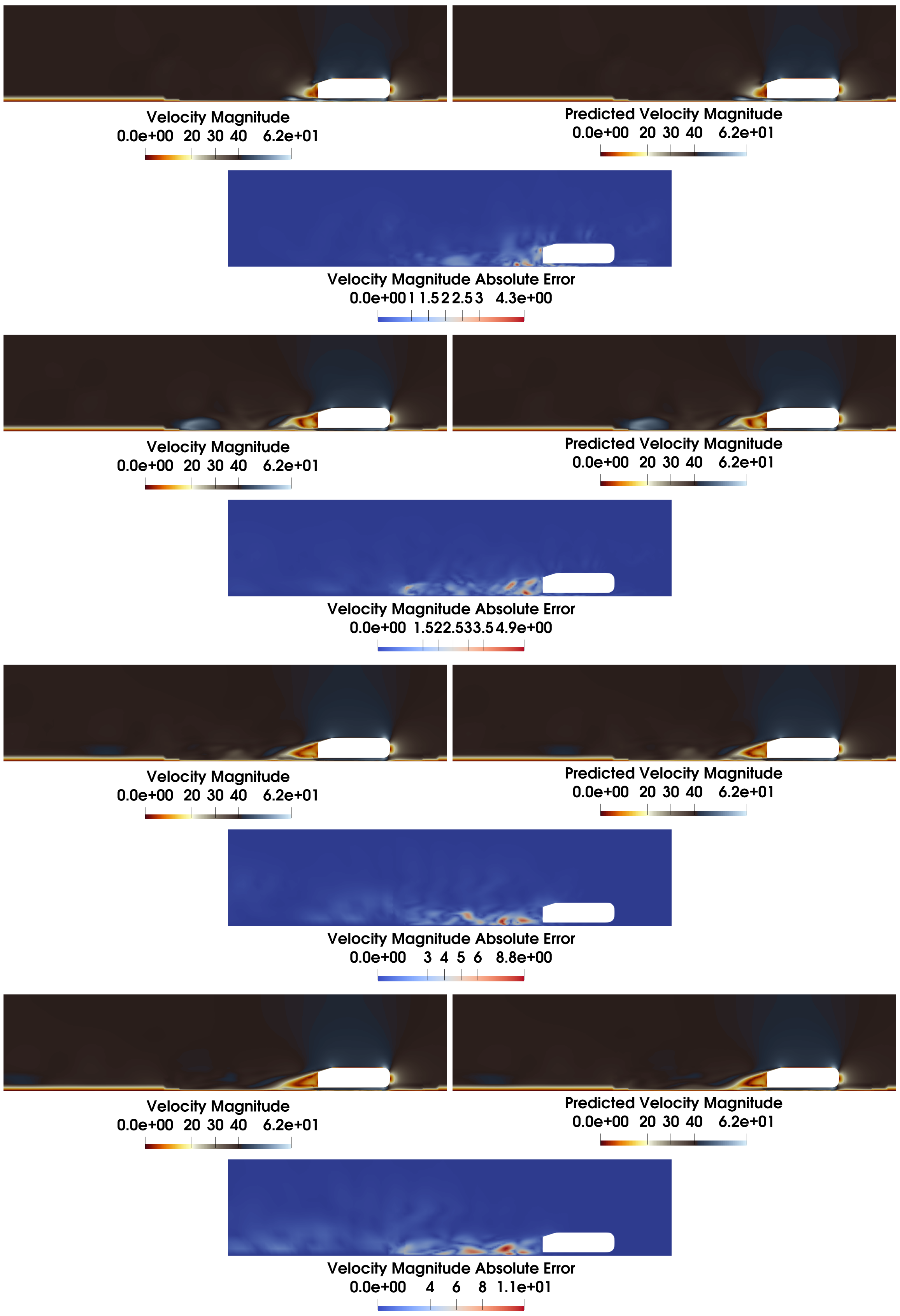


Fig. 7.25 **INS1.** Predicted velocity fields for the test case described in line 4 with corresponding to test parameter number **1** at the time instant s $t \in \{0.025\text{s}, 0.05\text{s}, 0.075\text{s}, 0.1\text{s}\}$. The number of cells is **32160** the total number of degrees of freedom is **192960**. The number of collocation nodes is $r_h = \mathbf{2000}$. The adaptive magic points are shown in Figure 7.27.

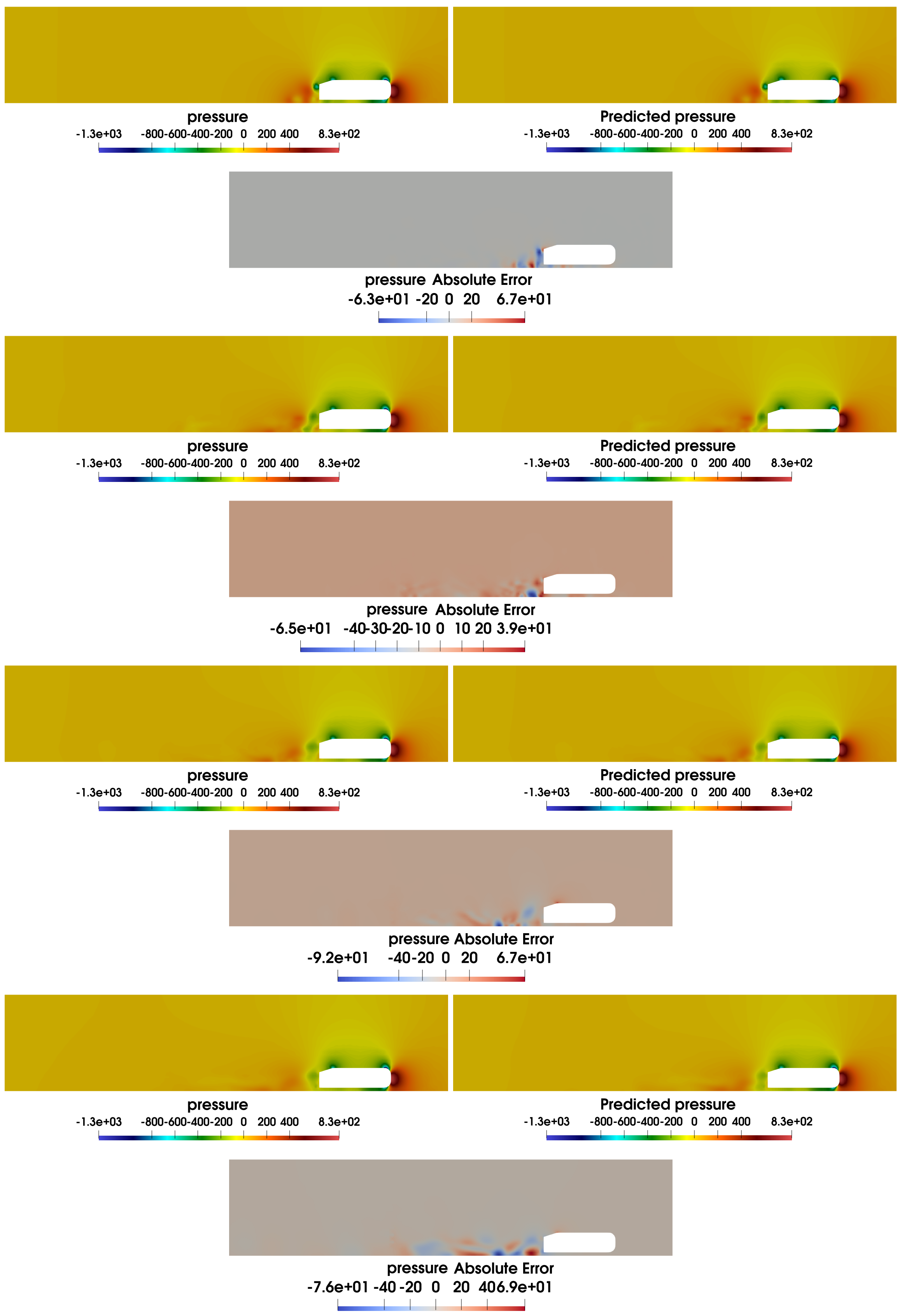


Fig. 7.26 **INS1.** Predicted pressure fields for the test case described in line 4 with corresponding to test parameter number **1** at the time instant s $t \in \{0.025\text{s}, 0.05\text{s}, 0.075\text{s}, 0.1\text{s}\}$. The number of cells is **32160** the total number of degrees of freedom is **192960**. The number of collocation nodes is $r_h = \mathbf{2000}$. The adaptive magic points are shown in Figure 7.27.

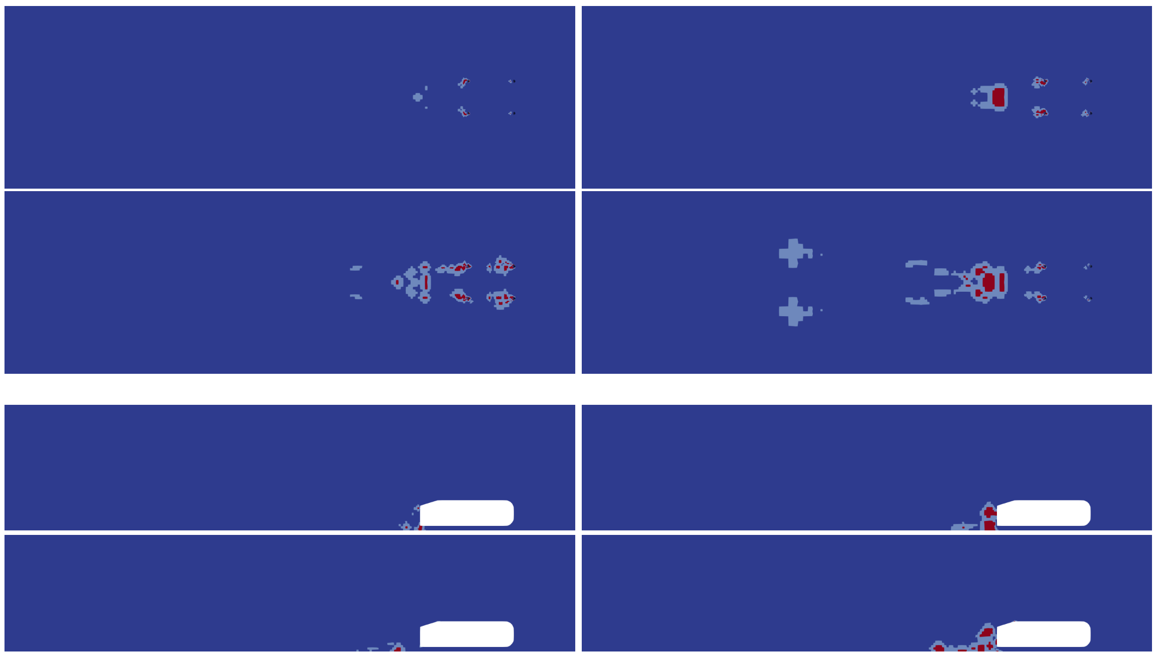


Fig. 7.27 **INS1.** Adaptive collocation nodes for the test case described in line 4 with corresponding test parameter number **1** at the time instants $t \in \{0.025\text{s}, 0.05\text{s}, 0.075\text{s}, 0.1\text{s}\}$. The number of collocation nodes is $r_h = \mathbf{2000}$. The number of cells is **198633** the total number of degrees of freedom is **993165**. **Above 4 slices:** bottom view of the Ahmed body. **Below 4 slices:** lateral view of the Ahmed body.

Chapter 8

Conclusions and final remarks

In this thesis, we have presented and validated through numerical experiments different methods for model order and parameter space reductions. They key strategy was to combine nonlinear dimension reduction methods from scientific machine learning with classical numerical methods. The results obtained rely on the increased expressiveness of neural networks with respect to linear approximants and the increased interpretability with respect to purely data-driven non-intrusive approaches.

Let us summarize some of the results:

- local linear approximants have been applied successfully in combination with the Active Subspaces method in chapter 2 for parameter space reduction, but also with the classical reduced basis method in chapter 5 for domain-decomposable ROMs and with nonlinear approximants in chapter 7. This also underlines the many analogies between model order and parameter space reductions. The notion of a local Grassmannian dimension, is used to define a new metric for clustering with the hierarchical top-down Active Subspaces in chapter 2, but also to define a new indicator for repartitioning the computational domain when performing MOR in chapter 5. Other analogies are confirmed by the application of other paradigms of nonlinear dimension reduction from machine learning to both the fields of model order and parameter space reductions, like kernel-based methods in [163] and [221], respectively.

- we have shown that the AS metric used to define the local Active Subspaces is effective for regression tasks, in combination with k-medoids or hierarchical top-down clustering algorithms. The subdivision of the parameter space and of the space of the outputs based on the local AS dimensions can be used to locally adapt the intrinsic dimension of the local response surfaces. The rationale behind this is similar to the hp-FEM but applied to data-driven surrogate modelling.

- we have shown that the presence of an Active Subspace can be employed as inductive bias for nonlinear multi-fidelity regression in chapter 3. In the low data regime, this strategy has been proven to be effective for data-driven surrogate modelling. It can also be employed successfully in combination with PODI for MOR as done in the work [252].

- a way to exactly impose multilinear constraints on generative models was proposed. The objective is not only to generate efficiently new geometries for real-world applications for which experimental data are missing but also to speed up the imposition of the constraints when the geometries are deformed manually. Typical contexts that may need cGMs are data assimilation and shape optimization studies for industrial and biomedical applications. Crucial is the validation of the posterior distributions with *ad hoc* metrics, as we have shown in chapter 4.

- we proposed Friedrichs' systems as a new framework for structure-preserving MOR. Among the advantages we include the possibility to obtain optimally stable error estimators, the reduction of PDEs of mixed elliptic and hyperbolic type, and the treatment in a unique formulation of many classes of complex, time-dependent and also nonlinear (through linearization) parametric PDEs.

- Domain Decomposable ROMs are developed for the DGM along with a posteriori error estimates for some Friedrichs' systems. They are based on distributed memory parallel full-order models' solvers. We showed how repartitioning strategies that achieve more tailored local approximations of the solution manifold can be implemented.

- a new paradigm for MOR is introduced with graph neural networks approximating vanishing viscosity solutions of hyperbolic PDEs, including some Friedrichs' systems. The approach is inspired by the proofs of convergence of high-viscosity solutions towards the vanishing viscosity one. The method is effective provided that the ROMs devised for the high viscosity regimes can be accurately approximated with low-dimensional linear spaces. We use GNNs to infer the limit of a succession of DD-ROMs predictions in real-time. Heavy autoencoders of GNNs that would be used for dimension reduction are avoided.

- we implemented efficient and accurate hyper-reduction methods for nonlinear manifold approximants. We applied them to convolutional autoencoders in chapter 6 but they can actually be extended to generic NN architectures so that the choice of nonlinear dimension reduction method is not a constraint. The hyper-reduction can be performed only on the residuals of the LSPG method or on both the residuals and the decoder maps with teacher-student training of a reduced decoder. Despite being more costly and more difficult to implement with respect to non-intrusive ROM alternatives, our new methodology offers the advantage of achieving a higher degree of interpretability and the possibility to study the results in a mathematical framework. In fact, the solutions are predicted minimizing first principles with numerical schemes related to the FOMs' solvers.

- we showed how, in some cases, high dimensional linear approximants, obtained from a truncated SVD of the snapshots, can be used, combined with convolutional autoencoders, as nonlinear manifold approximants. Similarly to the methodology devised in chapter 6, we use this new nonlinear approximant of the solution manifold inside numerical schemes in chapter 7. The advantage is that new adaptive hyper-reduction strategies can be implemented along with local nonlinear approximations of the solution manifold. In fact, with the employment of SVD modes, there is no need to further compress the decoder with teacher-student training as it is already sufficiently small. The evaluation of adaptive magic points for hyper-reduction and the computation of the change of basis between adjacent local nonlinear manifolds can be now performed efficiently.

- we briefly summarize the numerical test cases we have implemented to test the novel methodologies: an elliptic PDE with the permeability parametrized by the Karhunen-Loéve expansion's coefficients distributed as a standard multivariate Gaussian and the unsteady incompressible Navier-Stokes equations past a NACA airfoil in chapter 2; DDES and RANS for automotive shape optimization in chapter 3; multiphase RANS for a naval engineering test case in chapter 4; the advection–diffusion–reaction of a scalar concentration advected by the incompressible Navier-Stokes equations with different discontinuous inflow profiles, the Maxwell equations in stationary regime and the linear compressible elasticity both with discontinuous parameters, all interpreted as Friedrichs' systems in chapter 5; a nonlinear time-dependent conservation law and the shallow water equations both with parametrized initial conditions in chapter 6; a compressible flow past a NACA airfoil with different Mach numbers and an incompressible turbulent flow around the Ahmed body with different slant angles in chapter 7. The full-order numerical solvers are implemented in `OpenFoam` [264] or `deal.II` [12] for the FVM and DGM, respectively. While the open-source libraries used for parameter space and model order reductions are `ITHACA-FV` [237, 238] and `ATHENA` [218], respectively.

8.1 Future perspectives

During the investigation of the different methodologies presented in this thesis, we have identified several research lines that should be further studied and developed. We briefly report some of them.

With due precautions moving from one topic to the other, nonlinear parameter space reduction methods are used also in the investigation of new ways to approximate probability distributions. This is of great importance for data assimilation, inverse problems and uncertainty quantification in general. The aim is to lower the computational costs of Monte Carlo methods or data assimilation techniques through nonlinear dimension reduction methods, possibly involving architectures like variational autoencoders, generative models, normalizing flows, and transport maps [195]. This is an active field of research where the synergies between numerical methods and scientific ML, highlighted also in this thesis, produce relevant results.

Now that an efficient way to employ the nonlinear manifold LSPG method introduced in [161] has been developed with two possible variants introduced in chapters 6 and 7, we hope that new theoretical studies can be performed. Differently from black-box non-intrusive ROMs, now the methodology itself is framed in a mathematically formal environment in the online stage. Possible future directions of research include the addition of informative inductive biases, the use of different NN architectures, the development of stabilization and regularization mechanisms for hyper-reduction and the imposition of structure preservation for some specific models. Enlarging the point of view beyond MOR, other newly developed methods exploit a combination of NN architectures with numerical schemes: among them physics-informed neural networks [44], operator inference [208], machine-learning-accelerated numerical simulations [150], for a general review [261].

We believe that the employment of scientific machine learning formalized and adapted to numerical applications with the tools of numerical analysis will bring more interpretable and effective methodologies for the challenges of the future in model order reduction but also numerical modelling in general.

Acknowledgements

It has been a long journey even though the perceived time has passed quickly. It feels like just last month when I started. I don't hide that I have some regrets, but the balance of these past years is clearly positive, considering we've overcome a pandemic in the meantime. I wouldn't have been able to accomplish this feat on my own without the support of many people who encouraged me and created an environment that never let me lose my curiosity to explore new areas and my passion for research, even in the most critical moments.

My first thank you goes to Gianluigi for his humanity and for reminding me that before being talented researchers, we need to be good individuals. I thank Marco for believing in me from the very beginning, during the toughest times, and for teaching me the fundamentals and the most useful methods to do research. I thank Giovanni for his scientific input and for always being a point of reference I could rely on in times of need. I thank Davide for supporting me in my undoubtedly too ambitious endeavors and for being a companion in adventures and unexpected situations. I would also like to thank Markus and Carsten for their valuable contributions to some of the results of this thesis.

SISSA has been a pleasant place to spend long days of study and programming, mainly thanks to the people who surrounded me. A special thank you goes to the entire mathLab and my colleagues from AMMA: Ivan, Simone, Davide M., M.F., G.M.L., L.G., L.F., N.T., Warren, Riccardo, Ale Tama, Moaad, Pier, Nicola, Anna, Pasquale, Laura, Caterina, Michele, Paolo, Fabio, Martina, Giuseppe. Also to those scattered around: Francesco Roma., Giulio, Moreno, Maria, Federico. A thank you goes to the Master's students I have supervised Guglielmo and Giorgio. And to the friends from Trieste, especially Rocco.

To my family, the thanks of a thesis are not enough, now as always. I owe everything to my parents, in this context, especially for not hindering my perhaps too audacious choices and always providing me with an objective point of comparison and a reference on how to fully live one's life and what truly important values are. Making my choices was possible and easier because I could draw inspiration from following in the footsteps of my brother, to whom my respect and gratitude always go for constantly stimulating me.

The biggest thank you goes to Rara for literally being my compass and diary every day, always showing me how to rise again and find refuge, but also how to fill the journey with joy and meaning.

References

[1] Autoencoders. In *Deep Learning in Science*, pages 71–98. Cambridge University Press, April 2021. doi:10.1017/9781108955652.006. URL https://doi.org/10.1017/9781108955652.006.

[2] Upscale (upscaling product development simulation capabilities exploiting artificial intelligence for electric vehicles). https://www.upscaleproject.eu/, 2021. [Online; accessed 2023-01-04].

[3] Ira H Abbott and Albert E Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Courier Corporation, 2012.

[4] Maria Àngels Cerveró Abelló, Álvaro Vinacua Pla, and Pere Brunet Crosa. Volume-preserving deformation using generalized barycentric coordinates. 2010 (conference report). URL https://api.semanticscholar.org/CorpusID:45335879.

[5] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/9ce3c52fc54362e22053399d3181c638-Paper.pdf.

[6] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006. doi:10.1016/j.parco.2005.07.004.

[7] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *Aiaa Journal - AIAA J*, 46:1803–1813, 07 2008. doi:10.2514/1.35374.

[8] Nenad Antonic and Krešimir Burazin. On equivalent descriptions of boundary conditions for Friedrichs systems. *Math. Montisnigri*, 22(23):5–13, 2009.

[9] Nenad Antonić, Krešimir Burazin, Ivana Crnjac, and Marko Erceg. Complex Friedrichs systems and applications. *Journal of Mathematical Physics*, 58(10):101508, 2017. doi:10.1063/1.5005608.

[10] Florian Arbes, Constantin Greif, and Karsten Urban. The Kolmogorov N-width for linear transport: Exact representation and the influence of the data. *arXiv preprint arXiv:2305.00066*, 2023.

[11] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017. URL https://arxiv.org/abs/1701.04862.

[12] Daniel Arndt, Wolfgang Bangerth, Bruno Blais, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Uwe Köcher, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Sebastian Proell, Konrad Simon, Bruno Turcksin, David Wells, and Jiaqi Zhang.

The `deal.II` library, version 9.3. *Journal of Numerical Mathematics*, 29(3):171–186, 2021. doi:10.1515/jnma-2021-0081. URL https://dealii.org/deal93-preprint.pdf.

[13] Ivo Babuška, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45 (3):1005–1034, 2007.

[14] Joan Baiges, Ramon Codina, and Sergio Idelsohn. A domain decomposition strategy for reduced order models. application to the incompressible navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 267:23–42, 2013. doi:10.1016/j.cma.2013.08.001.

[15] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[16] Satish Balay, Shrirang Abhyankar, Mark Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, W Gropp, et al. Petsc users manual. 2019.

[17] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. https://petsc.org/, 2023. URL https://petsc.org/.

[18] Wolfgang Bangerth, Carsten Burstedde, Timo Heister, and Martin Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Trans. Math. Softw.*, 38:14/1–28, 2011. doi:10.1145/2049673.2049678.

[19] Joshua L Barnett, Charbel Farhat, and Yvon Maday. Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility of cfd models. *arXiv preprint arXiv:2212.08939*, 2022.

[20] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[21] Kinjal Basu and Art B Owen. Transformations and Hardy–Krause Variation. *SIAM Journal on Numerical Analysis*, 54(3):1946–1966, 2016. doi:10.1137/15M1052184.

[22] Christopher Beattie, Serkan Gugercin, and Volker Mehrmann. Structure-preserving interpolatory model reduction for port-hamiltonian differential-algebraic systems. In *Realization and Model Reduction of Dynamical Systems: A Festschrift in Honor of the 70th Birthday of Thanos Antoulas*, pages 235–254. Springer, 2022. doi:10.1007/978-3-030-95157-3_13.

[23] Einat Neumann Ben-Ari and David M Steinberg. Modeling data from computer experiments: an empirical comparison of kriging with MARS and projection pursuit regression. *Quality Engineering*, 19(4):327–338, 2007. doi:10.1080/08982110701580930.

[24] P. Benner, M. Ohlberger, A. Patera, G. Rozza, and K. Urban. *Model Reduction of Parametrized Systems*, volume 17 of *MS&A series*. Springer, 2017. doi:10.1007/978-3-319-58786-8.

[25] Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017. doi:10.1137/1.9781611974829.

[26] Florian Bernard, Angelo Iollo, and Sébastien Riffaud. Reduced-order model for the BGK equation based on POD and optimal transport. *Journal of Computational Physics*, 373:545–570, 2018. doi:10.1016/j.jcp.2018.07.001.

[27] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks, 2017 (preprint). URL https://arxiv.org/abs/1703.10717.

[28] Emil Beurer, Moritz Feuerle, Niklas Reich, and Karsten Urban. An ultraweak variational method for parameterized linear differential-algebraic equations. *Frontiers in Applied Mathematics and Statistics*, 8:910786, 2022. doi:10.3389/fams.2022.910786.

[29] Satyaki Bhattacharjee and Karel Matouš. A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. *Journal of Computational Physics*, 313:635–653, 2016. doi:10.1016/j.jcp.2016.01.040.

[30] Gabriele Boncoraglio, Charbel Farhat, and Charbel Bou-Mosleh. Model Reduction Framework with a New Take on Active Subspaces for Optimization Problems with Linearized Fluid-Structure Interaction Constraints. *International Journal for Numerical Methods in Engineering*, 2020. doi:10.1002/nme.6376.

[31] L Bonfiglio, P Perdikaris, S Brizzolara, and GE Karniadakis. Multi-fidelity optimization of super-cavitating hydrofoils. *Computer Methods in Applied Mechanics and Engineering*, 332: 63–85, 2018. doi:10.1016/j.cma.2017.12.009.

[32] Luca Bonfiglio, Paris Perdikaris, Giuliano Vernengo, João Seixas de Medeiros, and George Karniadakis. Improving SWATH Seakeeping Performance using Multi-Fidelity Gaussian Process and Bayesian Optimization. *Journal of Ship Research*, 62(4):223–240, 2018. doi:10.5957/JOSR.11170069.

[33] Robert A Bridges, Anthony D Gruber, Cristopher R Felder, Miki Verma, and Chelsey Hoff. Active Manifolds: A non-linear analogue to Active Subspaces. In *Proceddings of the 36th International Conference on Machine Learning, ICML 2019*, pages 764–772, Long Beach, California, USA, 9–15 June 2019.

[34] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017. doi:10.1109/MSP.2017.2693418.

[35] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019. doi:10.1017/9781108380690.

[36] Steven L Brunton, J Nathan Kutz, Krithika Manohar, Aleksandr Y Aravkin, Kristi Morgansen, Jennifer Klemisch, Nicholas Goebel, James Buttrick, Jeffrey Poskin, Adriana W Blom-Schieber, et al. Data-driven aerospace engineering: reframing the industry with machine learning. *AIAA Journal*, 59(8):2820–2847, 2021. doi:10.2514/1.J060131.

[37] Patrick Buchfink, Silke Glas, and Bernard Haasdonk. Symplectic model reduction of hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder. *SIAM Journal on Scientific Computing*, 45(2):A289–A311, 2023. doi:10.1137/21M1466657.

[38] Marcelo Buffoni, Haysam Telib, and Angelo Iollo. Iterative methods for model reduction by domain decomposition. *Computers & Fluids*, 38(6):1160–1167, 2009. doi:10.1016/j.compfluid.2008.11.008.

[39] Tan Bui-Thanh, Leszek Demkowicz, and Omar Ghattas. Constructively well-posed approximation methods with unity inf–sup and continuity constants for partial differential equations. *Mathematics of Computation*, 82(284):1923–1952, 2013. doi:10.1090/S0025-5718-2013-02697-X.

[40] Tan Bui-Thanh, Leszek Demkowicz, and Omar Ghattas. A unified discontinuous Petrov–Galerkin method and its analysis for Friedrichs' systems. *SIAM Journal on Numerical Analysis*, 51(4):1933–1958, 2013. doi:10.1137/110854369.

[41] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[42] Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011. doi:10.1137/100791634.

[43] N. Cagniart, Y. Maday, and B. Stamm. *Model Order Reduction for Problems with Large Convection Effects*, pages 131–150. Springer International Publishing, Cham, 2019. ISBN 978-3-319-78325-3. doi:10.1007/978-3-319-78325-3_10. URL https://doi.org/10.1007/978-3-319-78325-3_10.

[44] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021. doi:10.1007/s10409-021-01148-1.

[45] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011. doi:10.1002/nme.3050.

[46] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, Jun 2013. ISSN 0021-9991. doi:10.1016/j.jcp.2013.02.028. URL http://dx.doi.org/10.1016/j.jcp.2013.02.028.

[47] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi:10.1609/aaai.v32i1.11668.

[48] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010. doi:10.1137/090766498.

[49] Jau-Uei Chen, Shinhoo Kang, Tan Bui-Thanh, and John N Shadid. Unified *hp*-hdg frameworks for friedrichs' pde systems. *arXiv preprint arXiv:2304.03690*, 2023.

[50] Peng Chen and Omar Ghattas. Hessian-based sampling for high-dimensional model reduction. *International Journal for Uncertainty Quantification*, 9(2), 2019. doi:10.1615/Int.J.UncertaintyQuantification.2019028753.

[51] Yanlai Chen, Sigal Gottlieb, Lijie Ji, and Yvon Maday. An eim-degradation free reduced basis method via over collocation and residual hyper reduction-based error estimation. *Journal of Computational Physics*, 444:110545, 2021. doi:10.1016/j.jcp.2021.110545.

[52] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. Meshgan: Non-linear 3d morphable models of faces, 2019 (preprint). URL https://arxiv.org/abs/1903.10384.

[53] F. Chinesta, P. Ladeveze, and E. Cueto. A Short Review on Model Order Reduction Based on Proper Generalized Decomposition. *Archives of Computational Methods in Engineering*, 18 (4):395, 2011. doi:10.1007/s11831-011-9064-7.

[54] F. Chinesta, A. Huerta, G. Rozza, and K. Willcox. Model reduction methods. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics, Second Edition*, pages 1–36. John Wiley & Sons, Ltd., 2017. doi:10.1002/9781119176817.ecm2110.

[55] Youngsoo Choi and Kevin Carlberg. Space–time least-squares Petrov–Galerkin projection for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 41(1):A26–A58, 2019. doi:10.1137/17M1120531.

[56] Youngsoo Choi, Deshawn Coombs, and Robert Anderson. SNS: a solution-based nonlinear subspace method for time-dependent model order reduction. *SIAM Journal on Scientific Computing*, 42(2):A1116–A1146, 2020. doi:10.1137/19M1242963.

[57] Jorio Cocola, John Tencer, Francesco Rizzi, Eric Parish, and Patrick Blonigan. Hyper-reduced autoencoders for efficient and accurate nonlinear model reductions. *arXiv preprint arXiv:2303.09630*, 2023.

[58] Albert Cohen and Ronald DeVore. Approximation of high-dimensional parametric PDEs. *Acta Numerica*, 24:1–159, 2015. doi:10.1017/S0962492915000033.

[59] Albert Cohen and Ronald DeVore. Kolmogorov widths under holomorphic mappings. *IMA Journal of Numerical Analysis*, 36(1):1–12, 2016. doi:10.1093/imanum/dru066.

[60] Paul G Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, volume 2 of *SIAM Spotlights*. SIAM, 2015. doi:10.1137/1.9781611973860.

[61] Paul G Constantine and Paul Diaz. Global sensitivity metrics from active subspaces. *Reliability Engineering & System Safety*, 162:1–13, 2017. doi:10.1016/j.ress.2017.01.013.

[62] Paul G Constantine, Carson Kent, and Tan Bui-Thanh. Accelerating Markov Chain Monte Carlo with Active Subspaces. *SIAM Journal on Scientific Computing*, 38(5):A2779–A2805, 2016. doi:10.1137/15M1042127.

[63] Andrea F Cortesi, Paul G Constantine, Thierry E Magin, and Pietro M Congedo. Forward and backward uncertainty quantification with active subspaces: application to hypersonic flows around a cylinder. *Journal of Computational Physics*, 407:109079, 2020. doi:10.1016/j.jcp.2019.109079.

[64] Roxana Crisovan, Davide Torlo, Rémi Abgrall, and S Tokareva. Model order reduction for parametrized nonlinear hyperbolic problems as an application to uncertainty quantification. *Journal of Computational and Applied Mathematics*, 348:466–489, 2019. doi:10.1016/j.cam.2018.09.018.

[65] Lisandro Dalcin and Yao-Lung L. Fang. mpi4py: Status update after 12 years of development. *Computing in Science and Engineering*, 23(4):47–54, 2021. doi:10.1109/MCSE.2021.3083216.

[66] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124 – 1139, 2011. ISSN 0309-1708. doi:10.1016/j.advwatres.2011.04.013. New Computational Methods and Software Tools.

[67] Gregory A. Daly, Jonathan E. Fieldsend, and Gavin Tabor. Variational autoencoders without the variation, 2022. URL https://arxiv.org/abs/2203.00645.

[68] Andreas Damianou and Neil Lawrence. Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[69] Thomas Daniel, Fabien Casenave, Nissrine Akkari, and David Ryckelynck. Model order reduction assisted by deep neural networks (ROM-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7(1):1–27, 2020. doi:10.1186/s40323-020-00153-6.

[70] Aukje De Boer, Martijn S Van der Schoot, and Hester Bijl. Mesh deformation based on radial basis function interpolation. *Computers & structures*, 85(11-14):784–795, 2007. doi:10.1016/j.compstruc.2007.01.013.

[71] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. A non-intrusive approach for reconstruction of POD modal coefficients through active subspaces. *Comptes Rendus Mécanique de l'Académie des Sciences, DataBEST 2019 Special Issue*, 347(11):873–881, November 2019. doi:https://doi.org/10.1016/j.crme.2019.11.012.

[72] Nicola Demo, Marco Tezzele, Andrea Mola, and Gianluigi Rozza. Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing. *Journal of Marine Science and Engineering*, 9(2):185, 2021. doi:10.3390/jmse9020185.

[73] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems. *SIAM Journal on Scientific Computing*, 43(3):B831–B853, 2021. doi:10.1137/20M1345219.

[74] Francesco Denti, Diego Doimo, Alessandro Laio, and Antonietta Mira. The generalized ratios intrinsic dimension estimator. *Scientific Reports*, 12(1):20005, 2022. doi:10.1038/s41598-022-20991-1.

[75] Ronald A DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta mathematica*, 63(4):469–478, 1989. doi:10.1007/BF01171759.

[76] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media, 2011. doi:10.1007/978-3-642-22980-0.

[77] Alejandro N Diaz, Youngsoo Choi, and Matthias Heinkenschloss. A fast and accurate domain-decomposition nonlinear manifold reduced order model. *arXiv preprint arXiv:2305.15163*, 2023.

[78] Paul Diaz, Paul Constantine, Kelsey Kalmbach, Eric Jones, and Stephen Pankavich. A modified SEIR model for the spread of Ebola in Western Africa and metrics for resource allocation. *Applied Mathematics and Computation*, 324:141–155, 2018. doi:10.1016/j.amc.2017.11.039.

[79] Pedro Díez, Alba Muixí, Sergio Zlotnik, and Alberto García-González. Nonlinear dimensionality reduction for parametric problems: a kernel Proper Orthogonal Decomposition (kPOD). *arXiv preprint arXiv:2104.13765*, 2021. doi:10.23967/admos.2021.002.

[80] Ronald J DiPerna. Convergence of approximate solutions to conservation laws. In *Transonic, Shock, and Multidimensional Flows*, pages 313–328. Elsevier, 1982. doi:10.1016/B978-0-12-493280-7.50018-8.

[81] Ronald J DiPerna. Convergence of the viscosity method for isentropic gas dynamics. *Communications in mathematical physics*, 91(1):1–30, 1983. doi:10.1007/BF01206047.

[82] JL Eftang, DBP Huynh, DJ Knezevic, EM Ronquist, and AT Patera. Adaptive port reduction in static condensation. *IFAC Proceedings Volumes*, 45(2):695–699, 2012. doi:10.3182/20120215-3-AT-3016.00123.

[83] M. Eisenberger, Z. Lähner, and D. Cremers. Divergence-free shape correspondence by deformation. *Computer Graphics Forum*, 38(5):1–12, August 2019. doi:10.1111/cgf.13785. URL https://doi.org/10.1111/cgf.13785.

[84] Alexandre Ern and Jean-Luc Guermond. Discontinuous Galerkin Methods for Friedrichs' Systems. Part II. Second-order Elliptic PDEs. *SIAM Journal on Numerical Analysis*, 44(6): 2363–2388, January 2006. doi:10.1137/05063831x. URL https://doi.org/10.1137/05063831x.

[85] Alexandre Ern and Jean-Luc Guermond. Discontinuous Galerkin methods for Friedrichs' systems. I. General theory. *SIAM journal on numerical analysis*, 44(2):753–778, 2006. doi:10.1137/050624133.

[86] Alexandre Ern and Jean-Luc Guermond. Discontinuous Galerkin methods for Friedrichs' systems. Part III. Multifield theories with partial coercivity. *SIAM journal on numerical analysis*, 46(2):776–804, 2008. doi:10.1137/060664045.

[87] Alexandre Ern, Jean-Luc Guermond, and Gilbert Caplain. An intrinsic criterion for the bijectivity of Hilbert operators related to Friedrich' systems. *Communications in partial differential equations*, 32(2):317–341, 2007. doi:10.1080/03605300600718545.

[88] Charbel Farhat, Todd Chapman, and Philip Avery. Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. *International journal for numerical methods in engineering*, 102(5):1077–1110, 2015. doi:10.1002/nme.4820.

[89] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[90] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.

[91] Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269, 2007. doi:10.1098/rspa.2007.1900.

[92] Nicola Franco, Andrea Manzoni, and Paolo Zunino. A deep learning approach to reduced order modelling of parameter dependent partial differential equations. *Mathematics of Computation*, 92(340):483–524, 2023. doi:10.1090/mcom/3781.

[93] Nicola Rares Franco, Andrea Manzoni, and Paolo Zunino. Learning operators with mesh-informed neural networks. *arXiv preprint arXiv:2203.11648*, 2022.

[94] Thomas Franz, Ralf Zimmermann, Stefan Görtz, and Niklas Karcher. Interpolation-based reduced-order modelling for steady transonic flows via manifold learning. *International Journal of Computational Fluid Dynamics*, 28(3-4):106–121, 2014. doi:10.1080/10618562.2014.918695.

[95] Stefania Fresca and Andrea Manzoni. POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022. doi:10.1016/j.cma.2021.114181.

[96] Kurt Otto Friedrichs. Symmetric positive linear differential equations. *Communications on Pure and Applied Mathematics*, 11(3):333–418, 1958. doi:10.1002/cpa.3160110306.

[97] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019. doi:10.1017/jfm.2019.238.

[98] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[99] Daryl Geller and Isaac Z Pesenson. Kolmogorov and linear widths of balls in Sobolev spaces on compact manifolds. *Mathematica Scandinavica*, pages 96–122, 2014. doi:10.7146/math.scand.a-18005.

[100] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, January 2016. doi:10.1137/15m1009718. URL https://doi.org/10.1137/15m1009718.

[101] Seyede Fatemeh Ghoreishi, Samuel Friedman, and Douglas L Allaire. Adaptive Dimensionality Reduction for Fast Sequential Optimization With Gaussian Processes. *Journal of Mechanical Design*, 141(7):071404, 2019. doi:10.1115/1.4043202.

[102] Amur Ghose, Abdullah Rashwan, and Pascal Poupart. Batch norm with entropic regularization turns deterministic autoencoders into generative models. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1079–1088. PMLR, 03–06 Aug 2020. URL https://proceedings.mlr.press/v124/ghose20a.html.

[103] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[104] Edwige Godlewski and Pierre-Arnaud Raviart. *Hyperbolic systems of conservation laws*. Number 3-4. Ellipses, 1991.

[105] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. Johns Hopkins University Press, 2013. ISBN 9781421407944. doi:10.56021/9781421407944.

[106] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[107] Jonathan Goodman and Zhouping Xin. Viscous limits for piecewise smooth solutions to systems of conservation laws. *Archive for rational mechanics and analysis*, 121:235–265, 1992. doi:10.1007/BF00410614.

[108] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. doi:10.1007/s11263-021-01453-z.

[109] Pawan Goyal and Peter Benner. LQResNet: A Deep Neural Network Architecture for Learning Dynamic Processes. *arXiv preprint arXiv:2103.02249*, 2021.

[110] Constantin Greif and Karsten Urban. Decay of the Kolmogorov N-width for wave problems. *Applied Mathematics Letters*, 96:216–222, 2019. doi:10.1016/j.aml.2019.05.013.

[111] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[112] Nicola Guglielmi, María López-Fernández, and Mattia Manucci. Pseudospectral roaming contour integral methods for convection-diffusion equations. *Journal of Scientific Computing*, 89:1–31, 2021. doi:10.1007/s10915-021-01601-0.

[113] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1):014004, 2017. doi:10.1088/1361-6420/aa9a90.

[114] Stefanie Hahmann, Georges-Pierre Bonneau, Sébastien Barbier, Gershon Elber, and Hans Hagen. Volume-preserving FFD for programmable graphics hardware. *The Visual Computer*, 28(3):231–245, July 2011. doi:10.1007/s00371-011-0608-5. URL https://doi.org/10.1007/s00371-011-0608-5.

[115] Sara Hahner and Jochen Garcke. Mesh convolutional autoencoder for semi-regular meshes of different sizes. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, January 2022. doi:10.1109/wacv51458.2022.00240. URL https://doi.org/10.1109/wacv51458.2022.00240.

[116] Stefan Hain and Karsten Urban. An ultra-weak space-time variational formulation for the schr\" odinger equation. *arXiv preprint arXiv:2212.14398*, 2022.

[117] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, January 2011. doi:10.1137/090771806. URL https://doi.org/10.1137/090771806.

[118] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[119] Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. *The Morgan Kaufmann Series in Data Management Systems*, 5(4):83–124, 2012. doi:10.1016/C2009-0-61819-5. URL https://doi.org/10.1016/C2009-0-61819-5.

[120] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN. *ACM Transactions on Graphics*, 38(4):1–12, July 2019. doi:10.1145/3306346.3322959. URL https://doi.org/10.1145/3306346.3322959.

[121] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[122] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE transactions on pattern analysis and machine intelligence*, 18(6):607–616, 1996. doi:10.1109/34.506411.

[123] R Hawkins, MH Khalid, K Smetana, and J Trampert. Model order reduction for seismic waveform modelling: inspiration from normal modes. *Geophysical Journal International*, 234 (3):2255–2283, 2023. doi:10.1093/gji/ggad195.

[124] Julian Henning, Davide Palitta, Valeria Simoncini, and Karsten Urban. An ultraweak space-time variational formulation for the wave equation: Analysis and efficient numerical solution. *ESAIM: Mathematical Modelling and Numerical Analysis*, 56(4):1173–1198, 2022. doi:10.1051/m2an/2022035.

[125] Robin Herkert, Patrick Buchfink, and Bernard Haasdonk. Dictionary-based online-adaptive structure-preserving model order reduction for parametric hamiltonian systems. *arXiv preprint arXiv:2303.18072*, 2023.

[126] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016. doi:10.1007/978-3-319-22470-1.

[127] Jan S Hesthaven, Cecilia Pagliantini, and Nicolò Ripamonti. Structure-preserving model order reduction of hamiltonian systems. *arXiv preprint arXiv:2109.12367*, 2021.

[128] Raymond M Hicks and Preston A Henne. Wing Design by Numerical Optimization. *Journal of Aircraft*, 15(7):407–412, 1978. doi:10.2514/3.58379.

[129] G. Hirota, R. Maheshwari, and M.C. Lin. Fast volume-preserving free-form deformation using multi-level optimization. *Computer-Aided Design*, 32(8-9):499–512, August 2000. doi:10.1016/s0010-4485(00)00038-5. URL https://doi.org/10.1016/s0010-4485(00)00038-5.

[130] Chi Hoang, Youngsoo Choi, and Kevin Carlberg. Domain-decomposition least-squares petrov–galerkin (dd-lspg) nonlinear model reduction. *Computer methods in applied mechanics and engineering*, 384:113997, 2021. doi:10.1016/j.cma.2021.113997.

[131] Alexander Holiday, Mahdi Kooshkbaghi, Juan M Bello-Rivas, C William Gear, Antonios Zagaris, and Ioannis G Kevrekidis. Manifold learning for parameter reduction. *Journal of computational physics*, 392:419–431, 2019. doi:10.1016/j.jcp.2019.04.015.

[132] Paul Houston, John A Mackenzie, Endre Süli, and Gerald Warnecke. A posteriori error analysis for numerical approximations of Friedrichs systems. *Numerische Mathematik*, 82(3):433–470, 1999. doi:10.1007/s002110050426.

[133] Thomas JR Hughes, Michel Mallet, and Mizukami Akira. A new finite element formulation for computational fluid dynamics: II. beyond SUPG. *Computer methods in applied mechanics and engineering*, 54(3):341–355, 1986. doi:10.1016/0045-7825(86)90110-6.

[134] Dinh Bao Phuong Huynh, David J Knezevic, and Anthony T Patera. A static condensation reduced basis element method: approximation and a posteriori error estimation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(1):213–251, 2013.

[135] A. Iollo and D. Lombardi. Advection modes by optimal mass transfer. *Phys. Rev. E*, 89:022923, Feb 2014. doi:10.1103/PhysRevE.89.022923. URL https://link.aps.org/doi/10.1103/PhysRevE.89.022923.

[136] Angelo Iollo and Tommaso Taddei. Mapping of coherent structures in parameterized flows by learning optimal transportation with Gaussian models. *Journal of Computational Physics*, 471:111671, 2022. doi:10.1016/j.jcp.2022.111671.

[137] Tobin Isaac, Noemi Petra, Georg Stadler, and Omar Ghattas. Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the antarctic ice sheet. *Journal of Computational Physics*, 296:348–368, September 2015. doi:10.1016/j.jcp.2015.04.047. URL https://doi.org/10.1016/j.jcp.2015.04.047.

[138] RI Issa, B Ahmadi-Befrui, KR Beshay, and AD Gosman. Solution of the implicitly discretised reacting flow equations by operator-splitting. *Journal of computational physics*, 93(2):388–410, 1991. doi:10.1016/0021-9991(91)90191-M.

[139] Eastman N Jacobs, Kenneth E Ward, and Robert M Pinkerton. The Characteristics of 78 Related Airfoil Sections from Tests in the Variable-Density Wind Tunnel. Technical Report 430, N.A.C.A., 1933.

[140] Max Jensen. *Discontinuous Galerkin methods for Friedrichs systems with irregular solutions*. PhD thesis, Citeseer, 2004.

[141] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002. doi:10.1007/978-1-4757-1904-8_11.

[142] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[143] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, September 1998.

[144] Leonard Kaufman and Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344 of *Wiley Series in Probability and Statistics*. John Wiley & Sons, 2005. ISBN 978-0-471-73578-6. doi:10.1002/9780470316801.

[145] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000. doi:10.1093/biomet/87.1.1.

[146] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022. doi:10.1016/j.jcp.2021.110841.

[147] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[148] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013 (preprint). URL https://arxiv.org/abs/1312.6114.

[149] Robin Ben Klein and Benjamin Sanderse. Structure-preserving hyper-reduction and temporal localization for reduced order models of incompressible flows. *arXiv preprint arXiv:2304.09229*, 2023.

[150] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi:10.1073/pnas.2101784118.

[151] Boris Kramer, Alexandre Noll Marques, Benjamin Peherstorfer, Umberto Villa, and Karen Willcox. Multifidelity probability estimation via fusion of estimators. *Journal of Computational Physics*, 392:385–402, 2019. doi:10.1016/j.jcp.2019.04.071.

[152] Stanislav N Kružkov. First order quasilinear equations in several independent variables. *Mathematics of the USSR-Sbornik*, 10(2):217, 1970.

[153] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische mathematik*, 90(1):117–148, 2001. doi:10.1007/s002110100282.

[154] Remi R Lam, Olivier Zahm, Youssef M Marzouk, and Karen E Willcox. Multifidelity Dimension Reduction via Active Subspaces. *SIAM Journal on Scientific Computing*, 42(2): A929–A956, 2020. doi:10.1137/18M1214123.

[155] Toni Lassila, Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Generalized reduced basis methods and n-width estimates for the approximation of the solution manifold of parametric PDEs. In *Analysis and numerics of partial differential equations*, pages 307–329. Springer, 2013. doi:10.1007/978-88-470-2592-9_16.

[156] Jessica T Lauzon, Siu Wun Cheung, Yeonjong Shin, Youngsoo Choi, Dylan Matthew Copeland, and Kevin Huynh. S-opt: A points selection algorithm for hyper-reduction in reduced order models. *arXiv preprint arXiv:2203.16494*, 2022.

[157] Charles L Lawson and Richard J Hanson. *Solving least squares problems*. SIAM, 1995. doi:10.1137/1.9781611971217.

[158] Miguel Lázaro-Gredilla and Michalis K Titsias. Variational Heteroscedastic Gaussian Process regression. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 841–848, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[159] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[160] Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5):365–386, 2014. doi:10.1615/Int.J.UncertaintyQuantification.2014006914.

[161] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404: 108973, 2020. doi:10.1016/j.jcp.2019.108973.

[162] Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991. doi:10.2307/2290563.

[163] Wang Li, Miao Zhen, and Jiang Yaolin. Model order reduction based on galerkin kpod for partial differential equations with variable coefficients. *Journal on Numerica Methods and Computer Applications*, 42(3):226, 2021.

[164] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. doi:10.1109/TNNLS.2019.2957109.

[165] Tobias Long, Robert Barnett, Richard Jefferson-Loveday, Giovanni Stabile, and Matteo Icardi. A novel reduced-order model for advection-dominated problems based on radon-cumulative-distribution transform. *arXiv preprint arXiv:2304.14883*, 2023.

[166] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[167] Alf Emil Løvgren, Yvon Maday, and Einar M Rønquist. A reduced basis element method for the steady Stokes problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 40(3): 529–552, 2006. doi:10.1051/m2an:2006021.

[168] David J Lucia, Paul I King, and Philip S Beran. Domain decomposition for reduced-order modeling of a flow with moving shocks. *AIAA journal*, 40(11):2360–2362, 2002. doi:10.2514/2.1576.

[169] David J Lucia, Paul I King, and Philip S Beran. Reduced order modeling of a two-dimensional flow with moving shocks. *Computers & Fluids*, 32(7):917–938, 2003. doi:10.1016/S0045-7930(02)00035-X.

[170] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. In *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014. doi:10.2514/6.2014-1171.

[171] John A Mackenzie, E Süli, and Gerald Warnecke. A posteriori analysis for Petrov-Galerkin approximations of Friedrichs systems. 1995.

[172] Yvon Maday and Einar M Rønquist. A reduced-basis element method. *Journal of scientific computing*, 17:447–459, 2002. doi:10.1016/S1631-073X(02)02427-5.

[173] Yvon Maday and Einar M Ronquist. The reduced basis element method: application to a thermal fin problem. *SIAM Journal on Scientific Computing*, 26(1):240–258, 2004. doi:10.1137/S1064827502419932.

[174] Yvon Maday and Eitan Tadmor. Analysis of the spectral vanishing viscosity method for periodic conservation laws. *SIAM Journal on Numerical Analysis*, 26(4):854–870, 1989. doi:10.1137/0726047.

[175] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2015 (preprint). URL https://arxiv.org/abs/1511.05644.

[176] Francesco E. Maranzana. On the Location of Supply Points to Minimize Transport Costs. *Journal of the Operational Research Society*, 15(3):261–270, 1964. doi:10.1057/jors.1964.47.

[177] Luis F Gutiérrez Marcantoni, José P Tamagno, and Sergio A Elaskar. High speed flow simulation using openfoam. *Mecánica Computacional*, 31(16):2939–2959, 2012.

[178] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics*, 401:109020, 2020. doi:10.1016/j.jcp.2019.109020.

[179] Florianr Menter. Zonal two equation kw turbulence models for aerodynamic flows. In *23rd fluid dynamics, plasmadynamics, and lasers conference*, page 2906, 1993.

[180] Markus Mrosek, Carsten Othmer, and Rolf Radespiel. Reduced-order modeling of vehicle aerodynamics via proper orthogonal decomposition. *SAE International Journal of Passenger Cars — Mechanical Systems*, 12:225–236, 10 2019. doi:10.4271/06-12-03-0016.

[181] Nikolaj T Mücke, Sander M Bohté, and Cornelis W Oosterlee. Reduced Order Modeling for Parameterized Time-Dependent PDEs using Spatially and Memory Aware Deep Learning. *Journal of Computational Science*, page 101408, 2021. doi:10.1016/j.jocs.2021.101408.

[182] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. ISBN 9780262018029.

[183] Hai Nguyen, Jonathan Wittmer, and Tan Bui-Thanh. DIAS: A Data-Informed Active Subspace Regularization Framework for Inverse Problems. *Computation*, 10(3):38, 2022. doi:10.3390/computation10030038.

[184] Fabio Nobile and Davide Pradovera. Non-intrusive double-greedy parametric model reduction by interpolation of frequency-domain rational surrogates. *ESAIM: Mathematical Modelling and Numerical Analysis*, 55(5):1895–1920, 2021. doi:10.1051/m2an/2021040.

[185] Mario Ohlberger and Stephan Rave. Reduced basis methods: Success, limitations and future challenges. *arXiv preprint arXiv:1511.02021*, 2015.

[186] Thomas O'Leary-Roseberry, Umberto Villa, Peng Chen, and Omar Ghattas. Derivative-informed projected neural networks for high-dimensional parametric maps governed by pdes. *Computer Methods in Applied Mechanics and Engineering*, 388:114199, 2022. doi:10.1016/j.cma.2021.114199.

[187] OA Oleinik. Discontinuous solutions of nonlinear differential equations. *Amer. Math. Soc. Transl*, 26(2):95–172, 1963.

[188] Carsten Othmer, Trent W Lukaczyk, Paul Constantine, and Juan J Alonso. On active subspaces in car aerodynamics. In *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4294, 2016. doi:10.2514/6.2016-4294.

[189] Guglielmo Padula, Francesco Romor, Giovanni Stabile, and Gianluigi Rozza. Generative models for the deformation of industrial shapes with linear geometric constraints: model order and parameter space reductions, 2023.

[190] Andrei Paleyes, Mark Pullin, Maren Mahsereci, Neil Lawrence, and Javier González. Emulation of physical processes with Emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*, 2019.

[191] Davide Papapicco, Nicola Demo, Michele Girfoglio, Giovanni Stabile, and Gianluigi Rozza. The neural network shifted-proper orthogonal decomposition: A machine learning approach for non-linear reduction of hyperbolic equations, March 2022. URL https://doi.org/10.1016/j.cma.2022.114687.

[192] Mario Teixeira Parente, Jonas Wallin, Barbara Wohlmuth, et al. Generalized bounds for active subspaces. *Electronic Journal of Statistics*, 14(1):917–943, 2020. doi:10.1214/20-EJS1684.

[193] Eric J Parish and Francesco Rizzi. On the impact of dimensionally-consistent and physics-based inner products for pod-galerkin and least-squares model reduction of compressible flows. *arXiv preprint arXiv:2203.16492*, 2022. doi:10.1016/j.jcp.2023.112387.

[194] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009. doi:10.1016/j.eswa.2008.01.039.

[195] Matthew D Parno and Youssef M Marzouk. Transport map accelerated markov chain monte carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018. doi:10.1137/17M1134640.

[196] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., -, 2019.

[197] Suhas V Patankar and D Brian Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*, pages 54–73. Elsevier, 1983. doi:10.1016/0017-9310(72)90054-3.

[198] Benjamin Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020. doi:10.1137/19M1257275.

[199] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016. doi:10.1016/j.cma.2016.03.025.

[200] Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A*, 473(2198):20160751, 2017. doi:10.1098/rspa.2016.0751.

[201] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning Mesh-Based Simulation with Graph Networks. *arXiv preprint arXiv:2010.03409*, 2020.

[202] Federico Pichi, Francesco Ballarin, Gianluigi Rozza, and Jan S. Hesthaven. An artificial neural network approach to bifurcating phenomena in computational fluid dynamics, 2021.

[203] Federico Pichi, Beatriz Moya, and Jan S Hesthaven. A graph convolutional autoencoder approach to model order reduction for parametrized PDEs. *arXiv preprint arXiv:2305.08573*, 2023.

[204] Allan Pinkus. *Ridge functions*, volume 205. Cambridge University Press, 2015. doi:10.1017/CBO9781316408124.

[205] Dmitry Pozharskiy, Noah J. Wichrowski, Andrew B. Duncan, Grigorios A. Pavliotis, and Ioannis G. Kevrekidis. Manifold learning for accelerating coarse-grained optimization. *Journal of Computational Dynamics*, 7(2):511–536, 2020. ISSN 2158-2491. doi:10.3934/jcd.2020021.

[206] Christophe Prud'Homme, Dimitrios V Rovas, Karen Veroy, and Anthony T Patera. A mathematical and computational framework for reliable real-time solution of parametrized partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 36(5):747–771, 2002. doi:10.1051/m2an:2002035.

[207] Christophe Prud'Homme, Dimitrios V Rovas, Karen Veroy, Luc Machiels, Yvon Maday, Anthony T Patera, and Gabriel Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *J. Fluids Eng.*, 124(1):70–80, 2002. doi:10.1115/1.1448332.

[208] Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020. doi:10.1016/j.physd.2020.132401.

[209] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Matematica numerica*. Springer Science & Business Media, 2010. doi:10.1007/978-88-470-5644-2.

[210] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015. doi:10.1007/978-3-319-15431-2.

[211] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746, 2017. doi:10.1016/j.jcp.2017.01.060.

[212] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi:10.1016/j.jcp.2018.10.045.

[213] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3d faces using convolutional mesh autoencoders. In *Computer Vision – ECCV 2018*, pages 725–741. Springer International Publishing, 2018. doi:10.1007/978-3-030-01219-9_43. URL https://doi.org/10.1007/978-3-030-01219-9_43.

[214] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003. doi:10.1007/978-3-540-28650-9_4.

[215] Jeffrey Rauch. Symmetric positive systems with boundary characteristic of constant multiplicity. *Transactions of the American Mathematical Society*, 291(1):167–187, 1985. doi:10.1090/S0002-9947-1985-0797053-4.

[216] Jeffrey Rauch. Boundary value problems with nonuniformal characteristic boundary. *Journal de mathématiques pures et appliquées*, 73(4):347–353, 1994.

[217] Donsub Rim and Kyle T. Mandli. Displacement interpolation using monotone rearrangement. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1503–1531, 2018. doi:10.1137/18M1168315. URL https://doi.org/10.1137/18M1168315.

[218] Francesco Romor, Marco Tezzele, and Gianluigi Rozza. ATHENA: Advanced Techniques for High dimensional parameter spaces to Enhance Numerical Analysis. *arXiv preprint arXiv:2105.06713*, 2020. doi:10.1016/j.simpa.2021.100133.

[219] Francesco Romor, Marco Tezzele, and Gianluigi Rozza. Multi-fidelity data fusion for the approximation of scalar functions with low intrinsic dimensionality through active subspaces. In *Proceedings in Applied Mathematics & Mechanics*, volume 20. Wiley Online Library, 2021. doi:10.1002/pamm.202000349.

[220] Francesco Romor, Marco Tezzele, and Gianluigi Rozza. A local approach to parameter space reduction for regression and classification tasks. *arXiv preprint arXiv:2107.10867*, 2021.

[221] Francesco Romor, Marco Tezzele, Andrea Lario, and Gianluigi Rozza. Kernel-based active subspaces with application to computational fluid dynamics parametric problems using discontinuous Galerkin method. *International Journal for Numerical Methods in Engineering*, 123 (23):6000–6027, 2022. doi:10.1002/nme.7099.

[222] Francesco Romor, Giovanni Stabile, and Gianluigi Rozza. Explicable hyper-reduced order models on nonlinearly approximated solution manifolds of compressible and incompressible navier-stokes equations, 2023.

[223] Francesco Romor, Giovanni Stabile, and Gianluigi Rozza. Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. *Journal of Scientific Computing*, 94(3):74, 2023. doi:10.1007/s10915-023-02128-2.

[224] Francesco Romor, Davide Torlo, and Gianluigi Rozza. Friedrichs' systems discretized with the discontinuous galerkin method: domain decomposable model order reduction and graph neural networks approximating vanishing viscosity solutions, 2023.

[225] Francesco Romor, Marco Tezzele, Markus Mrosek, Carsten Othmer, and Gianluigi Rozza. Multi-fidelity data fusion through parameter space reduction with applications to automotive engineering. *arXiv preprint arXiv:2110.14396*, Submitted, 2021.

[226] Gianluigi Rozza, M Haris Malik, Nicola Demo, Marco Tezzele, Michele Girfoglio, Giovanni Stabile, and Andrea Mola. Advances in Reduced Order Methods for Parametric Industrial Problems in Computational Fluid Dynamics. In Roger Owen, René de Borst, Jason Reese, and Pearce Chris, editors, *ECCOMAS ECFD 7 - Proceedings of 6th European Conference on Computational Mechanics (ECCM 6) and 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, pages 59–76, Glasgow, UK, 2018.

[227] Gianluigi Rozza, Martin Hess, Giovanni Stabile, Marco Tezzele, and Francesco Ballarin. Basic Ideas and Tools for Projection-Based Model Reduction of Parametric Partial Differential Equations. In Peter Benner, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, Wilhelmus H. A. Schilders, and Luis Miguel Silveira, editors, *Model Order Reduction*, volume 2, chapter 1, pages 1–47. De Gruyter, Berlin, Boston, 2020. ISBN 9783110671490. doi:10.1515/9783110671490-001.

[228] Gianluigi Rozza, Giovanni Stabile, and Francesco Ballarin. *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics*. SIAM, 2022. ISBN 978-1-611977-24-0. doi:10.1137/1.9781611977257.

[229] Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms. In *International conference on similarity search and applications*, pages 171–187. Springer, 2019. doi:10.1007/978-3-030-32047-8_16.

[230] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86*. ACM Press, 1986. doi:10.1145/15922.15903. URL https://doi.org/10.1145/15922.15903.

[231] Yeonjong Shin and Dongbin Xiu. Nonadaptive quasi-optimal points selection for least squares linear regression. *SIAM Journal on Scientific Computing*, 38(1):A385–A411, 2016. doi:10.1137/15M1015868.

[232] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. doi:10.1109/CVPR.2017.11.

[233] Bart Smets, Jim Portegies, Erik Bekkers, and Remco Duits. PDE-based group equivariant convolutional neural networks. *arXiv preprint arXiv:2001.09046*, 2020. doi:10.1007/s10851-022-01114-x.

[234] Edward Snelson, Zoubin Ghahramani, and Carl E Rasmussen. Warped Gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 337–344, -, 2004. MIT Press.

[235] Thomas Sonar and Endre Süli. A dual graph-norm refinement indicator for finite volume approximations of the Euler equations. *Numerische Mathematik*, 78(4):619–658, 1998. doi:10.1007/s002110050328.

[236] Philippe Spalart and Steven Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992. doi:10.2514/6.1992-439.

[237] Giovanni Stabile and Gianluigi Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations. *Computers & Fluids*, 2018. doi:10.1016/j.compfluid.2018.01.035.

[238] Giovanni Stabile, Saddam Hijazi, Andrea Mola, Stefano Lorenzi, and Gianluigi Rozza. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. *Communications in Applied and Industrial Mathematics*, 8(1): 210–236, (2017). doi:10.1515/caim-2017-0011.

[239] Giovanni Stabile, Matteo Zancanaro, and Gianluigi Rozza. Efficient geometrical parametrization for finite-volume-based reduced order methods. *International Journal for Numerical Methods in Engineering*, 121(12):2655–2682, 2020. doi:10.1002/nme.6324.

[240] Masashi Sugiyama. Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. *Journal of machine learning research*, 8(5), 2007.

[241] Timothy John Sullivan. *Introduction to Uncertainty Quantification*, volume 63. Springer, 2015. doi:10.1007/978-3-319-23395-6.

[242] Tommaso Taddei. A registration method for model order reduction: Data compression and geometry reduction. *SIAM Journal on Scientific Computing*, 42(2):A997–A1027, 2020. doi:10.1137/19M1271270. URL https://doi.org/10.1137/19M1271270.

[243] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. doi:10.1109/cvpr.2018.00612. URL https://doi.org/10.1109/cvpr.2018.00612.

[244] Vladimir Nikolaevich Temlyakov. Nonlinear Kolmogorov widths. *Mathematical Notes*, 63(6): 785–795, 1998. doi:10.1007/BF02312773.

[245] John Tencer and Kevin Potter. A tailored convolutional neural network for nonlinear manifold learning of computational physics data using unstructured spatial discretizations. *SIAM Journal on Scientific Computing*, 43(4):A2581–A2613, 2021. doi:10.1137/20M1344263.

[246] Ayush Tewari, Mallikarjun B R, Xingang Pan, Ohad Fried, Maneesh Agrawala, and Christian Theobalt. Disentangled3d: Learning a 3d generative model with disentangled geometry and appearance from monocular images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. doi:10.1109/cvpr52688.2022.00157. URL https://doi.org/10.1109/cvpr52688.2022.00157.

[247] M. Tezzele, N. Demo, A. Mola, and G. Rozza. An integrated data-driven computational pipeline with model order reduction for industrial and applied mathematics. *Special Volume ECMI, In Press*, 2020. doi:10.1007/978-3-030-96173-2_7.

[248] Marco Tezzele, Francesco Ballarin, and Gianluigi Rozza. Combined parameter and model reduction of cardiovascular problems by means of active subspaces and POD-Galerkin methods. In Daniele Boffi, Luca F. Pavarino, Gianluigi Rozza, Simone Scacchi, and Christian Vergara, editors, *Mathematical and Numerical Modeling of the Cardiovascular System and Applications*, volume 16 of *SEMA-SIMAI Series*, pages 185–207. Springer International Publishing, 2018. doi:10.1007/978-3-319-96649-6_8.

[249] Marco Tezzele, Filippo Salmoiraghi, Andrea Mola, and Gianluigi Rozza. Dimension reduction in heterogeneous parametric spaces with application to naval engineering shape design problems. *Advanced Modeling and Simulation in Engineering Sciences*, 5(1):25, Sep 2018. ISSN 2213-7467. doi:10.1186/s40323-018-0118-3. URL https://doi.org/10.1186/s40323-018-0118-3.

[250] Marco Tezzele, Nicola Demo, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. Enhancing CFD predictions in shape design problems by model and parameter space reduction. *Advanced Modeling and Simulation in Engineering Sciences*, 7(40), 2020. doi:10.1186/s40323-020-00177-y.

[251] Marco Tezzele, Nicola Demo, Andrea Mola, and Gianluigi Rozza. Pygem: Python geometrical morphing. *Software impacts*, 7:100047, 2021. doi:10.1016/j.simpa.2020.100047.

[252] Marco Tezzele, Lorenzo Fabris, Matteo Sidari, Mauro Sicchiero, and Gianluigi Rozza. A multi-fidelity approach coupling parameter space reduction and non-intrusive POD with application to structural optimization of passenger ship hulls. *International Journal for Numerical Methods in Engineering*, 124(5):1193–1210, 2023. doi:10.1002/nme.7159.

[253] Jakub M. Tomczak. *Deep Generative Modeling*. Springer International Publishing, 2022. doi:10.1007/978-3-030-93158-2. URL https://doi.org/10.1007/978-3-030-93158-2.

[254] Davide Torlo. Model Reduction for Advection Dominated Hyperbolic Problems in an ALE Framework: Offline and Online Phases. *arXiv e-prints*, art. arXiv:2003.13735, March 2020.

[255] Davide Torlo and Mario Ricchiuto. Model order reduction strategies for weakly dispersive waves. *Mathematics and Computers in Simulation*, 205:997–1028, 2023. ISSN 0378-4754. doi:10.1016/j.matcom.2022.10.034.

[256] Joel A Tropp. User-Friendly Tail Bounds for Sums of Random Matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012. doi:10.1007/s10208-011-9099-z.

[257] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. ACM Press, 1994. doi:10.1145/192161.192241. URL https://doi.org/10.1145/192161.192241.

[258] Wayne Isaac Tan Uy, Dirk Hartmann, and Benjamin Peherstorfer. Operator inference with roll outs for learning reduced models from scarce and low-quality data. *Computers & Mathematics with Applications*, 145:224–239, 2023. doi:10.1016/j.camwa.2023.06.012.

[259] Arjan Van Der Schaft, Dimitri Jeltsema, et al. Port-hamiltonian systems theory: An introductory overview. *Foundations and Trends® in Systems and Control*, 1(2-3):173–378, 2014. doi:10.1561/2600000002.

[260] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[261] Ricardo Vinuesa and Steven L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, June 2022. doi:10.1038/s43588-022-00264-7. URL https://doi.org/10.1038/s43588-022-00264-7.

[262] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *ACM Transactions on Graphics*, 25(3):1118–1125, July 2006. doi:10.1145/1141911.1142002. URL https://doi.org/10.1145/1141911.1142002.

[263] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. In *Computer Vision ECCV 2018*, pages 55–71. Springer International Publishing, 2018. doi:10.1007/978-3-030-01252-6_4. URL https://doi.org/10.1007/978-3-030-01252-6_4.

[264] Henry G Weller, Gavin Tabor, Hrvoje Jasak, and Christer Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998. doi:10.1063/1.168744.

[265] David Wells, Zhu Wang, Xuping Xie, and Traian Iliescu. An evolve-then-filter regularized reduced order model for convection-dominated flows. *International Journal for Numerical Methods in Fluids*, 84(10):598–615, 2017. doi:10.1002/fld.4363.

[266] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, October 2019. doi:10.1109/iccv.2019.00113. URL https://doi.org/10.1109/iccv.2019.00113.

[267] Yuxiao Wen, Eric Vanden-Eijnden, and Benjamin Peherstorfer. Coupling parameter and particle dynamics for adaptive sampling in neural galerkin schemes. *arXiv preprint arXiv:2306.15630*, 2023.

[268] Paul F White, Bradford G Knight, Grzegorz P Filip, and Kevin J Maki. Numerical simulations of the duisburg test case hull maneuvering in waves. In *SNAME Maritime Convention*, page D033S001R004. SNAME, 2019.

[269] David C Wilcox et al. *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA, 1998.

[270] SM Wild. Solving derivative-free nonlinear least squares problems with pounders. 2014. *Argonne National Lab*.

[271] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT press Cambridge, MA, 2006. doi:10.1007/978-3-540-28650-9_4.

[272] Qiang Wu, Feng Liang, and Sayan Mukherjee. Localized sliced inverse regression. *Journal of Computational and Graphical Statistics*, 19(4):843–860, 2010. doi:10.1198/jcgs.2010.08080.

[273] Nathan Wycoff, Mickaël Binois, and Stefan M Wild. Sequential Learning of Active Subspaces. *Journal of Computational and Graphical Statistics*, pages 1–14, 2021. doi:10.1080/10618600.2021.1874962.

[274] D Xiao, CE Heaney, F Fang, L Mottet, R Hu, DA Bistrian, E Aristodemou, IM Navon, and CC Pain. A domain decomposition non-intrusive reduced order model for turbulent flows. *Computers & Fluids*, 182:15–27, 2019. doi:10.1016/j.compfluid.2019.02.012.

[275] Dunhui Xiao, Fangxin Fang, Claire E Heaney, IM Navon, and CC Pain. A domain decomposition method for the non-intrusive reduced order modelling of fluid flow. *Computer Methods in Applied Mechanics and Engineering*, 354:307–330, 2019. doi:10.1016/j.cma.2019.05.039.

[276] Xuping Xie, David Wells, Zhu Wang, and Traian Iliescu. Numerical analysis of the leray reduced order model. *Journal of Computational and Applied Mathematics*, 328:12–29, 2018. doi:10.1016/j.cam.2017.06.026.

[277] Junda Xiong, Xin Cai, and Jinglai Li. Clustered active-subspace based local gaussian process emulator for high-dimensional and complex computer models. *arXiv preprint arXiv:2101.00057*, 2020. doi:10.1016/j.jcp.2021.110840.

[278] Jiayang Xu and Karthik Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020. doi:10.1016/j.cma.2020.113379.

[279] Yu-Jie Yuan, Yu-Kun Lai, Jie Yang, Qi Duan, Hongbo Fu, and Lin Gao. Mesh variational autoencoders with edge contraction pooling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2020. doi:10.1109/cvprw50498.2020.00145. URL https://doi.org/10.1109/cvprw50498.2020.00145.

[280] Olivier Zahm, Tiangang Cui, Kody Law, Alessio Spantini, and Youssef Marzouk. Certified dimension reduction in nonlinear Bayesian inverse problems. *arXiv preprint arXiv:1807.03712*, 2018. doi:10.1090/mcom/3737.

[281] Olivier Zahm, Paul G Constantine, Clementine Prieur, and Youssef M Marzouk. Gradient-based dimension reduction of multivariate vector-valued functions. *SIAM Journal on Scientific Computing*, 42(1):A534–A558, 2020. doi:10.1137/18M1221837.

[282] Matteo Zancanaro, Markus Mrosek, Giovanni Stabile, Carsten Othmer, and Gianluigi Rozza. Hybrid neural network reduced order modelling for turbulent flows with geometric parameters. *Fluids*, 6(8):296, 2021. doi:10.3390/fluids6080296.

[283] Guannan Zhang, Jiaxin Zhang, and Jacob Hinkle. Learning nonlinear level sets for dimensionality reduction in function approximation. In *Advances in Neural Information Processing Systems*, pages 13199–13208, 2019.

[284] Ralf Zimmermann. Manifold interpolation. *System-and Data-Driven Methods and Algorithms, P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and LM Silveira, eds., Model Order Reduction*, 1:229–274, 2021. doi:10.1515/9783110498967-007.