Advances in Computational Mathematics (2026) 52:31
https://doi.org/10.1007/s10444-026-10295-7

Deep orthogonal decomposition: a continuously adaptive neural network approach to model order reduction of parametrized partial differential equations

Nicola Rares Franco[1] · **Andrea Manzoni**[1] · **Paolo Zunino**[1] · **Jan S. Hesthaven**[2]

Received: 11 February 2025 / Accepted: 13 February 2026

Abstract
We develop a novel deep learning technique, termed deep orthogonal decomposition (DOD), for dimensionality reduction and reduced order modeling of parameter-dependent partial differential equations. The approach involves constructing a deep neural network model that approximates the solution manifold using a continuously adaptive local basis. In contrast to global methods, such as proper orthogonal decomposition (POD), this adaptivity allows the DOD to mitigate the Kolmogorov barrier when dealing with space-interacting parameters, making the approach applicable to a wide spectrum of parametric problems. Leveraging this idea, we use the DOD to construct an adaptive alternative to the so-called POD-NN method, here termed DOD-NN. The approach is fully data-driven and non-intrusive but, at the same time, allows for a tight control on error propagation and remains highly interpretable thanks to the rich structure present in the latent space. For this reason, the proposed approach stands out as a valuable alternative to other nonlinear model order reduction techniques, such as those based on deep autoencoders. The methodology is discussed both theoretically and practically, evaluating its performances on problems involving nonlinear PDEs, parametrized geometries, and high-dimensional parameter spaces. Finally, we conclude with a brief discussion on potential applications of the DOD beyond DOD-NN, featuring, for instance, the integration of our approach within intrusive reduced order models such as the reduced basis method.

Keywords Reduced order modeling · Parametrized PDEs · Adaptive methods · Neural networks

Mathematics Subject Classification (2010) 65N99 · 35B30 · 68T07

Communicated by: Bernard Haasdonk

Extended author information available on the last page of the article

Published online: 17 April 2026

1 Introduction

Reduced order modeling aims at creating effective model surrogates, known as reduced order models (ROMs), capable of mimicking the precision of traditional methods in price of a lower computational cost. Usually, these ROMs achieve their proficiency by learning from high-quality simulations obtained by means of classical numerical solvers, commonly referred to as full order models (FOMs), extracting essential information for replicating the behavior of complex systems. ROMs are particularly useful in the context of parameter-dependent PDEs, applications involving inverse problems [1–5], optimal control [6–9], or uncertainty quantification [10–13], which can easily become prohibitive due to prolonged processing times and unbearable computational demands. As a remedy, reduced order modeling techniques [14, 15] are now gaining the attention of many scientists and researchers.

Most ROMs are based on a common foundation, which is to rely on *dimensionality reduction* techniques. The latter consists of suitable algorithms capable of compressing and reconstructing high-fidelity simulations by mapping them to a small feature space, also known as the *latent space*. To this end, researchers can rely on a large pletora of different approaches, from linear techniques based on orthogonal projections, such as proper orthogonal decomposition (POD) [15], and other basis expansions, such as DeepONets [16], to fully nonlinear strategies, employing, for example, wavelet transforms [17] and deep autoencoders [18]. In practice, the choice between one approach or the other is typically problem-specific: in some cases, linear projection techniques can provide a high-level of accuracy at a high compression rate, as in the case of diffusion processes [19]; in other situations, instead, nonlinear techniques are more favorable as, although returning much complex representations, they can effectively capture nontrivial features such as sharp edges, moving fronts, and singularities [20, 21]. Mathematically speaking, this distinction is perfectly represented by the concept of Kolmogorov n-width. The latter quantifies to which extent the solution manifold $\mathscr{S}$ can be approximated in terms of linear subspaces of dimension n: if the Kolmogorov n-width, $d_n(\mathscr{S})$, decays rapidly, then linear methods are to be favored; conversely, if the decay is slow, then nonlinear approaches may provide an appealing alternative [22].

In this work, we would like to focus on a specific scenario that, compared to the previous ones, is arguably somewhere in between. To illustrate the idea, consider the case of a flow around an obstacle whose dynamics is described by the steady incompressible Navier–Stokes equations. The problem may depend on several parameters: some of them, which we collect in the parameter vector $\boldsymbol{\mu}$, may determine the position/orientation of the obstacle, while others, denoted as $\boldsymbol{\nu}$, might be related to physical quantities, such as, for example, the fluid density and/or the inflow condition. As testified by multiple works in the ROM literature, especially in the low Reynolds regime, where the dynamics can be approximated by the Stokes equations (see, e.g., [23, 24]), linear methods can be extremely effective in capturing the variability of the fluid flow for a fixed geometric configuration $\boldsymbol{\mu}_0$ but varying physical parameters $\boldsymbol{\nu}$. However, if we allow the obstacle location to change, that is, if we let $\boldsymbol{\mu}$ vary within a suitable parameter space, then the situation becomes notably different: linear methods begin to struggle, and the Kolmogorov n-width starts to deteriorate, reflecting the typical

behavior of parametric PDEs with space-interacting parameters [22, 25]. This is a prototypical example of a situation in which: (i) $d_n(\mathscr{S})$ decays slowly, but (ii) the solution manifold admits a decomposition into suitable submanifolds, $\mathscr{S}_{\boldsymbol{\mu}}$, with fast decaying Kolmogorov n-widths. In particular, these submanifolds can be obtained by fixing the values of the "bad" parameters, $\boldsymbol{\mu}$, while leaving the others, $\boldsymbol{\nu}$, free to change.

To address these difficulties, the ROM literature provides several alternatives, which, to our knowledge, can be summarized as falling into one of the following categories: dictionary-based approaches [26–30], basis interpolation methods [31–33], and time-adaptive techniques [34–36]. Here, given our focus on stationary PDEs, we shall limit our discussion to the first two classes. Particularly, our attention will be limited to *offline* adaptive approaches, that is, ROMs relying on a family of local bases that are learned during training; *online* adaptive schemes, in contrast, implement suitable strategies that allow ROMs to flexibly adapt as new parametric configurations are encountered, or as time flows. Thus, one of their main purposes is to anticipate unseen behavior. We refer the interested reader to [37, 38].

Alongside dictionary-based ROMs and basis interpolation methods, some researchers are now exploring the interplay between adaptive basis methods and deep learning algorithms (see, e.g., [39, 40]). Rather than replacing existing techniques, these efforts stem from the fact that no universal nor comprehensive method has been developed yet, which drives the exploration of new ideas. Our purpose for this work is to expand upon this emerging trend by presenting a novel approach to model order reduction that can exploit the intrinsic regularity of certain problem classes by leveraging a *continuously adaptive* linear subspace. Specifically, our idea is to construct a deep neural network architecture that, to each parameter vector $\boldsymbol{\mu}$, associates a corresponding linear subspace $\mathscr{V}_{\boldsymbol{\mu}}$ approximating the submanifold $\mathscr{S}_{\boldsymbol{\mu}} \subset \mathscr{S}$. Our proposal can be thought as a continuous generalization of localized POD algorithms, or, alternatively, it can also be viewed as an adaptive version of DeepONet. Given the use of neural networks and the intimate connection with classical POD, we term our approach deep orthogonal decomposition (DOD)[1].

The paper is organized as follows. First, in Section 2, we start by setting some notation, introducing the problem of interest, the underlying assumptions, and their consequences. Then, in Section 3, we introduce the DOD algorithm, discussing the whole idea from the sole perspective of dimensionality reduction; the use of DOD for reduced order modeling, namely, the DOD-NN method, instead, is deferred to Section 5. In both cases, we assess the proposed approaches through numerical experiments, comparing their performance against other well-established techniques. These results are reported in Sections 4 and 6, respectively. Lastly, we devote Section 7 to a

[1] It has come to our attention that the term "Deep Orthogonal Decomposition" is not entirely new: in fact, the same terminology can be found in [41], an unpublished work by Daniel J. Tait (2020). However, the two approaches are entirely different, as they pursue fundamentally different goals: ours focuses on parametrized stationary PDEs leveraging a suitable decoupling of the parameter space for constructing a continuously adaptive basis; [41], instead, deals with (unparametrized) time-dependent PDEs, leveraging neural networks for the construction of a memory-aware time-adaptive local basis. In addition to this, the two approaches differ in terms of neural network architectures, training routines, and online computations. However, given that both works share the common idea of exploiting deep learning to build an adaptive local basis, we insist on using the same terminology.

concluding discussion, where we underscore both the strengths and limitations of the proposed approach, offering additional information on possible future developments.

2 Problem setup

We start by fixing some notation. Let $(V_h, \|\cdot\|_{V_h})$ be a finite-dimensional Hilbert state space, $V_h \cong \mathbb{R}^{N_h}$, arising, for instance, from a suitable finite element discretization of a given stationary PDE, so that $V_h \subset L^2(\Omega)$ for some spatial domain $\Omega \subset \mathbb{R}^d$. Let $\{\varphi_i\}_{i=1}^{N_h}$ be a basis for V_h. Given $u \in V_h$, we write

$$\mathbf{u} := [\mathrm{u}^{(1)}, \ldots, \mathrm{u}^{(N_h)}]^\top$$

for the corresponding vector of degrees of freedom (dof), that is, the set of basis coefficients that produce the representation of u in terms of $\varphi_1, \ldots, \varphi_{N_h}$, in the sense that

$$u(\mathbf{x}) = \sum_{i=1}^{N_h} \mathrm{u}^{(i)} \varphi_i(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega. \tag{1}$$

Let $\langle\cdot,\cdot\rangle_{V_h}$ be the inner product associated to $\|\cdot\|_{V_h}$. We define the Gram matrix $\mathbb{G} \in \mathbb{R}^{N_h \times N_h}$ as the symmetric positive definite matrix given by

$$\mathbb{G} := \begin{bmatrix} \langle\varphi_1,\ \varphi_1\rangle_{V_h} & \ldots & \langle\varphi_1,\ \varphi_{N_h}\rangle_{V_h} \\ \ldots & \ldots & \ldots \\ \langle\varphi_{N_h}, \varphi_1\rangle_{V_h} & \ldots & \langle\varphi_{N_h}, \varphi_{N_h}\rangle_{V_h} \end{bmatrix}. \tag{2}$$

When $\|\cdot\|_{V_h} = \|\cdot\|_{L^2}$ is the L^2 norm, the latter is commonly referred to as the *mass* matrix. The Gram matrix allows us to equip $\mathbb{R}^{N_h}$ with the following norm

$$\|\mathbf{u}\| := \sqrt{\mathbf{u}^\top \mathbb{G} \mathbf{u}},$$

which is nothing but the discrete equivalent of $\|\cdot\|_{V_h}$. In fact, it is easy to see that $\|\mathbf{u}\| = \|u\|_{V_h}$ whenever $\mathbf{u}$ is the dof representation of u. Note that this norm can differ substantially from the Euclidean norm $|\mathbf{u}| := \sqrt{\mathbf{u}^\top \mathbf{u}}$. In particular, the two coincide if and only if $\varphi_1, \ldots, \varphi_{N_h}$ are orthonormal with respect to $\langle\cdot,\cdot\rangle_{V_h}$, in which case $\mathbb{G} = \mathbb{I}$ is the identity matrix.

With this setup, let us now introduce the parametric problem. Let $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\nu} \in \mathbb{R}^{p'}$ be two vectors of parameters. Ideally, we collect in $\boldsymbol{\mu}$ all parameters that have a geometric or space-varying nature; all the remaining parameters, which typically concern the physical properties of the model at hand, are collected in $\boldsymbol{\nu}$. We allow both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ to vary within a suitable parameter space, herein assumed to be compact: we shall write $\boldsymbol{\mu} \in \Theta$ and $\boldsymbol{\nu} \in \Theta'$. We consider a parametrized problem of the form: given $(\boldsymbol{\mu}, \boldsymbol{\nu}) \in \Theta \times \Theta'$ find $u \in V_h$ such that

$$\mathscr{R}(\boldsymbol{\mu}, \boldsymbol{\nu}, u) = 0, \tag{3}$$

where $\mathscr{R} : \Theta \times \Theta' \times V_h \to \mathbb{R}$ is a given parameter-dependent nonlinear operator, inclusive of external quantities such as, e.g., boundary conditions or source terms. We think of (3) as a discretized parameter-dependent stationary PDE, formulated in a weak or strong form, with $\mathscr{R}$ representing the norm of the PDE residual.

By leveraging the dof representation in (1), problem (3) naturally defines a map from $\Theta \times \Theta' \to \mathbb{R}^{N_h}$ which maps every parameter combination onto the basis coefficients of the corresponding PDE solution, namely

$$\mathcal{F} : (\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} := [\mathrm{u}^{(1)}_{\boldsymbol{\mu},\boldsymbol{\nu}}, \dots, \mathrm{u}^{(N_h)}_{\boldsymbol{\mu},\boldsymbol{\nu}}]$$

such that $u_{\boldsymbol{\mu},\boldsymbol{\nu}} := \sum_{i=1}^{N_h} \mathrm{u}^{(i)}_{\boldsymbol{\mu},\boldsymbol{\nu}} \varphi_i$ solves (3). Our purpose is to provide an efficient approximation of this parameter-to-solution operator $\mathcal{F}$, so that we can avoid repeated calls to the PDE solver when a large number of evaluations are required.

In general, changes in model parameters may produce different effects on the PDE solution. Here, we address a specific scenario, which is easily explained via the concept of *Kolmogorov n-width*. Given a set $S \subset \mathbb{R}^{N_h}$, we define its Kolmogorov n-width $d_n(S)$ as the error achieved by its "best approximation" in terms of linear subspaces of dimension n, that is,

$$d_n(S) := \inf_{\mathbb{V} \in \mathbb{R}^{N_h \times n}} \sup_{\mathbf{u} \in S} \|\mathbf{u} - \mathbb{V}\mathbb{V}^\top \mathbb{G}\mathbf{u}\|. \tag{4}$$

Note in fact that if $\mathbb{V}$ is $\mathbb{G}$-orthonormal, meaning $\mathbb{V}^\top \mathbb{G}\mathbb{V}$ is the $n \times n$ identity matrix, then $\mathbb{V}\mathbb{V}^\top \mathbb{G}\mathbf{u}$ is the projection of $\mathbf{u}$ onto span($\mathbb{V}$). In this work, we focus our attention on those cases in which:

A1. The solution manifold $\mathscr{S} := \{\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \mid \boldsymbol{\mu} \in \Theta, \boldsymbol{\nu} \in \Theta'\} \subset V_h$ exhibits a slow decay of the Kolmogorov n-width, e.g.,

$$d_n(\mathscr{S}) \leq C n^{-\alpha}$$

for some $C > 0$ and $\alpha \in (0, 1)$.

A2. The geometrical/space-varying parameters are the main cause to (A.1), in the sense that the submanifolds $\mathscr{S}_{\boldsymbol{\mu}} := \{\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \mid \boldsymbol{\nu} \in \Theta'\} \subset \mathscr{S}$, corresponding to $\boldsymbol{\mu}$-slices of $\mathscr{S}$, have either a uniformly fast decay,

$$\sup_{\boldsymbol{\mu} \in \Theta} d_n(\mathscr{S}_{\boldsymbol{\mu}}) \leq C' n^{-\beta},$$

with $C' > 0$ and $\beta \geq 1$, or present much smaller n-widths, e.g., $\beta \approx \alpha$ but $C' \ll C$.

This scenario can be extremely common in parametrized problems featuring space-parameter interaction. To illustrate this, we report below a simple, yet remarkable, example.

2.1 An instructive example

Let us consider a stationary 2D fluid flow, modeled by the steady incompressible Navier–Stokes equations, occurring in the spatial domain depicted in Fig. 1a. The flow goes from left to right, passing around an almond-shaped obstacle whose diameter is roughly a fifth of the channel width. We consider a parametrized scenario depending on five scalar parameters:

- $\boldsymbol{\nu} = [\alpha, \beta]$, with $0 \leq \alpha, \beta \leq 10$, which parametrize the inflow condition at boundary Γ_{in}, imposing a Dirichlet condition of the form

$$\boldsymbol{u}(0, y) = y(1-y)\left(\alpha e^{-100(y-0.25)^2} + \beta e^{-100(y-0.75)^2}\right)^{1/2} \qquad \forall y \in [0, 1].$$

  Larger values of α correspond to a stronger flow at the bottom, while larger values of β increase the fluid velocity at the top;
- $\boldsymbol{\mu} = [\theta, x_0, y_0]$, which parametrize the rotation, $0 \leq \theta \leq 2\pi$, and the location of the obstacle, $0.25 \leq x_0, y_0 \leq 0.75$.

For each parametric configuration $(\boldsymbol{\mu}, \boldsymbol{\nu})$, we discretize the parametric domain in Fig. 1 using a triangular mesh and using the finite element method to solve the Navier–Stokes equations. Then, in order to embed all PDE solutions within a common state space $V_h \cong \mathbb{R}^{N_h}$, we interpolate all velocity fields over a uniform grid defined over the unit square $\Omega := (0, 1)^2$. For further details on the matter, we refer the interested reader to Section 4.1.

Intuitively, it is clear that the two vectors of parameters, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, play fundamentally different roles. The former are more geometrical in nature and can significantly affect the behavior of the solution in multiple ways. The variability introduced by the

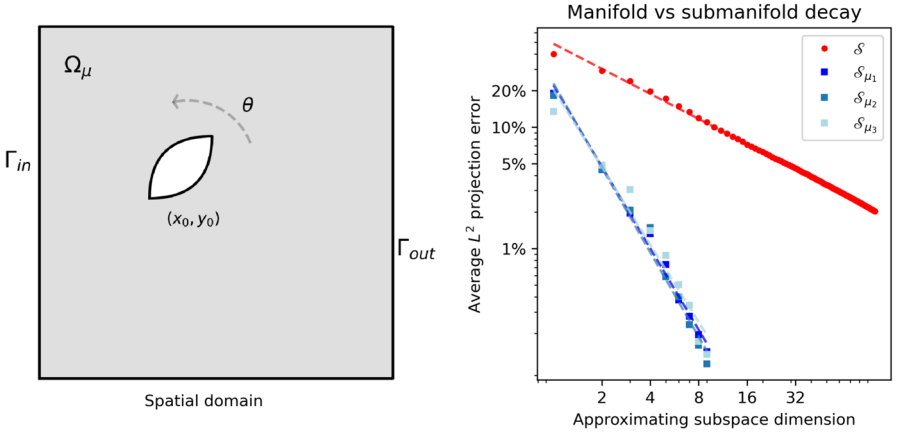

Fig. 1 Spatial domain (left) and projection error analysis (right) for the Navier–Stokes case study, Sections 2.1 and 4.1. Left: the domain $\Omega_{\boldsymbol{\mu}}$ is obtained by removing an almond-shaped object from the unit square $(0, 1)^2$. The parameters $\boldsymbol{\mu} = [\theta, x_0, y_0]$ determine the rotation and the position of the obstacle. Right: decay of the projection error for increasingly larger subspaces, highlighting the differences between the solution manifold $\mathscr{S}$ and its $\boldsymbol{\mu}$-slices $\mathscr{S}_{\boldsymbol{\mu}}$. Here, $\boldsymbol{\mu}_1 = [0, 0.5, 0.5]$, $\boldsymbol{\mu}_2 = [\pi/4, 0.4, 0.6]$ and $\boldsymbol{\mu}_3 = [\pi/2, 0.7, 0.3]$

remaining parameters is, instead, much simpler to describe, as they can only distribute the flow intensity either at the top or at the bottom. For example, if we fix a geometric configuration $\boldsymbol{\mu}$, and let $\boldsymbol{\nu}$ vary, we can speculate that the corresponding flow will be given, roughly speaking, by the superposition of two main modes: one describing the flow at the top and one describing the flow at the bottom. In this sense, the submanifold $\mathscr{S}_{\boldsymbol{\mu}}$, which consists of all PDE solutions for fixed $\boldsymbol{\mu}$ and variable $\boldsymbol{\nu}$, should be well approximated by a small linear subspace. In contrast, if we allow $\boldsymbol{\mu}$ to change, this well-structured behavior is likely to disappear.

Indeed, this is also what we observe in practice. Figure 1b shows the decay of the projection error for increasingly large linear subspaces, comparing the behavior of the whole solution manifold $\mathscr{S}$ with that of the submanifolds $\mathscr{S}_{\boldsymbol{\mu}}$, computed for different values of $\boldsymbol{\mu}$. Here, in order to estimate these trends, we relied on proper orthogonal decomposition: while we refrain from delving into technical details in this context, readers keen on a more rigorous explanation can refer to Section 4.

Even from a qualitative point of view, it is clear that Fig. 1b confirms our intuition. Furthermore, a more thorough examination suggests that Assumptions A1 and A2 hold true: in fact, a least-square regression in logarithmic scale yields a decay rate of 0.686 for $\mathscr{S}$, and a much steeper slope for the submanifolds $\mathscr{S}_{\boldsymbol{\mu}}$ (exponent ranging from 2.119 to 2.310).

In general, despite its simplicity, this case study highlights how certain problems strongly call for an adaptive basis approach and how common this situation can be.

3 Deep orthogonal decomposition (I): dimensionality reduction

We devote this section to the presentation of the DOD algorithm, first discussing its applicability in the broader context of dimensionality reduction. A subsequent discussion on the use of DOD for reduced order modeling will be provided in Section 5. Following the notation in Section 2, let

$$\mathcal{F} : \Theta \times \Theta' \ni (\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^{N_h}$$

be a parameter-to-solution operator, where the model parameters have been subdivided into two groups: those responsible for the slow decay in the Kolmogorov n-width of $\mathscr{S} = \mathcal{F}(\Theta \times \Theta')$, collected in the vector $\boldsymbol{\mu}$, and those whose effect is rapidly captured by linear combinations, stored in $\boldsymbol{\nu}$. The idea is to construct a deep neural network model $\mathbf{V} : \Theta \to \mathbb{R}^{N_h \times n}$, called the DOD, which is capable of parametrizing a suitable modal basis $\mathbb{V}_{\boldsymbol{\mu}} := \mathbf{V}(\boldsymbol{\mu})$ that changes adaptively with $\boldsymbol{\mu}$ in a highly efficient manner. The reason for this is that we would like to take advantage of the nice behavior of the variables $\boldsymbol{\nu}$ as much as possible, while simultaneously isolating the difficulty of handling $\boldsymbol{\mu}$. In practice, having fixed a latent dimension n, we seek for a suitable matrix-valued DNN architecture

$$\mathbf{V} : \Theta \ni \boldsymbol{\mu} \mapsto \mathbb{V}_{\boldsymbol{\mu}} \in \mathbb{R}^{N_h \times n}, \tag{5}$$

such that, for any given $\boldsymbol{\mu} \in \Theta$, the matrix $\mathbb{V}_{\boldsymbol{\mu}}$ acts as a good local basis for the submanifold $\mathscr{S}_{\boldsymbol{\mu}} = \{\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\}_{\boldsymbol{\nu}\in\Theta'} = \mathcal{F}(\boldsymbol{\mu}, \Theta')$. From a quantitative point of view, this boils down to requiring

$$\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \approx \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}$$

in $\|\cdot\|$-norm. Then, if the projection error is sufficiently small, the n-dimensional vector of DOD coefficients

$$\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}$$

can be used as a proxy for the overall high-fidelity solution $\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^{N_h}$. In fact, the latter can be easily recovered via the lifting $\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} \mapsto \mathbb{V}_{\boldsymbol{\mu}}\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}}$.

As exemplified by the theoretical result below, this procedure is mathematically sound and further motivated whenever Assumptions A1 and A2 are satisfied. Hereon, we use $\mathbb{E}$ to denote the expectation operator. Specifically, given two probability distributions, $\mathbb{P}$ and $\mathbb{Q}$, defined over Θ and Θ', respectively, for any measurable map $f : \Theta \times \Theta' \to [0, +\infty]$ we let

$$\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}[f(\boldsymbol{\mu}, \boldsymbol{\nu})] := \int_{\Theta}\int_{\Theta'} f(\boldsymbol{\mu}, \boldsymbol{\nu})\mathbb{P}(d\boldsymbol{\mu})\mathbb{Q}(d\boldsymbol{\nu}).$$

Similarly, given $g : \Theta \to [0, +\infty]$, we set $\mathbb{E}_{\boldsymbol{\mu}}[g(\boldsymbol{\mu})] := \int_{\Theta} g(\boldsymbol{\mu})\mathbb{P}(d\boldsymbol{\mu})$. Finally, we write *ReLU* for the *Rectified Linear Unit*, that is, the scalar map $x \mapsto \max\{0, x\}$. We refer to a neural network as being a *ReLU network* if all of its hidden layers employ the ReLU activation.

Theorem 1 *Let $\Theta \subset \mathbb{R}^p$ and $\Theta' \subset \mathbb{R}^{p'}$ be two compact sets, equipped, respectively, with two probability distributions, $\mathbb{P}$ and $\mathbb{Q}$. Let*

$$\mathcal{F} : \Theta \times \Theta' \ni (\boldsymbol{\mu}, \boldsymbol{\nu}) \to \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^{N_h}$$

be continuous. For each $\boldsymbol{\mu} \in \Theta$, let $\mathscr{S}_{\boldsymbol{\mu}} := \{\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\}_{\boldsymbol{\nu}\in\Theta'} \subset \mathcal{F}(\Theta \times \Theta')$ be the $\boldsymbol{\mu}$-submanifold in the image of $\mathcal{F}$. Let $\mathbb{G}$ be the Gram matrix associated with a given inner product in $\mathbb{R}^{N_h}$, and let $\|\cdot\|$ be the corresponding norm. Then, for every $\varepsilon > 0$, there exists a ReLU matrix-valued deep neural network $\mathbf{V} : \mathbb{R}^p \to \mathbb{R}^{N_h \times n}$ such that

$$\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\| < \varepsilon + \mathbb{E}_{\boldsymbol{\mu}}\left[d_n(\mathscr{S}_{\boldsymbol{\mu}})\right],$$

where $\mathbb{V}_{\boldsymbol{\mu}} := \mathbf{V}(\boldsymbol{\mu})$.

Proof With little abuse of notation, we write $|\cdot|$ to indicate both the Euclidean norm and the Frobenius norm. Fix $n \in \mathbb{N}$. Let $\lambda > 0$ be the smallest eigenvalue of the positive definite matrix $\mathbb{G}$. We notice that, if $\mathbb{V}$ is a matrix whose columns are orthonormal with respect to $\mathbb{G}$, then all the entries of $\mathbb{V}$ must be smaller, in modulus, than $1/\sqrt{\lambda}$. In fact, by the min-max properties of eigenvalues, for any entry $v_{i,j}$ in $\mathbb{V}$, one has

$$\lambda v_{i,j}^2 \leq \lambda|\mathbf{v}_i|^2 \leq \mathbf{v}_i^{\top}\mathbb{G}\mathbf{v}_i = 1,$$

with $\mathbf{v}_i$ the ith column in $\mathbb{V}$. In light of this, we shall assume, without loss of generality, that $\lambda = 1$ (if not, we may always rescale). Let $J : \Theta \times [-1,1]^{N_h \times n} \to \mathbb{R}$ be the following objective functional

$$J(\boldsymbol{\mu}, \mathbb{V}) = \mathbb{E}_{\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}\mathbb{V}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|,$$

and fix any $\varepsilon > 0$. We recall that, since J is continuous and $\Theta \times [-1,1]^{N_h}$ is compact, J is uniformly continuous. In particular, there exists $\delta > 0$ such that

$$|\boldsymbol{\mu} - \boldsymbol{\mu}'| + |\mathbb{V} - \mathbb{V}'| < \delta \implies |J(\boldsymbol{\mu}, \mathbb{V}) - J(\boldsymbol{\mu}', \mathbb{V}')| < \varepsilon.$$

Then, classical results on measurable selections (see, e.g., [42, Theorem 8.1.3]) show that there exists a Borel measurable map $s : \Theta \to [-1,1]^{N_h \times n}$ acting as

$$s : \boldsymbol{\mu} \to \operatorname*{argmin}_{\mathbb{V} \in [-1,1]^{N_h \times n}} J(\boldsymbol{\mu}, \mathbb{V}).$$

Since s is bounded, it follows that $s \in L^1(\Theta; \mathbb{R}^{N_h \times n})$ in the Bochner sense. In particular, see Lemma 1, there exists a deep ReLU neural network $\mathbf{V} : \Theta \to [-1,1]^{N_h \times n}$ such that $\mathbb{E}|\mathbb{V}_{\boldsymbol{\mu}} - s(\boldsymbol{\mu})| < \delta$. By uniform continuity of J,

$$|J(\boldsymbol{\mu}, s(\boldsymbol{\mu})) - J(\boldsymbol{\mu}, \mathbb{V}_{\boldsymbol{\mu}})| < \varepsilon \qquad \forall \boldsymbol{\mu} \in \Theta.$$

Consequently, as $\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}} \mathbb{V}_{\boldsymbol{\mu}}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\| = \mathbb{E}_{\boldsymbol{\mu}} \left[J(\boldsymbol{\mu}, \mathbb{V}_{\boldsymbol{\mu}}) \right]$, we have

$$\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}} \mathbb{V}_{\boldsymbol{\mu}}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\| \leq \mathbb{E}_{\boldsymbol{\mu}} \|J(\boldsymbol{\mu}, \mathbb{V}_{\boldsymbol{\mu}}) - J(\boldsymbol{\mu}, s(\boldsymbol{\mu}))\| + \mathbb{E}_{\boldsymbol{\mu}} \left[J(\boldsymbol{\mu}, s(\boldsymbol{\mu})) \right]$$

$$\implies \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}} \mathbb{V}_{\boldsymbol{\mu}}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\| < \varepsilon + \mathbb{E}_{\boldsymbol{\mu}} \left[J(\boldsymbol{\mu}, s(\boldsymbol{\mu})) \right].$$

Since

$$\mathbb{E}_{\boldsymbol{\mu}} \left[J(\boldsymbol{\mu}, s(\boldsymbol{\mu})) \right] = \mathbb{E}_{\boldsymbol{\mu}} \left[\min_{\mathbb{V}} J(\boldsymbol{\mu}, \mathbb{V}) \right] \leq \mathbb{E}_{\boldsymbol{\mu}} \left[\min_{\mathbb{V}} \sup_{\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}\mathbb{V}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\| \right] = \mathbb{E}_{\boldsymbol{\mu}} \left[d_n(\mathscr{S}_{\boldsymbol{\mu}}) \right],$$

the conclusion follows. □

Lemma 1 *Let $\mathbb{P}$ be a probability distribution over $\mathbb{R}^p$. Let $\boldsymbol{s} : \mathbb{R}^p \to [-1,1]^q$ be a measurable map. Then, for every $\delta > 0$, there exists a ReLU deep neural network $\boldsymbol{v} : \mathbb{R}^p \to \mathbb{R}^q$ such that $\mathbb{E}|\boldsymbol{v} - \boldsymbol{s}| < \delta$. Furthermore, $\boldsymbol{v}$ can be chosen so that $\boldsymbol{v}(\mathbb{R}^p) \subseteq [-1,1]^q$.*

Proof By Hornik's Theorem [43], there exists a ReLU deep neural network $\boldsymbol{v}_0 : \mathbb{R}^p \to \mathbb{R}^q$ such that $\mathbb{E}|\boldsymbol{v}_0 - \boldsymbol{s}| < \delta$. Let now ρ denote the ReLU activation function and consider the three-layer network $L : \mathbb{R}^q \to \mathbb{R}^q$ given by

$$L(\mathbf{x}) = \mathbf{e} - \mathbb{I}_q \rho(-\mathbb{I}_q \rho(\mathbb{I}_q \mathbf{x} + \mathbf{e}) + 2\mathbf{e}),$$

where $\mathbb{I}_q$ is the $q \times q$ identity matrix, and $\mathbf{e} := [1, \dots, 1]^\top \in \mathbb{R}^q$, so that, with little abuse of notation, one has $L(\mathbf{x}) = 1 - \rho(2 - \rho(\mathbf{x} + 1))$. The latter acts as follows: given any input $\mathbf{x} \in \mathbb{R}^q$, it leaves unchanged all those entries lying in $[-1, 1]$, while it squashes the rest to ± 1. Then, it is straightforward to see that $\boldsymbol{v} := L \circ \boldsymbol{v}_0$ fulfills all the desired properties. In fact, by construction, $\boldsymbol{v}(\boldsymbol{\mu}) \in [-1, 1]^q$ for all $\boldsymbol{\mu} \in \mathbb{R}^p$; furthermore,

$$\mathbb{E}|\boldsymbol{v} - \boldsymbol{s}| = \mathbb{E}|L \circ \boldsymbol{v}_0 - L \circ \boldsymbol{s}| \leq \mathbb{E}|\boldsymbol{v}_0 - \boldsymbol{s}| < \delta,$$

as L is 1-Lipschitz.

Clearly, Theorem 1 is only an existence result. In particular, it does not provide an answer to three main questions: (i) how to construct such a network, (ii) how to train it, and (iii) whether the overall approach is computationally feasible. We shall start by answering the first two questions, while we leave the third one to the numerical experiments (Section 4).

Remark 1 In this work, we adopt a fully algebraic perspective, as that can be more natural in the context of model order reduction. However, to better understand the overall idea, it may be useful to discuss the implications of the DOD approach at the continuous level. To this end, let us write the PDE solution as a map $u = u(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu})$, depending explicitly on the space variable and on the model parameters. Then, at the continuous level, the POD decomposition corresponds to a separation of variable approach of the form

$$u(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) \approx \sum_{i=1}^{n} v_i(\mathbf{x}) \phi_i(\boldsymbol{\mu}, \boldsymbol{\nu}), \tag{6}$$

where n is the reduced dimension, and v_i corresponds to the ith mode (represented by the ith column in the POD matrix), while $\phi_i(\boldsymbol{\mu}, \boldsymbol{\nu})$ is the corresponding parameter-dependent coefficient. With this formalism, the DOD approach can be regarded as

$$u(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) \approx \sum_{i=1}^{n} v_i(\mathbf{x}, \boldsymbol{\mu}) \phi_i(\boldsymbol{\mu}, \boldsymbol{\nu}), \tag{7}$$

effectively presenting a $\boldsymbol{\mu}$-adaptive basis. From an intuitive point of view, Eq. (7) emphasizes the fact that the "space-interacting" parameters, $\boldsymbol{\mu}$, should not be decoupled from the space variable, $\mathbf{x}$, when approximating u.

3.1 DOD architecture design

Having to deal with remarkably large dimensions, from p to $N_h \times n$, the construction of a DOD architecture requires some discussion. In principle, the high dimension at output could be tackled by relying on suitable layer types, specifically designed for handling high-dimensional data, such as, e.g., convolutional models (CNNs) [44], graph neural networks (GNNs) [45], or mesh-informed neural networks (MINNs) [46]. However, all these approaches have limited scalability: as of today, using these architectures to address problems with $N_h \sim 10^4 - 10^6$ degrees of freedom requires a significant amount of computational resources, often beyond practical feasibility.

For this reason, and to be as general as possible, we propose a simpler approach, based on the introduction of a suitable *ambient space*, approximating the original state space. Simply put, we start by introducing an ambient matrix $\mathbb{A} \in \mathbb{R}^{N_h \times N_A}$, where N_A is smaller than N_h but still fairly large, e.g., $N_A \sim 10^2 - 10^3$, such that $\mathbb{A}$ is $\mathbb{G}$-orthonormal and

$$\mathbf{u}_{\boldsymbol{\mu},\nu} \approx \mathbb{A}\mathbb{A}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\nu},$$

with a given tolerance. In practice, $\mathbb{A}$ can be constructed by computing a preliminary POD over the training snapshots: see, e.g., Algorithm 3 in Appendix 8.1. However, we remark that this is only a preliminary reduction whose purpose is to make the FOM data more manageable; by no means, we assume N_A to be small: the actual reduction in dimensionality will be carried out by the DOD. In a way, this intermediate step corresponds to rewriting FOM solutions in a more convenient, problem-specific way: in fact, while the FOM basis is defined a priori, the ambient space is constructed a posteriori by leveraging the training data. We also note that, in spirit, this is the same trick adopted by other techniques, such as POD-DL-ROM [47] and POD-DeepONet [48]. The main advantage of this maneuver is that the DOD network can now be constructed as

$$\mathbb{V}_{\boldsymbol{\mu}} := \mathbb{A}\tilde{\mathbb{V}}_{\boldsymbol{\mu}},$$

where $\tilde{\mathbf{V}} : \mathbb{R}^p \to \mathbb{R}^{N_A \times n}$ is the learnable component of the architecture and $\tilde{\mathbb{V}}_{\boldsymbol{\mu}} := \tilde{\mathbf{V}}(\boldsymbol{\mu})$ denotes its output, coherently with the notation adopted so far. In particular, since N_A can be orders of magnitude smaller than N_h, adopting this approach can substantially reduce the number of trainable parameters in the DOD, thus simplifying its design and optimization. We call $\tilde{\mathbf{V}}$ the *inner module* of the DOD. Notice that $\mathbb{V}_{\boldsymbol{\mu}}$ is $\mathbb{G}$-orthonormal if and only if $\tilde{\mathbb{V}}_{\boldsymbol{\mu}}$ is orthonormal in the Euclidean sense,

$$\mathbb{V}_{\boldsymbol{\mu}}^\top \mathbb{G}\mathbb{V}_{\boldsymbol{\mu}} = \tilde{\mathbb{V}}_{\boldsymbol{\mu}}^\top \mathbb{A}^\top \mathbb{G}\mathbb{A}\tilde{\mathbb{V}}_{\boldsymbol{\mu}} = \tilde{\mathbb{V}}_{\boldsymbol{\mu}}^\top \tilde{\mathbb{V}}_{\boldsymbol{\mu}}.$$

To construct the matrix-valued network $\tilde{\mathbb{V}}$, we use a composite architecture comprised of the following:

i) A *seed* module, $s : \mathbb{R}^p \to \mathbb{R}^l$, whose purpose is to pre-process the input parameters by mapping them onto a suitable feature space
ii) A collection of *root* modules, $R_1, \ldots, R_n : \mathbb{R}^l \to \mathbb{R}^{N_A}$ operating in parallel, whose purpose is to compute the several columns of the (inner) DOD projector $\tilde{\mathbb{V}}$
iii) An *ORTH unit*, that is, a nonlearnable block ensuring orthnormality of the final output. The latter accepts a matrix $\mathbb{W} \in \mathbb{R}^{N_A \times n}$ and returns a corresponding orthonormal matrix $\tilde{\mathbb{W}} := \mathrm{ORTH}(\mathbb{W}) \in \mathbb{R}^{N_A \times n}$ such that $\mathrm{span}(\mathbb{W}) = \mathrm{span}(\tilde{\mathbb{W}})$. In practice, this can be achieved in many equivalent ways, e.g., via reduced QR decomposition [49] or via Gram–Schmidt orthogonalization [50], both of which support backpropagation.

Both the seed and the root components are implemented via classical deep feed-forward neural networks. The overall workflow can be summarized as in Fig. 2 or, in formula, as

$$\mathbb{V}_{\boldsymbol{\mu}} = \mathbb{A}\mathrm{ORTH}\left(\left[R_1(s_{\boldsymbol{\mu}}), \ldots, R_n(s_{\boldsymbol{\mu}})\right]\right),$$

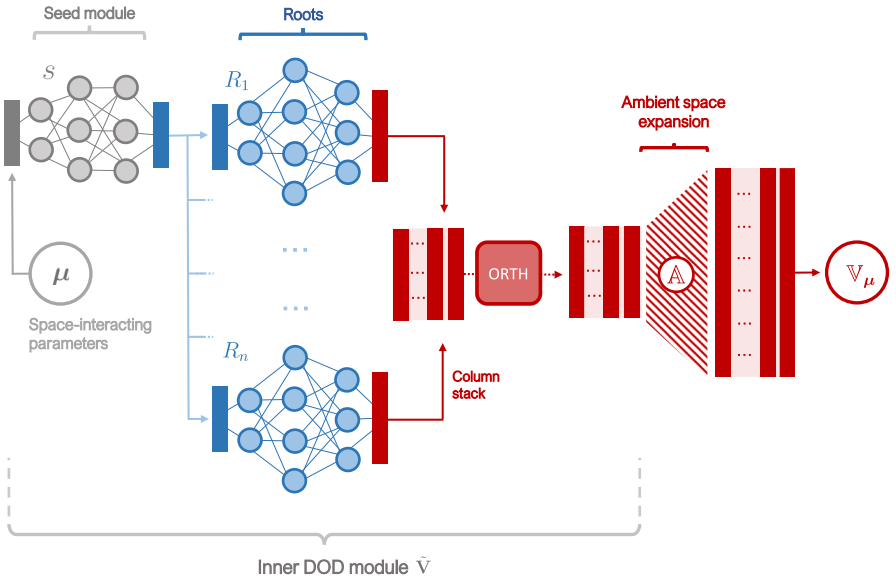

Fig. 2 Sketch of a DOD architecture. The seed module (gray block) performs a preprocessing of the input parameters, returning a suitable hidden representation. The latter is the input of the root modules (blue block), each of which outputs a column vector: together, these vectors form a basis for the desired local subspace. For convenience, all vectors are written using the notation inherited from the ambient space, entailing N_A degrees of freedom; after orthonormalization, the basis is expanded at the FOM level using the ambient matrix $\mathbb{A}$ (red block). The first two blocks (gray and blue) constitute the learnable part of the architecture

where $s_{\boldsymbol{\mu}} := s(\boldsymbol{\mu})$.

Remark 2 The ambient space is only a practical expedient that we have introduced in order to tackle arbitrarily large FOMs. However, if N_h is reasonably small, this step can be omitted, and one may work directly at the FOM level. In practice, if $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_{N_h}$ is an orthonormal basis derived from $\varphi_1, \ldots, \varphi_{N_h}$, that is, from the original FOM basis, this would be equivalent to setting $\mathbb{A} := [\tilde{\boldsymbol{\varphi}}_1, \ldots, \tilde{\boldsymbol{\varphi}}_{N_h}]$, so that $\mathbb{A}^\top \mathbb{G} \mathbb{A} = \mathbb{I}$ and $\text{span}(\mathbb{A}) = \mathbb{R}^{N_h}$.

Remark 3 The problem of mitigating the issues posed by a slow decay of the Kolmogorov n-width is timely. Recently, in [40], some authors have proposed the use of neural networks combined with POD in order to address this challenge. The idea, however, is fundamentally different from ours. In [40], the authors use POD to compress PDE solutions and then rely upon neural networks to improve the quality of the decoding process, which can be understood both as a closure modeling technique but also as a nonlinear manifold approximation method. Here, instead, we are adopting an adaptive basis perspective. For each $\boldsymbol{\mu}$, the encoding-decoding process is purely linear; indeed, it is the dependency on $\boldsymbol{\mu}$ that makes the method nonlinear as a whole. In particular, our strategy is tailored to a specific class of problems where assumptions A1-A2 hold. Nonetheless, the two methods do share some similarities, such as the use of an ambient space to represent PDE solutions, as well as the deployment of

neural networks to overcome the Kolmogorov barrier and thus obtain more compact representations.

3.2 Model training

In order to learn the DOD basis, we propose a supervised training strategy based on a variational principle, where the DOD architecture $\mathbf{V}$ is trained by minimizing the reconstruction error over the training data, as depicted in Algorithm 1. In other words, we learn the DOD by minimizing the loss function below:

$$\mathscr{L}(\mathbf{V}) := \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{V}_{\boldsymbol{\mu}_i} \mathbb{V}_{\boldsymbol{\mu}_i}^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2, \tag{8}$$

where $\{\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\}_{i=1}^{N_{\text{train}}} \subset \Theta \times \Theta' \times \mathbb{R}^{N_h}$ are high-fidelity samples generated—at random—by repeated calls to the FOM solver. This approach is fairly intuitive, as it defines the DOD projector following the same minimization principle of POD.

In practice, since optimizing (8) can be computationally demanding, we can take advantage of the existence of the ambient space, $\mathbb{A}$, in order to ease computational effort. In fact, minimizing (8) is equivalent to minimizing

$$\mathscr{L}_{\mathbb{A}}(\tilde{\mathbf{V}}) := \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} |\tilde{\mathbf{u}}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \tilde{\mathbb{V}}_{\boldsymbol{\mu}_i} \tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}^{\top} \tilde{\mathbf{u}}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}|^2, \tag{9}$$

where $\tilde{\mathbf{u}}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} := \mathbb{A}^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$ and $|\cdot|$ denotes the Euclidean norm. Indeed, the two loss functions only differ by a constant: see Lemma 2 in the following.

Lemma 2 *Let $\mathbb{A} \in \mathbb{R}^{N_h \times N_A}$ be $\mathbb{G}$-orthonormal, and let $n \leq N_A$. Let $\tilde{\mathbf{V}} : \mathbb{R}^p \to \mathbb{R}^{N_A \times n}$ be any matrix-valued map. Define $\mathbf{V} : \mathbb{R}^p \to \mathbb{R}^{N_h \times n}$ as $\mathbf{V}(\boldsymbol{\mu}) := \mathbb{A}\tilde{\mathbf{V}}(\boldsymbol{\mu})$. Then,*

$$\mathscr{L}(\mathbf{V}) = c_{\mathbb{A}} + \mathscr{L}_{\mathbb{A}}(\tilde{\mathbf{V}}), \tag{10}$$

where $c_{\mathbb{A}} > 0$ is a constant depending on $\mathbb{A}$ and on the training data.

Proof Fix any $\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$ in the training set. Since the ambient residual $\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{A}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$ is $\mathbb{G}$-orthonormal to span$(\mathbb{A})$, it follows that

$$\begin{aligned}
\|\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{V}_{\boldsymbol{\mu}_i}\mathbb{V}_{\boldsymbol{\mu}_i}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2 &= \|\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{A}\tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}\tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}^{\top}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2 = \\
&= \|\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{A}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2 + \|\mathbb{A}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{A}\tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}\tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}^{\top}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2 = \\
&= \|\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \mathbb{A}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\|^2 + |\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}\tilde{\mathbb{V}}_{\boldsymbol{\mu}_i}^{\top}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}|^2.
\end{aligned}$$

Averaging over the training set yields (10). □

Algorithm 1 Construction and training of the DOD architecture.

Input : FOM solver $\texttt{FOM} = \texttt{FOM}(\boldsymbol{\mu}, \boldsymbol{\nu})$, parameter spaces Θ and Θ', inner module architecture class $\mathscr{D}$ of reduced dimension n, sample size N_{train}, Gram matrix $\mathbb{G}$, ambient dimension N_A with $N_A > n$.

Output: Trained DOD model $\mathbf{V}$.

$[\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{N_{\text{train}}}] \leftarrow$ i.i.d. random sample from Θ

$[\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_{N_{\text{train}}}] \leftarrow$ i.i.d. random sample from Θ'

$[\mathbf{u}_1, \dots, \mathbf{u}_{N_{\text{train}}}] \leftarrow [\texttt{FOM}(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i) \text{ for } i = 1 : N_{\text{train}}]$ *// sampling*

$\mathbb{A} \leftarrow \texttt{POD}([\mathbf{u}_1, \dots, \mathbf{u}_{N_{\text{train}}}],\ \mathbb{G},\ N_A)$ *// ambient space definition*

$[\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_{N_{\text{train}}}] \leftarrow [\mathbb{A}^\top \mathbb{G} \mathbf{u}_i \text{ for } i = 1 : N_{\text{train}}]$ *// ambient projection*

$\tilde{\mathbf{V}}_* \leftarrow \operatorname{argmin}_{\tilde{\mathbf{V}} \in \mathscr{D}} \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} |\tilde{\mathbf{u}}_i - \tilde{\mathbf{V}}(\boldsymbol{\mu}_i)\tilde{\mathbf{V}}^\top(\boldsymbol{\mu}_i)\tilde{\mathbf{u}}_i|^2$ *// training*

$\mathbf{V} \leftarrow \mathbb{A}\tilde{\mathbf{V}}_*$ *// map composition*

return $\mathbf{V}$

Hence, to summarize, the implementation and training of a DOD network can be carried out as follows. First, following the guidelines presented in Section 3.1, we design the model architecture. According to Fig. 2, this corresponds to fixing an ambient space $\mathbb{A}$, a latent dimension n, and a set of neural network architectures defining, respectively, the seed and the roots modules. From an abstract point of view, this is equivalent to identifying a suitable hypothesis class

$$\mathscr{D} \subset \{\tilde{\mathbf{V}} : \Theta \to \mathbb{R}^{N_A \times n}\}.$$

for the inner DOD module. Then, we rely on classical optimization algorithms, such as L-BFGS or Adam, to solve the following minimization problem

$$\min_{\tilde{\mathbf{V}} \in \mathscr{D}} \mathscr{L}_{\mathbb{A}}(\tilde{\mathbf{V}}),$$

and thus train the (inner) DOD network.

3.3 Quantifying adaptivity

After training, it can be useful to quantify the actual adaptivity of the DOD basis: that is, to which extent the map $\boldsymbol{\mu} \mapsto \mathbb{V}_{\boldsymbol{\mu}}$ is non-constant over the parameter space Θ. As we shall see in the next section, this postprocessing can substantially increase our understanding of the DOD approach and help us in designing better models.

In principle, measuring the variability of the DOD basis across Θ might seem straightforward: in fact, since the DOD is explicitly given (and in closed form) by a neural network model, we can easily compute quantities such as derivatives and variances. However, this is not the full story, and things are actually more complicated.

To appreciate this, let us consider a very simple example where $p = 1$, $N_h = 3$ and $\|\cdot\|$ is the Euclidean norm. Consider the DOD model below:

$$\mu \mapsto \mathbb{V}_\mu := \begin{bmatrix} \cos\mu & -\sin\mu \\ \sin\mu & \cos\mu \\ 0 & 0 \end{bmatrix}. \tag{11}$$

At first sight, it may look like the DOD basis is adaptively changing with the input parameter μ. However, this is not really the case. In fact,

$$\text{span}(\mathbb{V}_\mu) = \text{span}(\mathbb{V}_0)$$

for all $\mu \in \mathbb{R}$. In particular, since the projection error depends only on the underlying subspace and not on the matrix representation, the maps $\mu \mapsto \mathbb{V}_\mu$ and $\mu \mapsto \mathbb{V}_0$ are actually equivalent. In this sense, a DOD network acting as (11) would not be adaptive at all.

These considerations are key, as they bring us to the following observation: for what we care, the outputs of a DOD network are not matrices, but *subspaces*. As such, the variability of the DOD basis is better understood in terms of the so-called *Grassmann manifold* [31, 51–53]. For a given dimension n and a suitable ambient space $\mathcal{A}$, the Grassmann manifold consists of all subspaces $\mathcal{V} \subseteq \mathcal{A}$ of dimension n, namely

$$\mathcal{G}_n(\mathcal{A}) := \{\mathcal{V} \subseteq \mathcal{A} \text{ such that } \mathcal{V} \text{ is linear and } \dim(\mathcal{V}) = n\},$$

defined whenever $1 \leq n \leq \dim(\mathcal{A})$. The Grassmann manifold can be equipped with different metrics—see, e.g., [54] for a comprehensive list—specifically designed for measuring distances between subspaces. Here, by noting that the variability of $\mathbf{V}$ ultimately depends on that of its inner module $\tilde{\mathbf{V}}$, we focus on the case $\mathcal{A} = \mathbb{R}^{N_A}$ and we endow the Grassmann manifold with the following metric, commonly referred to as the *projection 2-norm* [54],

$$d(\mathcal{V}, \mathcal{W}) := \max_{\substack{\mathbf{w}\in\mathcal{W} \\ |\mathbf{w}|=1}} \min_{\mathbf{v}\in\mathcal{V}} |\mathbf{w} - \mathbf{v}|,$$

so that $d : \mathcal{G}_n(\mathbb{R}^{N_A}) \times \mathcal{G}_n(\mathbb{R}^{N_A}) \to [0, 1]$. Equivalently, if $\tilde{\mathbb{V}}$ and $\tilde{\mathbb{W}}$ are orthonormal matrices representing the two subspaces $\mathcal{V}$ and $\mathcal{W}$, respectively, then $d(\mathcal{V}, \mathcal{W}) = \sqrt{1 - \sigma_n^2}$, where σ_n is the smallest singular value of $\tilde{\mathbb{V}}^\top \tilde{\mathbb{W}}$: cf. Algorithm 2. In light of this, with little abuse of notation, we shall write $d(\tilde{\mathbb{V}}, \tilde{\mathbb{W}})$ to intend $d(\text{span}(\tilde{\mathbb{V}}), \text{span}(\tilde{\mathbb{W}})) = d(\mathcal{V}, \mathcal{W})$.

With this setup, we can think of the inner DOD module as a map

$$\tilde{\mathbf{V}} : \Theta \to \mathcal{G}_n(\mathbb{R}^{N_A}),$$

and exploit the metric d to define a suitable adaptivity-score. We do this by relying on a generalization of the statistical variance, specifically designed for random variables

Algorithm 2 Computation of the distance d over the Grassmann manifold $\mathcal{G}_n(\mathbb{R}^{N_A})$.

Input : Matrices $\mathbb{V}, \mathbb{W} \in \mathbb{R}^{N_A \times n}$.
Output: Metric distance d between span($\mathbb{V}$) and span($\mathbb{W}$).
$\tilde{\mathbb{V}} \leftarrow$ ORTH($\mathbb{V}$) \\ *Orthonormalization*
$\tilde{\mathbb{W}} \leftarrow$ ORTH($\mathbb{W}$)
$[\sigma_1, \ldots, \sigma_n] \leftarrow$ singular values of $\tilde{\mathbb{V}}^\top \tilde{\mathbb{W}}$ (in decreasing order)
return $\sqrt{1 - \sigma_n^2}$

in metric spaces—see, e.g., [55] or Remark 4 at the end of this section—that is,

$$\mathrm{Var}(\tilde{\mathbf{V}}) := \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\mu}'}\left[\frac{1}{2} d^2(\tilde{\mathbb{V}}_{\boldsymbol{\mu}}, \tilde{\mathbb{V}}_{\boldsymbol{\mu}'})\right], \tag{12}$$

where $\boldsymbol{\mu}, \boldsymbol{\mu}' \sim \mathbb{P}$ are i.i.d. (independent and identically distributed). Since, $0 \leq \mathrm{Var}(\tilde{\mathbf{V}}) \leq 1/2$ by construction, we define the *DOD adaptivity-score* as a normalized standard-deviation,

$$\mathrm{Adpt}(\mathbf{V}) := \sqrt{2\mathrm{Var}(\tilde{\mathbf{V}})} = \mathbb{E}^{1/2}_{\boldsymbol{\mu},\boldsymbol{\mu}'}\left[d^2(\tilde{\mathbb{V}}_{\boldsymbol{\mu}}, \tilde{\mathbb{V}}_{\boldsymbol{\mu}'})\right], \tag{13}$$

so that $0 \leq \mathrm{Adpt}(\mathbf{V}) \leq 1$. The adaptivity-score has the following interpretation: values close to zero indicate that the DOD is collapsing towards a unique global basis; conversely, larger scores correspond to a larger variability across Θ. See also Remark 5 for further insights.

In practice, since the DOD can be evaluated at a negligible computational cost, we can estimate (13) via classical Monte Carlo. Here, we shall rely on the following estimator:

$$\mathrm{Adpt}_{\mathrm{MC}}(\mathbf{V}) := \sqrt{\frac{1}{N_\mathrm{r}} \sum_{i=1}^{N_\mathrm{r}} d^2(\tilde{\mathbb{V}}_{\boldsymbol{\mu}^\mathrm{r}_{2i-1}}, \tilde{\mathbb{V}}_{\boldsymbol{\mu}^\mathrm{r}_{2i}})}, \tag{14}$$

where $\{\boldsymbol{\mu}^\mathrm{r}_i\}_{i=1}^{2N_\mathrm{r}} \subset \Theta$ is a suitable i.i.d. random sample, independent of the training set. In general, letting $N_\mathrm{r} \sim 10^4$ typically suffices, as demonstrated by the following.

Lemma 3 *Fix any $\delta, \epsilon > 0$. Let $N_r = \lceil \delta^{-1}\epsilon^{-4}/4 \rceil$, and let $\{\boldsymbol{\mu}^r_i\}_{i=1}^{2N_r} \subset \Theta$ be an i.i.d. random sample, independent of the training set. Then,*

$$\left|\mathrm{Adpt}(\mathbf{V}) - \mathrm{Adpt}_{\mathrm{MC}}(\mathbf{V})\right| \leq \epsilon$$

with probability $1 - \delta$.

Proof Let Prob denote the probability law of the entire random sample. We note that, for all $a, b \geq 0$, one has $|a^2 - b^2| \geq |a - b|^2$. Therefore,

$$\begin{aligned}\mathsf{Prob}\left(\left|\mathrm{Adpt}(\mathbf{V}) - \mathrm{Adpt}_{\mathrm{MC}}(\mathbf{V})\right| > \epsilon\right) \\ \leq \mathsf{Prob}\left(\left|\mathrm{Adpt}^2(\mathbf{V}) - \mathrm{Adpt}^2_{\mathrm{MC}}(\mathbf{V})\right| > \epsilon^2\right) \\ \leq \epsilon^{-4}\mathrm{Var}(\mathrm{Adpt}^2_{\mathrm{MC}}(\mathbf{V})),\end{aligned}$$

by Chebyshev's inequality. Let $X := \mathrm{d}^2(\tilde{\mathbb{V}}_{\boldsymbol{\mu}_1^{\mathrm{r}}}, \tilde{\mathbb{V}}_{\boldsymbol{\mu}_2^{\mathrm{r}}})$. By independence, and since both X and $\mathbb{E}[X]$ take values in $[0, 1]$, we have

$$\begin{aligned}\mathrm{Var}(\mathrm{Adpt}^2_{\mathrm{MC}}(\mathbf{V})) = \frac{1}{N_{\mathrm{r}}}\mathrm{Var}(X) = \frac{1}{N_{\mathrm{r}}}\left(\mathbb{E}[X^2] - \mathbb{E}^2[X]\right) \leq \\ \leq \frac{1}{N_{\mathrm{r}}}\left(\mathbb{E}[X] - \mathbb{E}^2[X]\right) = \frac{1}{N_{\mathrm{r}}}\mathbb{E}[X]\,(1 - \mathbb{E}[X]) \leq \frac{1}{4N_{\mathrm{r}}}.\end{aligned}$$

Consequently, $\mathsf{Prob}\left(\left|\mathrm{Adpt}(\mathbf{V}) - \mathrm{Adpt}_{\mathrm{MC}}(\mathbf{V})\right| > \epsilon\right) \leq (4N_{\mathrm{r}})^{-1}\epsilon^{-4} \leq \delta$. □

Remark 4 Equation (12) can be seen as a generalization of the variance to metric spaces. To see this, consider the case of a real-valued random variable X. Classically, its variance is defined as $\mathrm{Var}(X) = \mathbb{E}[|X - \mathbb{E}[X]|^2] = \mathbb{E}[X^2] - \mathbb{E}^2[X]$. We now note that if Y is another independent random variable and $Y \sim X$ (identical distribution), then by classical properties,

$$\begin{aligned}\mathbb{E}\left[\frac{1}{2}|X - Y|^2\right] = \frac{1}{2}\left(\mathbb{E}[X^2] - 2\mathbb{E}[XY] + \mathbb{E}[Y^2]\right) = \\ = \frac{1}{2}\left(\mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[Y] + \mathbb{E}[Y^2]\right) = \\ = \frac{1}{2}\left(\mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X^2]\right) = \mathrm{Var}(X).\end{aligned}$$

In particular, if we denote the Euclidean distance by $\mathrm{d} = |\cdot|$, then $\mathbb{E}\left[\frac{1}{2}\mathrm{d}(X, Y)^2\right] = \mathrm{Var}(X)$.

Remark 5 It is worth pointing out that both the cases $\mathrm{Adpt}(\mathbf{V}) = 0$ and $\mathrm{Adpt}(\mathbf{V}) = 1$ are somewhat pathological. In the former case, in fact, the DOD basis is constant, which is equivalent to a classical POD. In the latter case, instead, the DOD would have to be discontinuous. To see this, note that the continuity of the DOD implies that of the map $g : \boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \mapsto \mathrm{d}^2(\tilde{\mathbb{V}}_{\boldsymbol{\mu}_1}, \tilde{\mathbb{V}}_{\boldsymbol{\mu}_2})$. Assume now that $\boldsymbol{\mu}$ is an absolutely continuous random variable whose density never vanishes over Θ. Then, $\mathrm{Adpt}(\mathbf{V}) = 1$ would imply $g = 1$ almost everywhere. If $\mathbf{V}$ were to be continuous, this would imply $g \equiv 1$ over $\Theta \times \Theta$; however, this is not possible, since $g(\boldsymbol{\mu}_1, \boldsymbol{\mu}_1) = 0$ for all $\boldsymbol{\mu}_1 \in \Theta$. Thus, $\mathrm{Adpt}(\mathbf{V}) = 1$ can only be achieved by a discontinuously adaptive basis. Of note, a deep learning model could never realize such scenario as the relationship between input and output in (trainable) neural networks is always continuous.

4 Numerical experiments (I): dimensionality reduction

The purpose of this section is to provide some preliminary insights into the capabilities of the DOD algorithm as a tool for dimensionality reduction. To this end, we shall present a couple of numerical experiments in which we compare the DOD with other well-established approaches, such as POD, local POD, basis interpolation, and autoencoders.

Similarly to DOD, all these techniques are data-driven, meaning that they require the preliminary collection of some FOM snapshots, $\{\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}\}_{i=1}^{N}$, randomly sampled, which serve as training data. All these approaches define a latent space, where solutions are projected (linearly or nonlinearly) and from which they can be later recovered. To evaluate the quality of the reconstruction, we rely on the mean relative projection error (MRPE)

$$\text{MRPE} := \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\|\mathbf{u}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i} - \mathbf{u}^{\text{proj}}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i}\|}{\|\mathbf{u}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i}\|}, \tag{15}$$

where $\|\cdot\|$ is the norm induced by $\mathbb{G}$ over $\mathbb{R}^{N_h}$, corresponding to the L^2-norm in V_h, while $\{\check{\boldsymbol{\mu}}_i, \check{\boldsymbol{\nu}}_i, \mathbf{u}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i}\}_{i=1}^{N_{\text{test}}}$ is the so-called test set, a collection of high-quality data generated independently of the training set. With little abuse of notation, we denote by $\mathbf{u}^{\text{proj}}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$ the reconstruction of $\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$, even though, in some cases, the encoding-deconding process may involve nonlinear transformations that go beyond linear projections. For the DOD, the model reconstruction reads

$$\mathbf{u}^{\text{proj}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}} \mathbb{V}_{\boldsymbol{\mu}}^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}.$$

In the other cases, instead, the formulas are slightly different. We report them below.

- **POD** [15]. In this case,
$$\mathbf{u}^{\text{proj}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}\mathbb{V}^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}$$
where, for a given reduced dimension n, $\mathbb{V} \in \mathbb{R}^{N_h \times n}$ represents a global basis computed via generalized SVD (analogously to Algorithm 3, up to replacing N_A with n).
- **Local POD** using clusters [26, 27, 29]. Here, the reconstruction reads
$$\begin{cases} \mathbf{u}^{\text{proj}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_j \mathbb{V}_j^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}, \\ j = \text{argmin}_{k=1,\dots,c} \ \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_k \mathbb{V}_k^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|, \end{cases}$$
where c is the number of clusters, $\mathbb{V}_1, \dots \mathbb{V}_c \in \mathbb{R}^{N_h \times n}$ is a collection of basis and n is the reduced dimension. In practice, given c and n, the FOM data are first subdivided into c clusters by grouping together similar solutions (here, we rely on the k-means algorithm); then, a POD basis is computed for each cluster, yielding the matrices $\mathbb{V}_1, \dots \mathbb{V}_c$. Then, each solution is projected and reconstructed using its own POD basis, defined as the "best" among the ones available. It can be

regarded as a primitive form of DOD, where the basis is piecewise constant over the parameter space and changes discontinuously. Typically, methods known as *dictionary-based ROMs* tend to rely on this approach for their construction [56, 57].

- **Basis interpolation** [31–33]. Following [32], we consider the implementation proposed in [31] combined with the use of radial basis interpolation. The model reconstruction reads

$$\mathbf{u}^{\text{proj}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \quad \text{with} \quad \mathbb{V}_{\boldsymbol{\mu}} = \mathcal{I}(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_c, \mathbb{V}_1, \dots, \mathbb{V}_c)(\boldsymbol{\mu}),$$

  where $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_c \in \Theta$ are the interpolation points, $\mathbb{V}_1, \dots, \mathbb{V}_c$ are a set of pre-computed local basis (each being representative of the submanifold associated to the corresponding $\boldsymbol{\mu}_i$), and $\mathcal{I}$ encapsulates the interpolation routine, which is carried out over the Grassmann manifold. For a precise definition of $\mathcal{I}$, we refer the interested reader to Appendix 8.2. Notice the absence of the Gramian matrix $\mathbb{G}$: consistently with the literature, the algorithm is implemented using the Euclidean metric for the state space. It is also worth remarking that this procedure cannot be implemented using the same training set used for the other approaches. In fact, one first needs to select c interpolation points $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_c$, and then, for each $i = 1, \dots, c$,

  - Sample $M > n$ random values of the remaining parameters, $\boldsymbol{\nu}_{i,1}, \dots \boldsymbol{\nu}_{i,M}$.
  - Solve the FOM to generate $\mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_{i,1}}, \dots, \mathbf{u}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_{i,M}}$.
  - Use those M simulations to extract the basis $\mathbb{V}_i$ via classical POD.

  In our experiments, we shall choose c and M such that the total number of FOM simulations is larger than or equal to the one used for the other approaches. Finally, in order to select the interpolation points, we use structured grids if $p \leq 3$ and random sampling if $p > 3$.
- **Autoencoders** [20, 21, 25]. In this case, the formula is just

$$\mathbf{u}^{\text{proj}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \Psi(\Psi'(\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}})),$$

  where $\Psi' : \mathbb{R}^{N_h} \to \mathbb{R}^n$ and $\Psi : \mathbb{R}^n \to \mathbb{R}^{N_h}$ are the encoder and decoder networks, respectively. To foster interpretability and provide a meaningful comparison, we shall construct these models following the same ideas adopted for the design of DOD architectures: in particular, we shall rely on POD-enhanced autoencoders [47, 58], thus leveraging the existence of the ambient space $\mathbb{A}$. In other words, we let

$$\Psi'(\mathbf{u}) = \psi'(\mathbb{A}^{\top}\mathbf{u}) \quad \text{and} \quad \Psi(\mathbf{c}) = \mathbb{A}\psi(\mathbf{c}),$$

  where $\psi' : \mathbb{R}^{N_A} \to \mathbb{R}^n$ and $\psi : \mathbb{R}^n \to \mathbb{R}^{N_A}$ are the trainable parts of the two architectures, respectively.

We conduct the analysis as follows. First, we compare DOD, POD, and AE for varying n, so as to better understand how the reduced dimension impacts the projection error. Then, we fix a latent dimension n and assess the performances of the DOD in comparison with the basis interpolation method and local POD.

All the code was implemented in Python 3 using the *dlroms* library [59], a Python package that relies on FEniCS and Pytorch to construct deep learning-based ROMs. The *dlroms* package is freely available on Github at https://github.com/NicolaRFranco/dlroms. All deep learning models were trained offline and evaluated online using a Tesla V100-PCIE-32GB GPU accelerator.

4.1 Stationary Navier–Stokes flow around a parametrized obstacle

To start, we consider the model problem discussed in Section 2.1, concerning a steady fluid flow around an obstacle. For better readability, we take the opportunity to restate the problem, specifying the governing equations and their parameterization. Specifically, we consider a 2D fluid flow modeled by the following parametrized Navier–Stokes equations,

$$\begin{cases} -\epsilon \Delta \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla q = 0 & \text{in } \Omega_{\boldsymbol{\mu}}, \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega_{\boldsymbol{\mu}}, \\ \boldsymbol{u} = \mathbf{g}_{\boldsymbol{\nu}} & \text{on } \Gamma_{\text{in}}, \\ \boldsymbol{u} = 0 & \text{on } \partial\Omega_{\boldsymbol{\mu}} \setminus (\Gamma_{\text{in}} \cup \Gamma_{\text{out}}), \\ q = 0 & \text{on } \Gamma_{\text{out}} \end{cases} \tag{16}$$

where $\Omega_{\boldsymbol{\mu}} := (0,1)^2 \setminus O_{\boldsymbol{\mu}}$ is a parameter-dependent domain, obtained by removing an almond-shaped object, $O_{\boldsymbol{\mu}}$, from the unit square (see Fig. 1). We focus our attention on the parameters-to-velocity map,

$$(\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \boldsymbol{u}.$$

Here, $\boldsymbol{\mu} = [\theta, x_0, y_0]$ is a vector parametrizing the center of the obstacle, (x_0, y_0), and its angle of rotation, θ. To ensure that the obstacle $O_{\boldsymbol{\mu}}$ always lies within the unit square, we let $\boldsymbol{\mu} \in \Theta := [0, 2\pi] \times [0.25, 0.75]^2$. The other parameters, $\boldsymbol{\nu} = [\alpha, \beta] \in \Theta' := [0, 10]^2$, instead, parametrize the inflow condition as

$$\mathbf{g}_{\boldsymbol{\nu}}(x, y) = y(1-y)\left(\alpha e^{-100(y-0.25)^2} + \beta e^{-100(y-0.75)^2}\right)^{1/2},$$

that is, by combining the contribution of two jet flows: one coming from the bottom (centered at $y = 0.25$), whose strength is determined by α, and one concentrated at the top ($y = 0.75$), whose intensity depends on β. For simplicity, we fix the viscosity coefficient to $\epsilon := 5 \cdot 10^{-3}$. We equip both parameter spaces, Θ and Θ', with a uniform probability distribution.

Notice that, here, the parametric dependence of the geometry poses an additional challenge in the construction of a ROM for Eq. (16) stemming from the difficulty in comparing different problem instances. As we mentioned in Section 2.1, to simplify

the analysis and focus on the phenomenon of interest—that is, the slow decay in the Kolmogorov n-width—we circumvent this issue by extending all PDE solutions to the whole unit square $\Omega = (0, 1)^2$ and then interpolating them over a common finite element space (see also Remark 6 for a deeper discussion on the matter). Precisely, for each parametric configuration $(\boldsymbol{\mu}, \boldsymbol{\nu}) \in \Theta \times \Theta'$, (i) we introduce a suitable mesh over $\Omega_{\boldsymbol{\mu}}$, tailored for the given geometry; (ii) we solve (16) by discretizing the function spaces for the pressure and velocity fields using a stable finite element pair based on mini-elements (continuous P1 elements for q, and P1-Bubble vector elements $\boldsymbol{u}$, i.e., linear velocity shape functions enriched with an internal bubble function that vanishes on element boundaries to ensure stability); (iii) we extend $\boldsymbol{u}$ to Ω by setting it equal to zero inside the obstacle; and (iv) we interpolate $\boldsymbol{u}$ over a predefined finite element space $V_h \cong \mathbb{R}^{N_h}$, consisting of P1-Bubble vector elements defined over a structured triangular grid of stepsize $h = \sqrt{2}/50$. This allows us to represent all velocity fields within a common state space, and thus treat the FOM as a map of the form

$$(\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^{N_h},$$

where $N_h = 15202$ are the dof in the P1-Bubble space of vector fields, $V_h \subset H^1(\Omega) \times H^1(\Omega)$. We exploit the FOM to sample 1500 random solutions, 1350 for training, and 150 for testing. In order to construct the DOD projector, we introduce an ambient space of dimension $N_A = 300$ (average projection error over the test set: 0.87%). To construct the seed and the root modules, instead, we use a collection of dense architectures whose hyperparameters are reported in Table 1. Additional details concerning the other benchmark models (namely, the autoencoders) can be found in Appendix 9.

Results are in Figs. 3 and 4, and in Tables 2, 3, 4, and 5. As we can appreciate from Fig. 4 (left panel), for any fixed latent dimension n, the DOD approach emerges by far as the

Table 1 General DOD architecture for the Navier–Stokes case study, Section 4.1. All architectures employ the 0.1-leakyReLU activation at the *internal* layers. The notation $a \mapsto b$ denotes a dense layer from $\mathbb{R}^a$ to $\mathbb{R}^b$; longer sequences indicate a composition of multiple layers. The number of root modules depends on the DOD dimension, n. Here, $\overset{*}{\mapsto}$ denotes a non-learnable feature layer that acts as $[\theta, x_0, y_0] \overset{*}{\mapsto} [\cos 4\theta, \sin 4\theta, x_0, y_0]$, which we use to enforce rotational symmetry

Component	Specifics	Terminal activation
Seed	$p \overset{*}{\mapsto} 4 \mapsto 50$	0.1-leakyReLU
Root	$50 \mapsto 50 \mapsto N_A$	-
Orth	Reduced QR	-

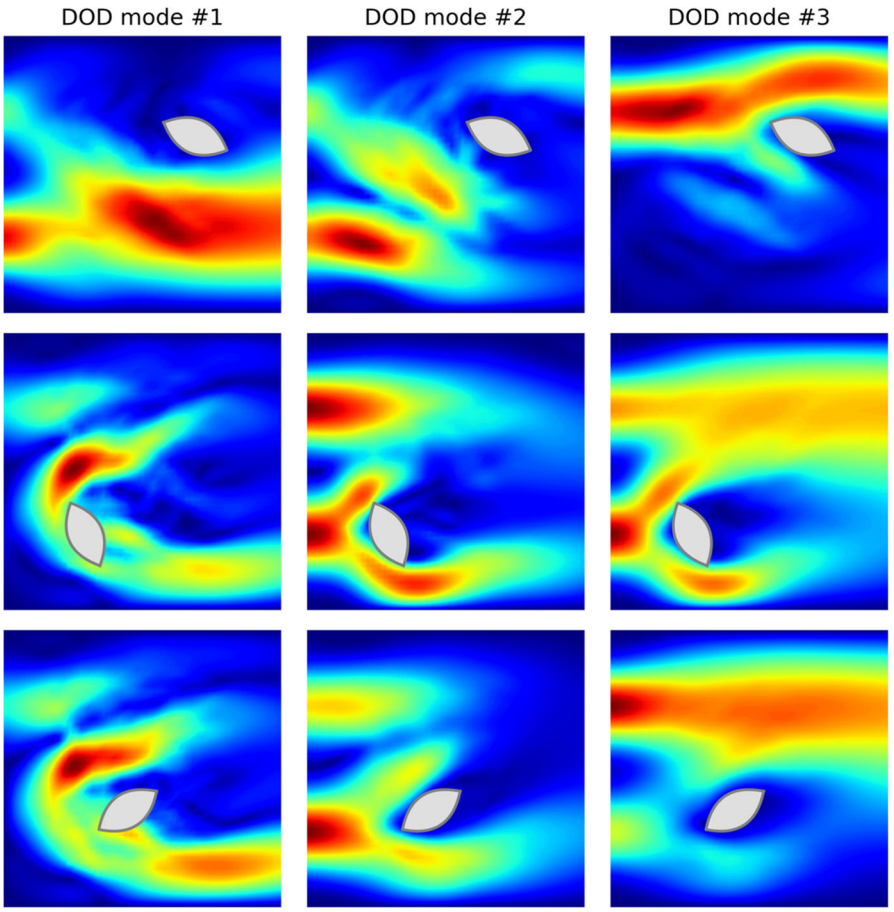

Fig. 3 DOD basis for different obstacle configurations in the Navier–Stokes example (Section 4.1). Each row refers to a different value of $\boldsymbol{\mu} = [\theta, x_0, y_0]$, while each column represents a DOD mode (here, $n = 3$). NB: For this case study, each DOD mode is actually a vector field $\mathbb{V}^j_{\boldsymbol{\mu}}\Omega_e \rightarrow \mathbb{R}^2$. However, to enhance readability, we are only plotting their magnitudes, $|\mathbb{V}^j_{\boldsymbol{\mu}}|$

best dimensionality reduction technique, reporting errors that are 3 to 4 times smaller than those achieved by POD and autoencoders. In turn, this results in a significant gain in terms of compression rate: notice, for instance, that 4 DOD modes can provide the same information as 27 POD modes. As the reduced dimension increases, we also observe an increased volatility of the DOD basis, with adaptivity scores ranging from 0.6 to 0.9 (cf. Fig. 4, right panel).

This adaptivity is also clearly depicted in Fig. 3, where we see how each DOD mode changes according to the position and the orientation of the obstacle. Interestingly, we also note that each DOD mode reflects different features of the problem. For instance, the third mode appears to focus on capturing the interaction between the obstacle and the jet flow at the top, while the first mode is more associated with the flow coming

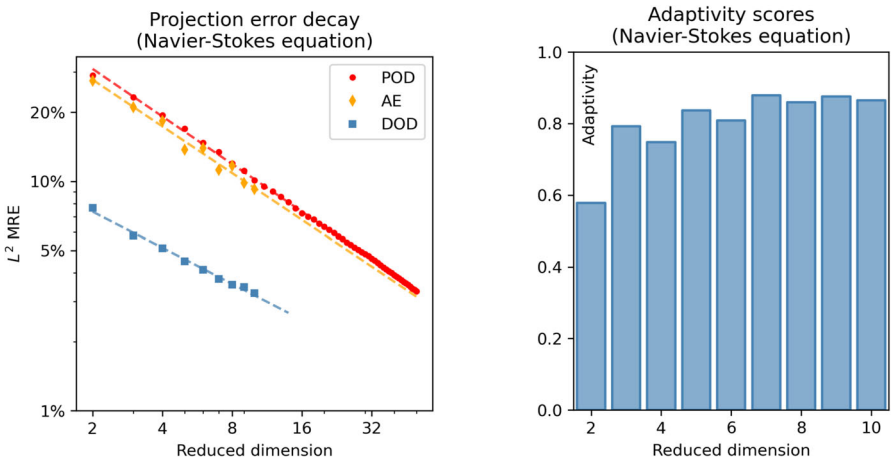

Fig. 4 Left: comparison between DOD and global dimensionality reduction strategies for the Navier–Stokes example (Section 4.1). Right: adaptivity of the DOD for different reduced dimensions n. Scores are defined as in Eq. (13)

Table 2 Comparison between DOD and other adaptive methods for the Navier–Stokes case study (Section 4.1). All models use $n = 4$ modes. c = number of clusters for Local POD. Projection error = MRPE (cf. Equation (15))

Reduction method	DOD	Local POD			Basis interpolation
		$c = 4$	$c = 16$	$c = 64$	
Projection error	5.109%	14.488%	10.872%	8.825%	10.878%

Table 3 General DOD architecture for the Eikonal Equation case study (Section 4.2). Table entries read as in Table 1. All architectures employ the 0.1-leakyReLU activation at the *internal* layers

Component	Specifics	Terminal activation
Seed	$p \mapsto 30 \mapsto 50$	0.1-leakyReLU
Root	$50 \mapsto 100 \mapsto N_A$	-
Orth	Reduced QR	-

Table 4 DOD in comparison with other adaptive methods for the Eikonal equation (Section 4.2). All models use $n = 6$ modes. c = num. of clusters. Projection error = MRPE (cf. Eq. (15))

Reduction method	DOD	Local POD			Basis interpolation
		$c = 4$	$c = 16$	$c = 64$	
Projection error	3.066%	6.495%	6.049%	6.006%	10.698%

Table 5 DOD architectures selected for model order reduction (see Section 6). Here, $p_{\text{tot}} := p + p'$ is the total number of parameters in the PDE models. Wall times refer to a single forward pass $\boldsymbol{\mu} \mapsto \mathbb{V}_{\boldsymbol{\mu}}$

Case study	p_{tot}	p	N_h	n	Projection error	Adaptivity	Evaluation time
Navier–Stokes	5	3	15,202	4	5.109%	0.748	0.001547 s
Eikonal Eq	102	100	6183	6	3.066%	0.545	0.001568 s

from the bottom. In general, it is evident that, even for very small latent dimensions, $n = 3$, the DOD approach can provide very rich representations. On the contrary, dictionary-based approaches, such as local POD, fail in replicating such complexity, unless the number of clusters becomes extremely large.

This is clearly seen in Table 2. For instance, when comparing the two approaches for $n = 4$, extrapolating from the overall trend indicates that approximately $c \approx 1200$ clusters would be required to match the accuracy of the DOD basis, a conclusion that is clearly impractical. Notably, DOD outperforms the basis interpolation method as well. We speculate that this could be due to the rigidity of the sampling strategy used for the interpolation method. In order to implement the latter, in fact, we selected $c = 144$ collocation points in Θ using a uniform grid, and then, for each of them, sampled 10 random points in Θ', which resulted in 1440 FOM simulations. However, despite exceeding the size of the training set employed for the DOD, $N_{\text{train}} = 1350$, these simulations are not as rich in terms of overall content. Specifically, due to their nested structure, they are not as efficient (in terms of cost investment) in revealing the actual complexity of the solution manifold. In turn, the interpolation procedure ends up relying on a set of local bases that are not sufficiently representative of the system. These considerations seem to suggest that the main strength of the DOD approach lies in its capability of combining a continuously adaptive local perspective together with a flexible training procedure.

Before continuing, we conclude with a final note on the error decay. As seen in Fig. 4, although the DOD manages to reduce the errors significantly, it is not capable of recovering the fast decay rate that we saw in Fig. 1. In other words, using the notation in Assumptions A1-A2, the DOD succeeds in reducing the error constant, $C' \ll C$, but fails in improving its decay rate, $\beta \approx \alpha$. In our view, this could be due to our design choice of introducing an underlying ambient space. Notice, in fact, that $\mathbb{E}\left[\|\mathbf{u}_{\boldsymbol{\mu},\nu} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\nu}\|/\|\mathbf{u}_{\boldsymbol{\mu},\nu}\|\right] \geq \mathbb{E}\left[\|\mathbf{u}_{\boldsymbol{\mu},\nu} - \mathbb{A}\mathbb{A}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\nu}\|/\|\mathbf{u}_{\boldsymbol{\mu},\nu}\|\right]$, meaning that the accuracy of the DOD cannot improve indefinitely if we just increase the latent dimension n. Nonetheless, the presence of $\mathbb{A}$ remains fundamental for the scalability of the approach.

As a side note, it is also worth adding a comment on the error decay associated with the autoencoder architectures. In fact, we see that in this example, the latter offers a limited improvement over POD. We argue that this fact is primarily due to the chosen latent dimension ($n \leq 10$), which is slightly below the theoretical threshold required to fully exploit nonlinear compression for this problem: cf. [21, Theorem 3]. In addition, the autoencoders were built on top of the ambient representation, which may hinder performances in certain scenarios: see, e.g., [60] and the discussion therein. Larger architectures or alternative representations, possibly coupled with more extensive training data, could unlock stronger performance, but were not explored here in order to favor comparability with the other reduction techniques.

Remark 6 Developing ROMs to tackle problems in varying geometries is a very challenging task. Possible strategies to achieve this goal typically consist of (i) relying on a fictitious domain approach or on a suitable postprocessing routine that interpolates all PDE solutions over a common mesh [61, 62]; (ii) exploiting mesh deformation strategies [63, 64] and/or registration methods [65]; and (iii) leveraging local operations,

as in graph neural networks (GNNs) [66–68]. Here, we consider the simplest of these approaches—that is, the first one—in order to maintain our focus on our primary objective, i.e., developing an adaptive local basis capable of overcoming the Kolmogorov barrier. Clearly, integrating DOD with, e.g., GNNs, would be an interesting research direction, potentially leading to very powerful and flexible ROMs. However, given that the DOD approach is still in its early stages, we leave these considerations for future work.

4.2 Eikonal equation in a parametrized medium

For our second case study, we consider a different scenario where the parametric problem is characterized by a high-dimensional parameter space, specifically, $p \gg p' \geq 1$. In doing so, we take the chance to showcase how DOD can handle nonlinear PDEs defined on complicated domains. We consider, in fact, a parameter-dependent Eikonal equation,

$$|\nabla u| = s_{\boldsymbol{\mu},\boldsymbol{\nu}}^{-1}, \tag{17}$$

defined over a simplified cartography of the italian peninsula, complemented with an internal Dirichlet condition, $u(\mathbf{x}_0) = 0$, where $\mathbf{x}_0 \in \Omega$. We recall that, among other things, the Eikonal equation also constitutes a prototypical example of wave propagation: for instance, in seismology and geophysics, it is commonly employed for modeling travelling times of seismic waves through the Earth's subsurface [69, 70]. In these cases, $\mathbf{x}_0$ typically represents the epicenter's location, while $s_{\boldsymbol{\mu},\boldsymbol{\nu}}(\mathbf{x})$ models the propagation speed at $\mathbf{x} \in \Omega$, which depends on the material properties of the soil at $\mathbf{x}$. Then, $u(\mathbf{x}) \geq 0$ corresponds to the time required for the wave to travel from $\mathbf{x}_0$ to $\mathbf{x}$. For the case at hand, we let $\mathbf{x}_0 \in \Omega$ represent a fixed source situated approximately near the city of Urbino, in central Italy.

For our analysis, we consider a situation in which (17) depends on $p + p' = 102$ scalar parameters, $\boldsymbol{\mu} = [\eta_1, \ldots, \eta_{100}]$, $\boldsymbol{\nu} = [\alpha, \beta]$, which parametrize the speed of travel as

$$s_{\boldsymbol{\mu},\boldsymbol{\nu}}(\mathbf{x}) := \sigma\left(\alpha \sum_{j=1}^{p} \eta_j \xi_j(\mathbf{x})\right) + \beta. \tag{18}$$

Here, $\sigma(z) := e^x/(1 - e^x)$ is the sigmoid, whereas $\xi_1, \ldots, \xi_{100} \in L^2(\Omega)$ are suitable orthonormal modes defined over Ω. In practice, we construct the latter by truncating the Karhunen-Loeve expansion of a mean-zero Gaussian process with covariance kernel $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-10|\mathbf{x} - \mathbf{x}'|^2)$ (cf. [71]). In particular, for each $j = 1, \ldots, 100$, one has

$$\int_\Omega \kappa(\mathbf{x}, \mathbf{x}')\xi_j(\mathbf{x}')d\mathbf{x}' = \lambda_j \xi_j(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega$$

for a suitable $\lambda_j > 0$. We sort the indices such that $\lambda_{j+1} \geq \lambda_j$. With this notation, we define the parameter space $\Theta \times \Theta'$ by setting

$$\Theta := \prod_{j=1}^{100} \left[-\sqrt{4\lambda_j}, 4\sqrt{\lambda_j}\right], \qquad \Theta' := [0, 0.1] \times [0.5, 1.0],$$

so that $p = 100$ and $p' = 2$. Essentially, in Eq. (18), β prescribes the minimum traveling speed for the wave, whereas $\sigma(\alpha \sum_{j=1}^{p} \eta_j \xi_j)$ models spatial heterogeneity. To this end, the η_j coefficients serve to characterize the spatial distribution of the material properties, while α controls the squashing of the sigmoidal transformation. Intuitively, for fixed values of $\boldsymbol{\mu} = [\eta_1, \ldots, \eta_p]$, we expect that the solution to (17) does not change dramatically. Conversely, different values of the η_j's can significantly affect the behavior of the solution due to their interaction with the space variable $\mathbf{x}$. We equip Θ' with a uniform distribution, whereas we let each η_j follow (independently) a truncated Gaussian distribution, so that $\eta_j \sim \mathcal{N}(0, \lambda_j)$ approximately. As ground truth reference, we consider a FOM based on an iterative scheme [72] relying upon a finite element discretization with continuous P1 elements defined over a triangular mesh of stepsize $h \approx 0.0369$. The resulting FOM dimension is $N_h = 6183$. We exploit the FOM to sample 3000 random solutions, 2700 for training, and 300 for testing.

To construct the DOD projector, we rely on an ambient space of dimension $N_A = 200$ (average ambient error = 0.28%). The remaining parts of the architecture are as in Table 3. To ensure a proper comparison, the autoencoders are constructed similarly; see Appendix 9 for further details.

Results are reported in Figs. 5 and 6 and Tables 4 and 5.

As seen in Fig. 5 (left panel), for small latent dimensions n, the DOD approach emerges by far as the best dimensionality reduction technique, reporting errors that are 2 to 3 times smaller than those achieved by POD and autoencoders. The trend, however, changes progressively as n increases, with POD reporting a steeper and steeper decay. At the same time, the adaptivity of the DOD appears to be reaching a plateau (cf. right panel of Fig. 5). This suggests that the solutions to (17) can be expressed in terms of a small number of adaptive modes—those captured by the DOD—combined with a global set of high-frequency modes, which are, instead, shared in between different problem instances.

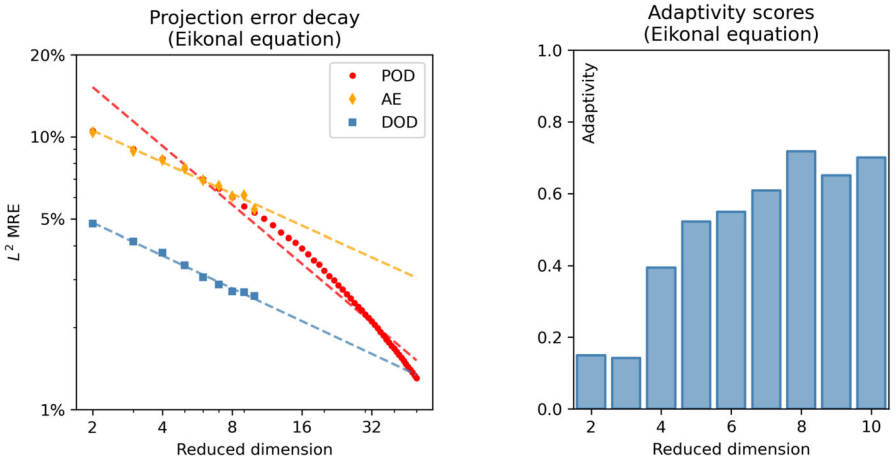

Fig. 5 Left: comparison between DOD and global dimensionality reduction strategies for the Eikonal equation (Section 4.2). Right: adaptivity of the DOD for different reduced dimensions n. Scores are defined as in Eq. (13)

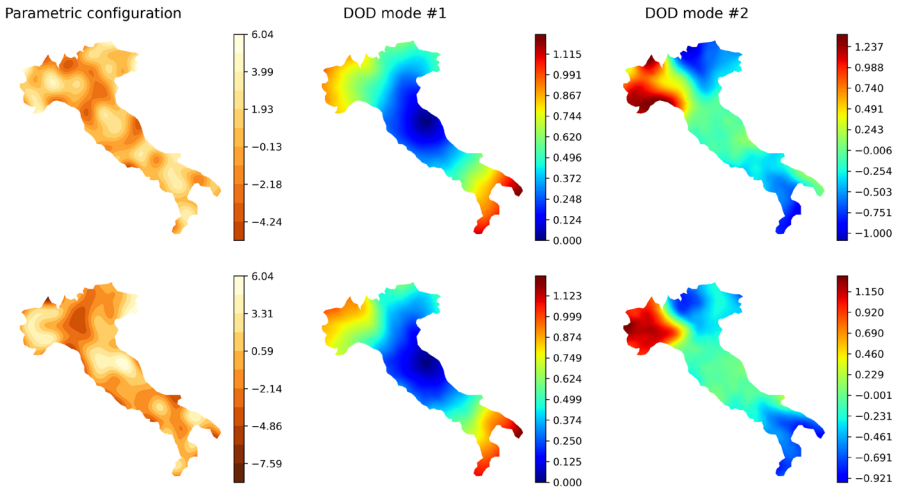

Fig. 6 DOD basis for two different realizations of the space-interacting parameter $\boldsymbol{\mu} \in \mathbb{R}^{100}$ appearing in the Eikonal equation example (Section 4.2). Each row refers to a different value of $\boldsymbol{\mu}$. 1st column: random field $g_{\boldsymbol{\mu}}$ defining the speed of travel map. 2nd–3rd column: DOD modes. Here, $n = 6$ but only two modes are shown to improve readability

As for the previous case study, at the latent level, the DOD basis manages to combine richness with interpretability. Notice, for instance, how the first DOD mode in Fig. 6 allows the model to distinguish between the central regions of Italy and the surrounding northern-south parts, coherently with the position of the epicenter $\mathbf{x}_0$. Conversely, the second mode captures differences between the east and west sides at the north of the peninsula. At the same time, these details are tuned depending on the underlying properties of the soil (left column in Fig. 6).

Notably, DOD also outperforms local POD and basis interpolation (see Table 4). The basis interpolation method, in fact, reports the worst performance, with an average test error above 10%. We believe this to be caused by the high dimensionality of the parameter space (recall that $p = 100$), which makes it difficult to capture a complex behavior with few collocation points. Finally, while the local POD method performs better compared to the interpolation method, the error decay in terms of the number of clusters, c, is extremely slow. Indeed, the reported trend would suggest picking more than 10^{11} clusters in order to match the accuracy of the DOD, which is clearly not feasible.

We also highlight that the advantages of the DOD approach also include a remarkable efficiency during the online phase. Indeed, after training, only a few milliseconds are required to compute the local basis $\mathbb{V}_{\boldsymbol{\mu}}$ for any given $\boldsymbol{\mu} \in \Theta$ (cf. Table 5). This is possible thanks to the efficiency of deep learning models, which do not require special routines aside from matrix–vector multiplication in order to be evaluated. In contrast, the basis interpolation method is considerably slower—by two orders of magnitude on our machine—as it requires repeated transformations between the Grassmann manifold and its tangent space, even when employed online (cf. Appendix 8.2).

Finally, concerning the training costs, we mention that all DOD architectures were trained on our machine in less than 3 min. In particular, in Fig. 7, the training time

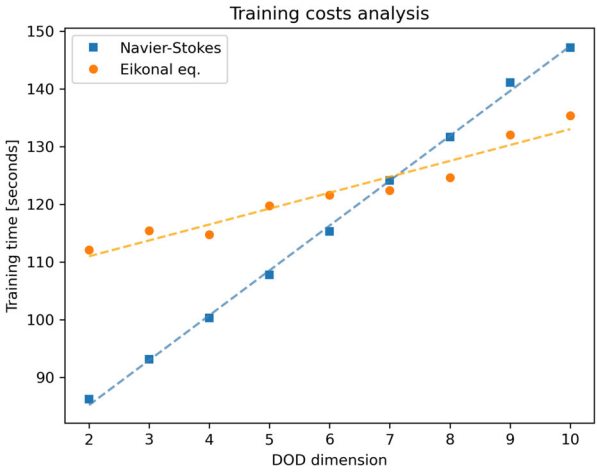

Fig. 7 DOD training time as a function of the reduced dimension n

is seen to scale linearly with the reduced dimension n, consistently with the fact that evaluating the loss function in Eq. (9) requires $O(N_{\text{train}} N_A n)$ operations. Interestingly, compared to the previous case study, the computational cost entailed by the training procedure is larger for small n and smaller for large n. This fact can be easily explained by recalling that, motivated by the larger number of problem parameters, in the Eikonal case study, compared to the Navier–Stokes example, we considered a larger sample size and a more complex DOD architecture. However, we also adopted a smaller number of ambient modes ($N_A = 200$ vs $N_A = 300$), which is what causes the two curves in Fig. 7 to diverge as n increases.

5 Deep orthogonal decomposition (II): reduced order modeling of parametrized PDEs

As we anticipated in Section 3, a DOD with n-modes allows us to reduce the complexity of the problem by shifting our attention from the parameter-to-solution map, $(\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^{N_h}$ to the parameter-to-DOD-coefficient map, i.e.

$$(\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}^{\top} \mathbb{G} \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \in \mathbb{R}^n, \tag{19}$$

Since $n \ll N_h$, learning the latter should be much easier when compared to the original problem. Before coming to our own proposal on *how* learn (19), it is worth making a few considerations of general interest. We summarize them below.

5.1 General considerations

Let $\phi : \mathbb{R}^p \times \mathbb{R}^{p'} \to \mathbb{R}^n$ be any algorithm of choice, be it intrusive or data-driven, that, given $(\boldsymbol{\mu}, \boldsymbol{\nu})$ seeks to approximate the corresponding DOD coefficient $\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}}$. The latter naturally gives rise to a DOD-based ROM via the ansatz

$$\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}^{\text{ROM}} := \mathbb{V}_{\boldsymbol{\mu}} \cdot \phi(\boldsymbol{\mu}, \boldsymbol{\nu}) \approx \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}},$$

where "·" emphasizes the presence of a matrix–vector multiplication. The quality of such an approximation will depend both on the DOD, $\mathbf{V}$, and on the parameter-to-coefficient algorithm, ϕ. To appreciate this, let

$$\mathcal{E}_A := \mathbb{E}^{1/2}_{\boldsymbol{\mu},\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2,$$

be the approximation error of the whole ROM, here measured according to a root-mean-square-error metric (RMSE). The two architectures, $\mathbf{V}$ and ϕ, are responsible for the following sources of error

$$\mathcal{E}_{\mathrm{DOD}} := \mathbb{E}^{1/2}_{\boldsymbol{\mu},\boldsymbol{\nu}} \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2, \qquad \mathcal{E}_{\mathrm{coeff}} := \mathbb{E}^{1/2}_{\boldsymbol{\mu},\boldsymbol{\nu}} |\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \phi(\boldsymbol{\mu},\boldsymbol{\nu})^2|,$$

respectively. Here, $\mathcal{E}_{\mathrm{DOD}}$ represents the DOD projection error, while $\mathcal{E}_{\mathrm{coeff}}$ reflects the quality of the approximation of the reduced problem (19): together, these two quantities uniquely characterize the general expressivity of the ROM. In fact, it is straightforward to see that the following identity holds.

Lemma 4 *For all DOD networks and all reduced algorithms, one has*

$$\mathcal{E}_A^2 = \mathcal{E}_{\mathrm{DOD}}^2 + \mathcal{E}_{\mathrm{coeff}}^2. \tag{20}$$

Proof We shall prove the stronger identity below,

$$\|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 = \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 + |\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \phi(\boldsymbol{\mu},\boldsymbol{\nu})|^2. \tag{21}$$

holding for all $\boldsymbol{\mu} \in \Theta$ and all $\boldsymbol{\nu} \in \Theta'$. Note, in fact, that (20) is just (21) in expectation. To see that (21) is valid, let $(\boldsymbol{\mu},\boldsymbol{\nu}) \in \Theta \times \Theta'$. Since $\mathbb{V}_{\boldsymbol{\mu}}$ is orthonormal, we have

$$|\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \phi(\boldsymbol{\mu},\boldsymbol{\nu})|^2 = \|\mathbb{V}_{\boldsymbol{\mu}}\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\phi(\boldsymbol{\mu},\boldsymbol{\nu})\|^2 = \|\mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2.$$

We now notice that, by definition,

$$\left(\mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\right) \in \mathrm{span}\left(\mathbb{V}_{\boldsymbol{\mu}}\right).$$

At the same time, by classical properties of linear projections,

$$\left(\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\right) \perp \mathrm{span}\left(\mathbb{V}_{\boldsymbol{\mu}}\right).$$

Then, by orthogonality,

$$\begin{aligned}\|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 + \|\mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 = \\ = \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \cancel{\mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}} + \cancel{\mathbb{V}_{\boldsymbol{\mu}}\mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 = \\ = \|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\mathrm{ROM}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2,\end{aligned}$$

as claimed. □

This splitting shows that errors in the approximation of the reduced map propagate through the DOD in a stable way, that is, an error of ϵ in the approximation of the reduced coefficients is reflected in a corresponding error of (at most) ϵ at the FOM level. We note that, typically, this property is exclusive to projection methods. Nonlinear techniques based on, e.g., autoencoders, instead, might suffer from error inflation. There, in fact, reduced coefficients are replaced by latent variables, and the lifting from $\mathbb{R}^n \to \mathbb{R}^{N_h}$ is obtained via a nonlinear decoder Ψ. Consequently, errors at the latent level can be bounded, at most, as

$$\|\Psi(\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}}) - \Psi(\phi(\boldsymbol{\mu}, \boldsymbol{\nu}))\| \leq L_\Psi |\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \phi(\boldsymbol{\mu}, \boldsymbol{\nu})|,$$

where L_Ψ is the Lipschitz constant of the decoder module. In particular, if $L_\Psi > 1$, errors may grow when passing through the decoder.

5.2 Learning the DOD coefficients

The hybrid nature of the DOD projector opens up a wide spectrum of possibilities for computing DOD coefficients, ranging from intrusive to data-driven approaches. For example, during the *online* phase, the DOD basis could be used to project and solve the governing equations, ultimately mimicking the idea underlying the POD-Galerkin ROMs. However, this approach would face major limitations when dealing, e.g., with nonlinear problems, as one would need to complement the DOD with a suitable hyper-reduction strategy, or when facing operators with nonaffine dependency on the parameters, as that would quickly increase the online computational cost (in fact, in order to project the equations, one would still need to assemble the FOM first). In light of this, and in order to be as general as possible, here, we shall focus on non-intrusive strategies. Given a candidate model class $\mathcal{C} \subset \{\tilde{\phi} : \mathbb{R}^p \times \mathbb{R}^{p'} \to \mathbb{R}^n\}$, which might consist of, e.g., neural network architectures, polynomials, or Gaussian processes, the idea is to construct the reduced algorithm ϕ via mean-square regression, namely

$$\phi := \underset{\tilde{\phi} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} |\mathbf{c}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i} - \tilde{\phi}(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)|^2,$$

where $\mathbf{c}_{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i}$ are defined according to (19).

In this work, we explore the use of neural network architectures, thus obtaining a ROM strategy that resembles the so-called POD-NN approach [73], except for the presence of the adaptive DOD basis. For this reason, we term this method DOD-NN.

The idea is to construct ϕ using a segregated architecture comprised of two sub-modules, ϕ_1 and ϕ_2, as to further differentiate between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. More precisely, we design ϕ as

$$\phi(\boldsymbol{\mu}, \boldsymbol{\nu}) := \operatorname{diag}\left[\phi_1(\boldsymbol{\mu})^\top \phi_2(\boldsymbol{\nu})\right],$$

where $\phi_1 : \mathbb{R}^p \to \mathbb{R}^{m \times n}$ and $\phi_2 : \mathbb{R}^{p'} \to \mathbb{R}^{m \times n}$ are two matrix-valued networks (implemented using classical architectures taking values in $\mathbb{R}^{mn}$, followed by a reshape layer). Mathematically speaking, this construction is equivalent to a separation of

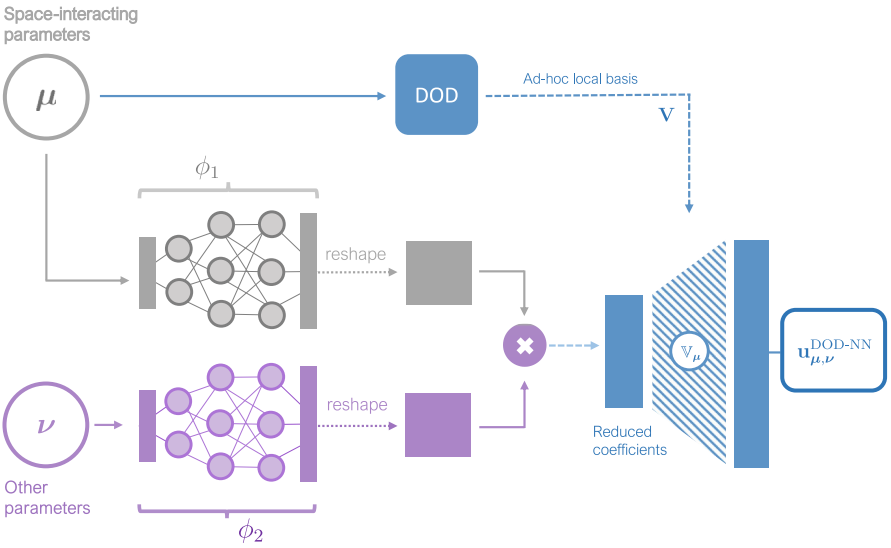

Fig. 8 Sketch of the DOD-NN approach (Section 5.2). See also Eq. (22)

variables approach, where a function of two variables, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, is expressed as the truncated sum (up to m terms) of simpler functions. Similar strategies have also been explored elsewhere, as in, e.g., DeepONets [16] and POD-MINN [74]. Here, the *diag* operator is merely a matter of mathematical notation: in practice, we refrain from calculating the matrix product $\phi_1(\boldsymbol{\mu})^\top \phi_2(\boldsymbol{\nu})$ and instead compute the Hadamard product of $\phi_1(\boldsymbol{\mu})$ and $\phi_2(\boldsymbol{\nu})$, followed by a columnwise summation.

With this setup, the DOD-NN ROM, $\mathbf{u}^{\text{DOD-NN}}_{\boldsymbol{\mu},\boldsymbol{\nu}} \approx \mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}$, can be summarized in formulas as

$$\mathbf{u}^{\text{DOD-NN}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}\phi(\boldsymbol{\mu},\boldsymbol{\nu}) = \mathbb{A}\text{ORTH}\left(\left[R_1(s_{\boldsymbol{\mu}}),\ldots,R_n(s_{\boldsymbol{\mu}})\right]\right)\cdot \text{diag}\left[\phi_1(\boldsymbol{\mu})^\top\phi_2(\boldsymbol{\nu})\right], \tag{22}$$

or, visually, as in Fig. 8. In general, the accuracy of the approximation will depend on the richness of the ambient space $\mathbb{A}$, the expressivity of the inner DOD module, $\tilde{\mathbb{V}}$, and the quality of the parameter-to-DOD-coefficient approximation, ϕ. In fact, it is straightforward to see that, as a direct consequence of Lemma 2 and Lemma 4, the following error decomposition formula is given.

Corollary 1 *Let* $(\boldsymbol{\mu},\boldsymbol{\nu}) \mapsto \mathbf{u}^{\text{DOD-NN}}_{\boldsymbol{\mu},\boldsymbol{\nu}}$ *be a* DOD-NN *reduced order model with ambient matrix* $\mathbb{A}$*, inner DOD module* $\tilde{\mathbb{V}}$*, and reduced network* ϕ*. Then,*

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbf{u}^{\text{DOD-NN}}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 = \;& \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \mathbb{A}\mathbb{A}^\top\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}\|^2 \\ &+ \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}|\mathbb{A}^\top\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \tilde{\mathbb{V}}_{\boldsymbol{\mu}}\tilde{\mathbb{V}}_{\boldsymbol{\mu}}^\top\mathbb{A}^\top\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}|^2 \\ &+ \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\nu}}|\tilde{\mathbb{V}}_{\boldsymbol{\mu}}^\top\mathbb{A}^\top\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \boldsymbol{\phi}(\boldsymbol{\mu},\boldsymbol{\nu})|^2,\end{aligned} \tag{23}$$

where we recall that $\mathbf{u}^{\text{DOD-NN}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}\phi(\boldsymbol{\mu},\boldsymbol{\nu})$ *with* $\mathbb{V}_{\boldsymbol{\mu}} := \mathbb{A}\tilde{\mathbb{V}}_{\boldsymbol{\mu}}$. *The three terms at the right-hand-side of* (23) *are the (i) ambient error, the (ii) intrinsic DOD projection error, and the (iii) coefficients error, respectively.*

Remark 7 Once a DOD architecture has been trained, replacing the orthonormalization block, ORTH, with a different one has no effect on the projection error. For instance, switching from a reduced QR algorithm to a Gram-Schmidt routine—and vice versa—has no impact on the accuracy of the DOD projection (assuming n is small and no numerical instabilities are introduced). In fact, as we noted in Section 3.3, the projection error depends only on the underlying subspace. In particular, given any rotation matrix $\mathbb{B} \in \mathbb{R}^{n\times n}$, replacing the output of the inner DOD module with $\tilde{\mathbb{V}}_{\boldsymbol{\mu}}\mathbb{B}$ results in the same reconstruction accuracy. Empirically, however, this modification appears to be relevant when training the final component of the DOD-NN. Notice, in fact, that introducing such rotation modifies the reduced representation, transforming the DOD coefficients from $\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}}$ to $\mathbb{B}^\top\mathbf{c}_{\boldsymbol{\mu},\boldsymbol{\nu}}$. This can directly impact the training dynamics of the reduced network ϕ, either hindering or improving convergence. For instance, in our experiments, flipping the signs of the DOD coefficients using a diagonal matrix $\mathbb{B}$ facilitated the training of ϕ, resulting in faster convergence. This can be explained by the fact that our architectures relied on the LeakyReLU activation function, which is inherently asymmetric and thus sensitive to sign changes, particularly during backpropagation.

6 Numerical experiments (II): reduced order modeling of parametrized PDEs

We are now ready to extend the analysis presented in Section 4, originally devoted to the sole dimensionality reduction, by considering the application of the DOD-NN strategy for model order reduction. To do so, we shall consider the same case studies discussed in Section 4 and, for each of them, proceed as follows.

First, we fix a reduced dimension n and a corresponding DOD network. In doing so, we shall opt for a suitable compromise between reconstruction accuracy, dimensionality reduction, and model volatility. Then, following the ideas presented in Section 5.2, we implement and train a reduced network ϕ. To do so, we rely on the same training data used for the DOD. Finally, we quantify the quality of the approximation by computing an empirical test error, calculated as

$$\text{MRE} := \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{\|\mathbf{u}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i} - \mathbf{u}^{\text{DOD}}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i}\|}{\|\mathbf{u}_{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i}\|},$$

where we recall that $\mathbf{u}^{\text{DOD}}_{\boldsymbol{\mu},\boldsymbol{\nu}} := \mathbb{V}_{\boldsymbol{\mu}}\phi(\boldsymbol{\mu},\boldsymbol{\nu})$, whereas $\{\check{\boldsymbol{\mu}}_i,\check{\boldsymbol{\nu}}_i\}_{i=1}^{N_{\text{test}}}$ are a collection of randomly sampled parameter configurations, drawn independently from the training set.

To better understand and appreciate the capabilities of the proposed approach, we also compare the performance of DOD-NN with two benchmark ROMs. In order to make the comparison as meaningful as possible, we first note the following. By construction,

$$\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}^{\mathrm{DOD}} \in \mathrm{span}(\mathbb{A})$$

for all $\boldsymbol{\mu}$ and all $\boldsymbol{\nu}$: that is, the outputs of a DOD-NN module are always elements of the ambient space. In particular, the inner module of the DOD-NN architecture,

$$(\boldsymbol{\mu}, \boldsymbol{\nu}) \mapsto \tilde{\mathbb{V}}_{\boldsymbol{\mu}} \boldsymbol{\phi}(\boldsymbol{\mu}, \boldsymbol{\nu}),$$

can be interpreted as approximating the parameter-to-ambient-coefficients map. In light of this, it is perfectly reasonable to ask whether a direct approximation would provide better results, i.e., with the ansatz,

$$\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} \approx \mathbb{A}\Phi(\boldsymbol{\mu}, \boldsymbol{\nu}) \tag{24}$$

where $\Phi : \mathbb{R}^p \times \mathbb{R}^{p'} \to \mathbb{R}^{N_A}$ is some neural network model. If so, the complex structure of the DOD-NN would be unmotivated, raising significant questions about the overall approach. Because of this, we believe this comparison to be highly valuable and of general interest. Notice also that Eq. (24) can be interpreted as a POD-NN model with $\mathbb{A}$ as the POD matrix; however, we refrain from using this terminology here, as that might generate confusion regarding the underlying latent dimension. The latter, in fact, would be n for DOD-NN but N_A for the surrogate model in (24). In view of these facts, we propose the following benchmark models.

- **Benchmark 1 (Standard $\mathbb{A}$-NN)**. Starting from Eq. (24), we design ϕ_{POD} as a classical DNN, taking as input the stacked vector of parameters $[\boldsymbol{\mu}, \boldsymbol{\nu}] \in \mathbb{R}^{p+p'}$.
- **Benchmark 2 (Segregated $\mathbb{A}$-NN)**. Here, we still rely on Eq. (24) but adopt a segregated architecture, $\Phi(\boldsymbol{\mu}, \boldsymbol{\nu}) = \mathrm{diag}\left[\Phi_1(\boldsymbol{\mu})^\top \Phi_2(\boldsymbol{\nu})\right]$, thus mimicking the idea in the DOD-NN approach.

Essentially, both approaches result in ROMs that approximate the solutions to the PDE by performing a suitable interpolation of the ambient coefficients. In both cases, in fact, we train the benchmark models by minimizing the mean square error associated with the discrepancy $|\mathbb{A}^\top \mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}} - \Phi(\boldsymbol{\mu}, \boldsymbol{\nu})|$. Furthermore, in order to make the comparison as fair as possible, in both cases, we design Φ to have the same complexity as the overall DOD-NN module. That is, we shall choose the number of layers and neurons so that Φ has approximately the same number of trainable parameters as the entire DOD-NN, including those of ϕ and $\tilde{\mathbb{V}}$.

6.1 Results

Table 5 contains the main information on the DOD architectures selected for the model order reduction phase. In general, we have opted for those architectures showing

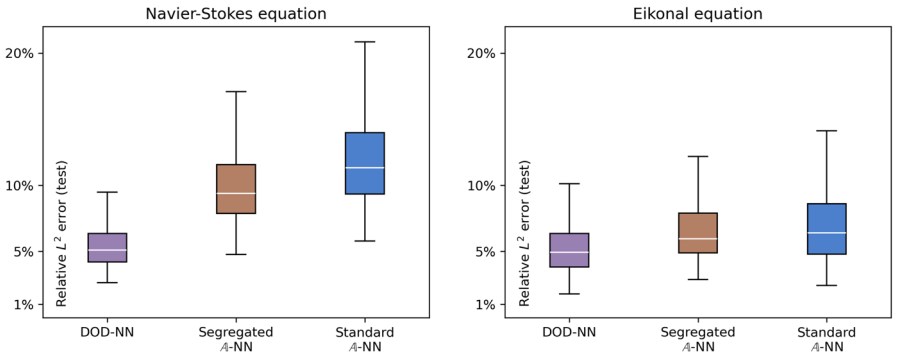

Fig. 9 Relative errors (test set) of DOD-NN and its competitors for the two case studies discussed in Section 4

a satisfactory accuracy but also moderate volatility. In fact, a higher variability of the DOD basis is typically reflected in a higher volatility of the DOD coefficients, $\mathbf{c}_{\boldsymbol{\mu}} = \mathbb{V}_{\boldsymbol{\mu}}^{\top}\mathbb{G}\mathbf{u}_{\boldsymbol{\mu},\boldsymbol{\nu}}$; thus, less volatile models are likely to yield representations that are simpler to learn.

A global overview of the final results is reported in Table 6. In particular, the DOD-NN approach consistently outperforms the benchmark models, at times exhibiting twice the accuracy. Overall, given the inherently data-driven nature of DOD-NN, its performance can be considered satisfactory, with average relative errors consistently below 6%. More precisely, as illustrated in Fig. 9, when evaluated on unseen problem parameters, DOD-NN generally yields solutions with relative errors in the 3%–7% range, while the benchmark models typically produce less accurate approximations.

Interestingly, the superiority of DOD-NN is not only quantitative but also qualitative, as seen in Figs. 10 and 11. There, we compared FOM solutions, DOD-NN

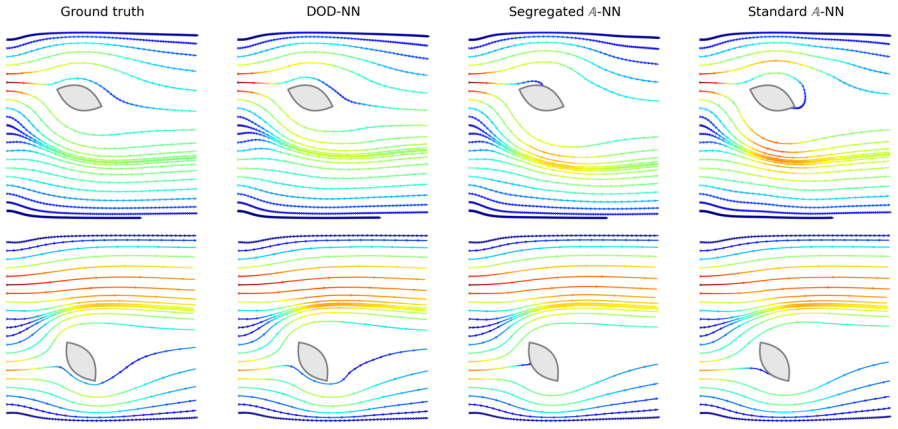

Fig. 10 Streamlines of the velocity fields for the Navier–Stokes example: FOM solutions vs DOD-NN approximations and benchmark outputs for two unseen configurations of the model parameters $(\boldsymbol{\mu}, \boldsymbol{\nu})$. Velocity magnitude varies with color (lower values in blue, higher values in red)

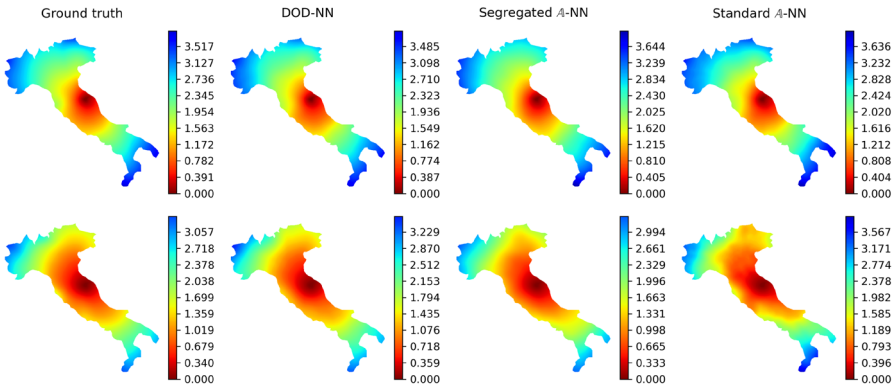

Fig. 11 Comparison between FOM solution, DOD-NN approximation, and benchmark ROMs for two unseen configurations of the model parameters appearing in the Eikonal equation example

approximations, and benchmark outputs for unseen values of the model parameters. We notice that the DOD-NN surrogate is much more aware of the space-interacting parameters, effectively capturing their effect over the global solution field. On the contrary, both benchmark models seem more likely to yield unphysical results: see, for example, Fig. 10, where both ROMs erroneously predict a fluid flow entering the obstacle.

Finally, it is worth emphasizing that these performances were achieved together with a significant compression of the FOM dimension. In fact, the proposed DOD-based ROMs only used $n = 4$ and $n = 6$ latent variables to represent the PDE solutions in the two case studies, resulting in a compression rate of 99.97% and 99.90%, respectively. For comparison, any POD-based ROM operating at the same latent dimension is guaranteed to yield significantly worse results (see the last row of Table 6). This includes reduced basis methods, such as POD-Galerkin [15], but also data-driven techniques such as POD-NN, POD-GPR [75], and POD-DeepONet [48]. In fact, the

Table 6 ROM performances (average relative L^2 error) for the two case studies presented in Section 4. 1st row: DOD-NN method (cf. Section 5). 2nd-3rd row: benchmark models using the ansatz in Eq. (24). 4th row: lower bound for POD-NN, POD-GPR, and POD-DeepONet assuming a number of spatial modes equal to that of DOD-NN

ROM	Navier–Stokes	Eikonal Eq.
DOD-NN	5.92%	5.39%
Standard $\mathbb{A}$-NN	12.37%	7.04%
Segregated $\mathbb{A}$-NN	10.13%	6.56%
POD-NN, POD-GPR, POD-DeepONet	$\geq$ 19.44%	$\geq$ 7.01%

accuracy of these approaches is confined by the projection error of the POD itself (compare the lower bounds in Table 6 with the values reported in Figs. 4 and 5.)

Remark 8 Our results were obtained by training the different ROMs over several thousand FOM simulations. It is then clear that such computational cost, although limited to the offline stage, is only justified in certain circumstances. A notable example is uncertainty quantification, where multi-fidelity estimators leveraging both high- and low-fidelity evaluations are commonly exploited to achieve a desired accuracy within a fixed computational budget (see, e.g., [76, 77]). In that context, if the ROM is sufficiently inexpensive and moderately accurate, one can effectively prove that training and using the ROM is computationally more efficient than relying exclusively on the FOM. Similar considerations apply to optimal control, inverse problems, and digital-twin settings, where ROMs can be used to accelerate exploration and provide effective initial guesses for FOM-based refinement. It is also important to emphasize that the target accuracy of a ROM depends on its intended use. In this work, errors of approximately 5% were considered acceptable given that the ROM is not meant to fully replace the FOM but rather to enable otherwise intractable many-query or real-time computations. Nonetheless, improving accuracy is certainly possible, for instance, by enlarging the training dataset and optimizing the model complexity, as supported by the theory [78, 79]. In our tests, particularly in the Eikonal equation, the generalization gap between training (3.4%) and testing (5.4%) errors suggests that increasing the amount of training data would be the most direct way to enhance predictive performance. Recall in fact that, in that case, DOD-NN only had access to 2700 samples during training, spread across a 102-dimensional space, which makes the regression problem fairly challenging. In general, necessitating of hundreds to thousands of samples is very common when dealing with time-independent problems, where each FOM simulation produces a single snapshot and thus yields a limited amount of information.

7 Conclusions

We presented a novel approach for reduced order modeling of parametrized PDEs, termed DOD-NN, designed to overcome the difficulties posed by the slow decay of the Kolmogorov n-width in the context of problems characterized by the presence of space-interacting parameters. By combining an adaptive basis perspective together with deep neural network models, DOD-NN can be interpreted as a generalization of

Table 7 Training and evaluation times for DOD-NN. Training includes both the DOD and the additional NN. Online times refer to querying 100,000 different PDE solutions simultaneously. dof = degrees of freedom of the $DOD-NN$ = total number of trainable parameters in the DOD and the NN

Problem	dof	Training time	Eval. time (100,000 calls)
Navier–Stokes	84750	4 m 37.78 s	0.0029 s
Eikonal Eq	176880	2 m 51.61 s	0.0038 s

the POD-NN strategy, which makes the approach capable of tackling a broader class of parametric problems.

At its core, DOD-NN uses DOD, a newly proposed neural network architecture that approximates the solution manifold through a continuously adaptive local basis, for the sake of dimensionality reduction. Compared to other techniques, such as deep autoencoders, the DOD stands out for its ability in providing a tight error control during the decoding phase combined with a rich and highly interpretable latent space. This is possible thanks to the hybrid linear-nonlinear nature of the DOD, which makes the approach closely related to basis interpolation methods and library ROMs. In this sense, we consider DOD-NN as a new, valuable tool for domain practitioners, whose usage is particularly recommended as follows:

i) *The problem at hand features multiple parameters, $p_{tot} = p + p'$, some of which lead to a slow decay in the Kolmogorov n-width.* Our experiments show that including this knowledge during the dimensionality reduction process can provide improved performances compared to POD and autoencoders.
ii) *The parameters responsible of the above issue are $p > 1$.* This is the regime where our approach becomes a valuable alternative to basis interpolation methods. DOD, in fact, can handle an arbitrarily large number of parameters and does not require special sampling strategies. While this is not an intrinsic issue of classical interpolation strategies (see, e.g., [33]), our experiments suggest that a significant improvement can be achieved for certain classes of problems.
iii) *Capturing global features is a priority, and accuracies in the range of 1%–10% are considered satisfactory.* This is in fact the scenario where the DOD-NN yields the best compromise in terms of accuracy and computational efficiency. Applications of this kind include, e.g., forward and inverse uncertainty quantification tasks, where low-fidelity models can be extremely valuable. If finer scales need to be captured, instead, resorting to closure models can be a more favorable option.
iv) *Reducing the computational cost is crucial*. By relying on deep learning models, DOD-NN can produce new simulations in a few milliseconds. Furthermore, the speed-up is even more advantageous when multiple solutions are sought at the same time (cf. Table 7), thanks to the ability of neural networks to operate in batches;
v) *Interpretability constitutes an added value*. In some cases, being able to inspect the ROM can suggest new ways for improving the accuracy of the model, or simply provide additional insights on how (or why) a certain prediction was made.

While the present work focuses on the development and assessment of a purely data-driven, non-intrusive framework, it is important to acknowledge that such an approach inherently limits the model's robustness when operating outside the training distribution. In this sense, the proposed DOD-NN is designed to operate in the interpolation regime, where sufficient coverage of the parameter space can be ensured by the training data. Extending the methodology to extrapolative scenarios would require the incorporation of suitable inductive biases, such as physics-informed constraints,

energy-based regularization, or a transition toward intrusive formulations that explicitly embed the governing equations into the learning process, avenues that are currently beyond the scope of this work.

Another interesting research question concerns the adaptation of DOD-NN to the time-dependent framework. In this work, in fact, we focused exclusively on stationary PDEs. In principle, integrating time as an additional parameter (to be coupled with $\boldsymbol{\mu}$) should make the extension of our approach to time-dependent problems straightforward. In fact, by querying multiple solutions independently, one could potentially produce a significant number of full trajectories—with thousands of time-stamps each—in a few milliseconds (cf. Table 7). However, this approach would completely ignore the dynamical structure of such problems, severely limiting its predictive power in the extrapolation regime. In this sense, developing more advanced strategies that incorporate some of the properties that characterize dynamical systems, such as being Markovian or autoregressive, could be a more promising research direction. Indeed, both intrusive and non-intrusive methods based on time-adaptive basis are an active area of research: see, e.g., [39, 80].

Parallel to this, another interesting question would be to investigate suitable generalizations of the DOD capable of tackling PDEs with parameter-dependent spatial domains: see, e.g, the recent work in [81]. Right now, in our framework, this is only possible in tandem with fictitious domain or mesh deformation approaches. Radically different tools, such as graph neural networks, might instead offer valuable insights. Similarly, another open question is whether one can *discover* the decoupling of the parameter space from data, i.e., how to distinguish between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, without relying on prior knowledge, similarly to what happens with slow-fast decompositions [82]. Last but not least, an interesting perspective could be to integrate the DOD with a physics-based framework, e.g., by combining it with reduced basis methods. This will likely require introducing ad hoc projection strategies, such as tailored hyper-reduction algorithms, in order to make the approach computationally suitable for real-time applications. We leave all these considerations for future work.

Appendix 1. Technical details: ambient extraction and basis interpolation

A.1 Ambient space computation

The ambient space is computed by applying a $\mathbb{G}$-orthonormal generalization of the POD. The corresponding Algorithm, reported below, can be found in most textbooks on model order reduction: see, e.g., [15, Section 6.3.2] or [14, Section 3.2.1].

A.2 Basis interpolation with radial basis functions

Let $1 \leq n < N_h$, and let $\Theta \subset \mathbb{R}^p$. We are given $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_c \in \Theta$ and corresponding $\mathbb{V}_1, \dots \mathbb{V}_c \in \mathbb{R}^{N_h \times n}$. The basis interpolation method defines an interpolator $\mathcal{I} =$

Algorithm 3 Construction of the ambient space.

Input : Training simulations $[\mathbf{u}_1, \ldots, \mathbf{u}_{N_{\text{train}}}]$, Gram matrix $\mathbb{G}$, ambient dimension N_A.
Output: Ambient matrix $\mathbb{A}$.

// Preprocessing
$\mathbb{U} \leftarrow$ stack $[\mathbf{u}_1, \ldots, \mathbf{u}_{N_{\text{train}}}]$
$\mathbb{M} \leftarrow \mathbb{U}^\top \mathbb{G} \mathbb{U}$

// Eigenvalues and eigenvectors, with $\lambda_i \geq \lambda_{i+1}$
$\lambda_1, \ldots, \lambda_{N_{\text{train}}}$ and $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_{\text{train}}} \leftarrow \text{eig}(\mathbb{M})$

// Modes truncation
$\boldsymbol{\Lambda} \leftarrow \text{diag}(\lambda_1, \ldots, \lambda_{N_A})$
$\Xi \leftarrow$ stack $[\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_A}]$
$\mathbb{A} \leftarrow \mathbb{U} \Xi \boldsymbol{\Lambda}^{-1/2}$.
return $\mathbb{A}$

$\mathcal{I}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, \mathbb{V}_1, \ldots, \mathbb{V}_c) : \Theta \to \mathbb{R}^{N_h \times n}$ such that

$$\text{span}\, \mathcal{I}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, \mathbb{V}_1, \ldots, \mathbb{V}_c)(\boldsymbol{\mu}_i) = \text{span}\, \mathbb{V}_i.$$

Equivalently, if we denote by $\mathcal{G} := \mathcal{G}_n(\mathbb{R}^{N_h})$ the Grassmann manifold consisting of all subspaces of dimension n embedded in $\mathbb{R}^{N_h}$, then interpolator $\mathcal{I}$ ensures that each $\boldsymbol{\mu}_i$ is mapped onto a matrix indistinguishable from $\mathbb{V}_i$ in $\mathcal{G}$. We define the interpolator following [31], specifically relying upon radial basis interpolation as in [32]. The idea goes as follows. Let $O_n \subset \mathbb{R}^{N_h \times n}$ be the set of orthonormal matrices. Define the map $\text{Log}_{\mathcal{G}} : O_n \times O_n \to \mathbb{R}^{N_h \times n}$ as

$$\text{Log}_{\mathcal{G}}(\mathbb{A}, \mathbb{B}) = \mathbb{U} \tan^{-1}(\boldsymbol{\Sigma}), \mathbb{V}^\top,$$

where $\mathbb{U}, \boldsymbol{\Sigma}, \mathbb{V}^\top = \text{svd}\left[\left(\mathbb{I} - \tilde{\mathbb{A}}\tilde{\mathbb{A}}^\top\right)\tilde{\mathbb{B}}(\tilde{\mathbb{A}}^\top\tilde{\mathbb{B}})^{-1}\right]$ are computed using the SVD algorithm. Similarly, define $\text{Exp}_{\mathcal{G}} : O_n \times \mathbb{R}^{N_h \times n} \to O_n$ as

$$\text{Exp}_{\mathcal{G}}(\mathbb{A}, \boldsymbol{\Gamma}) = \left(\mathbb{V}^\top \cos(\boldsymbol{\Sigma}) + \mathbb{U} \sin(\boldsymbol{\Sigma})\right)\mathbb{A},$$

where $\mathbb{U}, \boldsymbol{\Sigma}, \mathbb{V}^\top = \text{svd}(\boldsymbol{\Gamma})$. It can be shown that, from the perspective of the Grassmann manifold, computing $\text{Log}_{\mathcal{G}}(\mathbb{A}, \mathbb{B})$ corresponds to projecting $\text{span}(\mathbb{B})$ onto the tangent space of the Grassmann manifold at the point $\text{span}(\mathbb{A}) \in \mathcal{G}$. Conversely, given an element $\boldsymbol{\Gamma}$ in the tangent space of $\mathcal{G}$ at $\text{span}(\mathbb{A})$, computing $\text{Exp}_{\mathcal{G}}(\mathbf{A}, \boldsymbol{\Gamma})$ corresponds to mapping $\boldsymbol{\Gamma}$ back on $\mathcal{G}$. In this sense, $\text{Log}_{\mathcal{G}}$ and $\text{Exp}_{\mathcal{G}}$ can be seen as the Riemannian logarithm and exponential over $\mathcal{G}$, respectively: cf. [31].

In the basis interpolation method, the maps $\mathrm{Log}_{\mathcal{G}}$ and $\mathrm{Exp}_{\mathcal{G}}$ are used to lift the interpolation problem from the Grassmann manifold onto the tangent space (which, being a linear vector space, is much easier to handle). Within the tangent space, the interpolation can be carried out using any algorithm of choice [83]; here, we use radial basis interpolation as in [32]. Given a radial basis function $\kappa : [0, +\infty) \to \mathbb{R}$, a set of vector inputs $\{\boldsymbol{a}_i\}_{i=1}^m$ and a set of scalar outputs $\{b_i\}_{i=1}^m$, we write

$$\mathsf{rbf}_\kappa(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m, b_1, \ldots, b_m) : \mathbb{R} \to \mathbb{R}$$

for the RBF interpolator fitted such that $\kappa\ \boldsymbol{a}_i \mapsto b_i$.

With this notation, the full algorithm of the basis interpolation method is outlined in Algorithm 4. In our experiments, we use a Gaussian kernel $\kappa(d) = e^{-\eta d^2}$ with $\eta > 0$ chosen such that the reconstruction error over the test set is minimized.

Algorithm 4 Basis interpolation on the Grassmann manifold using RBF.

Input : Interpolation points $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c \in \mathbb{R}^p$, corresponding basis $\mathbb{V}_1, \ldots \mathbb{V}_c \in \mathbb{R}^{N_h \times n}$, radial basis function $\kappa : [0, +\infty) \to \mathbb{R}$, evaluation point $\boldsymbol{\mu} \in \mathbb{R}^p$.

Output: Interpolated basis $\mathbb{V}_{\boldsymbol{\mu}} = \mathcal{I}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, \mathbb{V}_1, \ldots \mathbb{V}_c)(\boldsymbol{\mu})$.

// Closest interpolation point

$i_* \leftarrow \mathrm{argmin}_{i \in \{1,\ldots,c\}} |\boldsymbol{\mu}_i - \boldsymbol{\mu}|$

// Mapping onto tangent space

$[\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_c] \leftarrow [\mathrm{Log}_{\mathcal{G}}(\mathbb{V}_{i_*}, \mathbb{V}_1), \ldots, \mathrm{Log}_{\mathcal{G}}(\mathbb{V}_{i_*}, \mathbb{V}_c)]$

// Fit interpolators

for $k = 1, \ldots, N_h$ **do**
  for $j = 1, \ldots, n$ **do**
    $r_{k,j} \leftarrow \mathsf{rbf}_\kappa(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, \gamma_1^{k,j}, \ldots, \gamma_c^{k,j})$ where $\boldsymbol{\Gamma}_i = [\gamma_i^{k,j}]_{k,j}\ \forall i = 1, \ldots, c$
  end
end

// Evaluate interpolators

$\boldsymbol{\Gamma}_{\boldsymbol{\mu}} \leftarrow [r_{k,j}(\boldsymbol{\mu})]_{k,j}$

// Map back to the Grassmann manifold

$\mathbb{V}_{\boldsymbol{\mu}} \leftarrow \mathrm{Exp}_{\mathcal{G}}(\mathbb{V}_{i_*}, \boldsymbol{\Gamma}_{\boldsymbol{\mu}})$

return $\mathbb{V}_{\boldsymbol{\mu}}$

Appendix 2. Technical details: neural network architectures and sampling

We report below the architectures of the autoencoder models in Section 4, together with the neural network models employed in Section 6 (DOD-NN and POD-NN benchmarks). All architectures employ the 0.1-leakyReLU activation at the *internal* layers. NB: in the Navier–Stokes example, some architectures use a nonlearnable feature layer, denoted as $\overset{*}{\mapsto}$, which either acts as

$$[\theta, x_0, y_0, \alpha, \beta] \overset{*}{\mapsto} [\cos 4\theta, \sin 4\theta, x_0, y_0, \alpha, \beta]$$

or $[\theta, x_0, y_0] \overset{*}{\mapsto} [\cos 4\theta, \sin 4\theta, x_0, y_0]$, depending on the input size. As discussed in Section 4, the latter is used to enforce rotational symmetry. At the end of this section, we also report additional details on the sampling strategy adopted for the basis interpolation method (number of collocation points, c, and samples per location, M).

Table 8 Autoencoder architectures for the case studies in Sections 4.2 and 4.1

Case study	Component	Specifics	Terminal activation
Navier–Stokes	Encoder	$N_h \mapsto n$	0.1-leakyReLU
	Decoder	$n \mapsto 1000 \mapsto N_h$	-
Eikonal equation	Encoder	$N_h \mapsto n$	0.1-leakyReLU
	Decoder	$n \mapsto 50 \mapsto 50 \mapsto N_h$	-

Table 9 POD-NN network architectures (Benchmark 1)

Case study	Specifics	Terminal activation
Navier–Stokes	$(p + p') \overset{*}{\mapsto} (4 + p') \mapsto 150 \mapsto 150 \mapsto N_h$	-
Eikonal equation	$(p + p') \mapsto 200 \mapsto 200 \mapsto N_h$	-

Table 10 Table C.3: POD-NN network architectures (Benchmark 2)

Case study	Component	Specifics	Terminal activation
Navier–Stokes	ϕ_1	$p \overset{*}{\mapsto} 4 \mapsto 30 \mapsto 5 \times N_h$	0.1-leakyReLU
	ϕ_2	$p' \mapsto 30 \mapsto 5 \times N_h$	-
Eikonal equation	ϕ_1	$p \mapsto 20 \mapsto 20 \times N_h$	0.1-leakyReLU
	ϕ_2	$p' \mapsto 20 \mapsto 20 \times N_h$	-

Table 11 DOD-NN network architectures

Case study	Component	Specifics	Terminal activation
Navier–Stokes	ϕ_1	$p \overset{*}{\mapsto} 4 \mapsto 50 \mapsto 50 \mapsto 25 \times n$	0.1-leakyReLU
	ϕ_2	$p' \mapsto 50 \mapsto 25 \times n$	-
Eikonal equation	ϕ_1	$p \mapsto 25 \mapsto 10 \times n$	0.1-leakyReLU
	ϕ_2	$p' \mapsto 25 \mapsto 10 \times n$	-

Table 12 Sampling strategy for basis interpolation

Case study	Method	c	M
Navier–Stokes	Uniform grid	144	10
Eikonal equation	Random	270	10

Acknowledgements NRF, AM, and PZ are members of the Gruppo Nazionale per il Calcolo Scientifico (GNCS) of the Istituto Nazionale di Alta Matematica (INdAM), of which they acknowledge the support.

Author contribution Conceptualization: NRF, AM, PZ. Methodology: NRF, AM, PZ, JH. Formal analysis: NRF. Supervision: AM, PZ, JH. Funding acquisition: AM, PZ. Resources: PZ. Data curation: NRF. Writing (original draft): NRF. Writing (reviewing and editing): NRF, AM, PZ, JH.

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. NRF and AM have been supported by the Project "Reduced Order Modeling and Deep Learning for the real-time approximation of PDEs (DREAM)" (Starting Grant No. FIS00003154), funded by the Italian Science Fund (FIS) - Ministero dell'Università e della Ricerca.
NRF and PZ have been supported by the project Cal.Hub.Ria (Piano Operativo Salute, traiettoria 4), funded by MSAL. PZ acknowledges the support of the grant MUR PRIN 2022 No. 2022WKWZA8 "Immersed methods for multiscale and multiphysics problems (IMMEDIATE)" part of the Next Generation EU program. AM acknowledges the PRIN 2022 Project "Numerical approximation of uncertainty quantification problems for PDEs by multi-fidelity methods (UQ-FLY)" (No. 202222PACR), funded by the European Union - NextGenerationEU and the project FAIR (Future Artificial Intelligence Research), funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence). The present research is part of the activities of the project Dipartimento di Eccellenza 2023-2027, Department of Mathematics, Politecnico di Milano, funded by MUR.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Cao, L., O'Leary-Roseberry, T., Jha, P.K., Oden, J.T., Ghattas, O.: Residual-based error correction for neural operator accelerated infinite-dimensional Bayesian inverse problems. J. Comput. Phys. **486**, 112104 (2023). https://doi.org/10.1016/j.jcp.2023.112104
2. Benfenati, A., Causin, P., Quinteri, M.: A modular deep learning-based approach for diffuse optical tomography reconstruction (2024). arXiv preprint arXiv:2402.09277. https://doi.org/10.48550/arXiv.2402.09277
3. Lähivaara, T., Malehmir, A., Pasanen, A., Kärkkäinen, L., Huttunen, J.M., Hesthaven, J.S.: Estimation of groundwater storage from seismic data using deep learning. Geophys. Prospect. **67**(8), 2115–2126 (2019). https://doi.org/10.1111/1365-2478.12831
4. Mücke, N.T., Sanderse, B., Bohté, S.M., Oosterlee, C.W.: Markov chain generative adversarial neural networks for solving Bayesian inverse problems in physics applications. Comput. Math. Appl. **147**, 278–299 (2023). https://doi.org/10.1016/j.camwa.2023.07.028
5. Zabaras, N.: Solving stochastic inverse problems: a sparse grid collocation approach. In: Large-scale Inverse Problems and Quantification of Uncertainty, pp. 291–319. Wiley Online Library, Chichester, UK (2011). https://doi.org/10.1002/9780470685853
6. Bader, E., Kärcher, M., Grepl, M.A., Veroy, K.: Certified reduced basis methods for parametrized distributed elliptic optimal control problems with control constraints. SIAM J. Sci. Comput. **38**(6), 3921–3946 (2016). https://doi.org/10.1137/16M1059898
7. Kleikamp, H., Lazar, M., Molinari, C.: Be greedy and learn: efficient and certified algorithms for parametrized optimal control problems. ESAIM: Math. Model. Num. Anal. **59**(1), 291–330 (2025). https://doi.org/10.1051/m2an/2024074
8. Ravindran, S.S.: A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. Int. J. Numer. Meth. Fluids **34**(5), 425–448 (2000). https://doi.org/10.1002/1097-0363(20001115)34:5<425::AID-FLD67>3.0.CO;2-W
9. Strazzullo, M., Ballarin, F., Mosetti, R., Rozza, G.: Model reduction for parametrized optimal control problems in environmental marine sciences and engineering. SIAM J. Sci. Comput. **40**(4), 1055–1079 (2018). https://doi.org/10.1137/17M1150591
10. Cicci, L., Fresca, S., Guo, M., Manzoni, A., Zunino, P.: Uncertainty quantification for nonlinear solid mechanics using reduced order models with gaussian process regression. Comput. Math. Appl. **149**, 1–23 (2023). https://doi.org/10.1016/j.camwa.2023.08.016
11. Reiner, J., Linden, N., Vaziri, R., Zobeiry, N., Kramer, B.: Bayesian parameter estimation for the inclusion of uncertainty in progressive damage simulation of composites. Compos. Struct. **321**, 117257 (2023)
12. Vitullo, P., Franco, N.R., Zunino, P.: Deep learning enhanced cost-aware multi-fidelity uncertainty quantification of a computational model for radiotherapy. Found Data Sci. **7**(1), 386–417 (2025). https://doi.org/10.3934/fods.2024022
13. Zhu, Y., Zabaras, N.: Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. J. Comput. Phys. **366**, 415–447 (2018). https://doi.org/10.1016/j.jcp.2018.04.018
14. Hesthaven, J.S., Rozza, G., Stamm, B., et al.: Certified reduced basis methods for parametrized partial differential equations. Springer, Cham, Switzerland (2016). https://doi.org/10.1007/978-3-319-22470-1
15. Quarteroni, A., Manzoni, A., Negri, F.: Reduced basis methods for partial differential equations: an introduction. Springer, Cham, Switzerland (2016). https://doi.org/10.1007/978-3-319-15431-2
16. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. **3**(3), 218–229 (2021). https://doi.org/10.1038/s42256-021-00302-5
17. DeVore, R.A., Kyriazis, G., Leviatan, D., Tikhomirov, V.M.: Wavelet compression and nonlinear n-widths. Adv. Comput. Math. **1**(2), 197–214 (1993). https://doi.org/10.1007/BF02071385
18. Lee, K., Carlberg, K.T.: Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J. Comput. Phys. **404**, 108973 (2020). https://doi.org/10.1016/j.jcp.2019.108973
19. Cohen, A., DeVore, R.: Kolmogorov widths under holomorphic mappings. IMA J. Numer. Anal. **36**(1), 1–12 (2016). https://doi.org/10.1093/imanum/dru066

20. Fresca, S., Dede, L., Manzoni, A.: A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J. Sci. Comput. **87**(2), 1–36 (2021). https://doi.org/10.1007/s10915-021-01462-7
21. Franco, N., Manzoni, A., Zunino, P.: A deep learning approach to reduced order modelling of parameter dependent partial differential equations. Math. Comput. **92**(340), 483–524 (2023). https://doi.org/10.1090/mcom/3781
22. Ohlberger, M., Rave, S.: Reduced basis methods: success, limitations and future challenges, pp. 1–12. Publishing House of Slovak University of Technology in Bratislava, Bratislava, Slovakia (2016). http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/algoritmy/article/view/389
23. Apacoglu, B., Paksoy, A., Aradag, S.: CFD analysis and reduced order modeling of uncontrolled and controlled laminar flow over a circular cylinder. Eng. Appl. Comput. Fluid. Mechan. **5**(1), 67–82 (2011). https://doi.org/10.1080/19942060.2011.11015353
24. Lorenzi, S., Cammi, A., Luzzi, L., Rozza, G.: POD-Galerkin method for finite volume approximation of Navier-Stokes and RANS equations. Comput. Methods Appl. Mech. Eng. **311**, 151–179 (2016). https://doi.org/10.1016/j.cma.2016.08.006
25. Romor, F., Stabile, G., Rozza, G.: Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. J. Sci. Comput. **94**(3), 74 (2023). https://doi.org/10.1007/s10915-023-02128-2
26. Amsallem, D., Zahr, M.J., Farhat, C.: Nonlinear model order reduction based on local reduced-order bases. Int. J. Numer. Meth. Eng. **92**(10), 891–916 (2012). https://doi.org/10.1002/nme.4371
27. Pagani, S., Manzoni, A., Quarteroni, A.: Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method. Comput. Methods Appl. Mech. Eng. **340**, 530–558 (2018). https://doi.org/10.1016/j.cma.2018.06.003
28. Bonito, A., Cohen, A., DeVore, R., Guignard, D., Jantsch, P., Petrova, G.: Nonlinear methods for model reduction. ESAIM: Math. Model. Num. Anal. **55**(2), 507–531 (2021). https://doi.org/10.1051/m2an/2020057
29. Geelen, R., Willcox, K.: Localized non-intrusive reduced-order modelling in the operator inference framework. Phil. Trans. R. Soc. A **380**(2229), 20210206 (2022). https://doi.org/10.1098/rsta.2021.0206
30. Nouy, A., Pasco, A.: Dictionary-based model reduction for state estimation. Adv. Comput. Math. **50**(3), 1–31 (2024)
31. Amsallem, D., Farhat, C.: Interpolation method for adapting reduced-order models and application to aeroelasticity. AIAA J. **46**(7), 1803–1813 (2008). https://doi.org/10.2514/1.35374
32. Boncoraglio, G., Farhat, C., Bou-Mosleh, C.: Model reduction framework with a new take on active subspaces for optimization problems with linearized fluid-structure interaction constraints. Int. J. Numer. Meth. Eng. **122**(19), 5450–5481 (2021). https://doi.org/10.1002/nme.6376
33. Boncoraglio, G., Farhat, C.: Active manifold and model-order reduction to accelerate multidisciplinary analysis and optimization. AIAA J. **59**(11), 4739–4753 (2021). https://doi.org/10.2514/1.J060581
34. Peherstorfer, B.: Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. SIAM J. Sci. Comput. **42**(5), 2803–2836 (2020). https://doi.org/10.1137/19M1257275
35. Hesthaven, J.S., Pagliantini, C., Ripamonti, N.: Rank-adaptive structure-preserving model order reduction of hamiltonian systems. ESAIM: Math. Model. Num. Anal. **56**(2), 617–650 (2022). https://doi.org/10.1051/m2an/2022013
36. Hesthaven, J., Pagliantini, C., Ripamonti, N.: Adaptive symplectic model order reduction of parametric particle-based vlasov-poisson equation. Math. Comput. **93**(348), 1153–1202 (2024). https://doi.org/10.1090/mcom/3885
37. Peherstorfer, B., Willcox, K.: Online adaptive model reduction for nonlinear systems via low-rank updates. SIAM J. Sci. Comput. **37**(4), 2123–2150 (2015). https://doi.org/10.1137/140989169
38. Singh, R., Uy, W.I.T., Peherstorfer, B.: Lookahead data-gathering strategies for online adaptive model reduction of transport-dominated problems. Chaos: Interdiscipl. J. Nonlin. Sci. **33**(11), (2023). https://doi.org/10.1063/5.0169392
39. Berman, J., Peherstorfer, B.: Colora: continuous low-rank adaptation for reduced implicit neural modeling of parameterized partial differential equations. In: Proceedings of the 41st International Conference on Machine Learning. ICML'24. JMLR.org, online (2024). https://doi.org/10.5555/3692070.3692212
40. Barnett, J., Farhat, C., Maday, Y.: Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility. J. Comput. Phys. **492**, 112420 (2023). https://doi.org/10.1016/j.jcp.2023.112420

41. Tait, D.J.: Deep orthogonal decompositions for convective nowcasting (2020). arXiv preprint arXiv:2006.15628. https://doi.org/10.48550/arXiv.2006.15628
42. Aubin, J.-P., Frankowska, H.: Set-valued Analysis. Springer, Harrisonburg, Virginia (2009). https://doi.org/10.1007/978-0-8176-4848-0
43. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991). https://doi.org/10.1016/0893-6080(91)90009-T
44. Franco, N.R., Fresca, S., Manzoni, A., Zunino, P.: Approximation bounds for convolutional neural networks in operator learning. Neural Netw. **161**, 129–141 (2023). https://doi.org/10.1016/j.neunet.2023.01.029
45. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks (2018). arXiv preprint arXiv:1806.01261. https://doi.org/10.48550/arXiv.1806.01261
46. Franco, N.R., Manzoni, A., Zunino, P.: Mesh-informed neural networks for operator learning in finite element spaces. J. Sci. Comput. **97**(2), 35 (2023). https://doi.org/10.1007/s10915-023-02331-1
47. Fresca, S., Manzoni, A.: POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Comput. Methods Appl. Mech. Eng. **388**(114181), (2022). https://doi.org/10.1016/j.cma.2021.114181
48. Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., Karniadakis, G.E.: A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Comput. Methods Appl. Mech. Eng. **393**, 114778 (2022). https://doi.org/10.1016/j.cma.2022.114778
49. Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU press, Baltimore, Maryland (2013). https://doi.org/10.56021/9781421407944
50. Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics, vol. 37. Springer, Berlin Heidelberg (2006). https://doi.org/10.1007/b98885
51. Zimmermann, R., Peherstorfer, B., Willcox, K.: Geometric subspace updates with applications to online adaptive nonlinear model reduction. SIAM J. Matrix Anal. Appl. **39**(1), 234–261 (2018). https://doi.org/10.1137/17M1123286
52. Bendokat, T., Zimmermann, R., Absil, P.-A.: A grassmann manifold handbook: Basic geometry and computational aspects. Adv. Comput. Math. **50**(1), 1–51 (2024). https://doi.org/10.1007/s10444-024-10115-w
53. Wong, Y.-C.: Differential geometry of grassmann manifolds. Proc. Natl. Acad. Sci. **57**(3), 589–594 (1967). https://doi.org/10.1073/pnas.57.3.589
54. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. **20**(2), 303–353 (1998). https://doi.org/10.1137/S0895479895290954
55. Dubey, P., Müller, H.-G.: Functional models for time-varying random objects. J. R. Stat. Soc. Ser. B Stat Methodol. **82**(2), 275–327 (2020). https://doi.org/10.1111/rssb.12337
56. Daniel, T., Casenave, F., Akkari, N., Ryckelynck, D.: Model order reduction assisted by deep neural networks (rom-net). Adv. Model. Simul. Eng. Sci. **7**, 1–27 (2020). https://doi.org/10.1186/s40323-020-00153-6
57. Herkert, R., Buchfink, P., Haasdonk, B.: Dictionary-based online-adaptive structure-preserving model order reduction for parametric hamiltonian systems. Adv. Comput. Math. **50**(1), 12 (2024). https://doi.org/10.1007/s10444-023-10102-7
58. Brivio, S., Fresca, S., Franco, N.R., Manzoni, A.: Error estimates for pod-dl-roms: a deep learning framework for reduced order modeling of nonlinear parametrized pdes enhanced by proper orthogonal decomposition. Adv. Comput. Math. **50**(33), (2024). https://doi.org/10.1007/s10444-024-10110-1
59. Franco, N.R.: NicolaRFranco/dlroms: First release. Zenodo (2024). https://doi.org/10.5281/zenodo.13254758
60. Brivio, S., Franco, N.R.: Deep symmetric autoencoders from the eckart-young-schmidt perspective. arXiv:2506.11641 (2025)
61. Bourguet, R., Braza, M., Dervieux, A.: Reduced-order modeling of transonic flows around an airfoil submitted to small deformations. J. Comput. Phys. **230**(1), 159–184 (2011). https://doi.org/10.1016/j.jcp.2010.09.019
62. Liberge, E., Hamdouni, A.: Reduced order modelling method via proper orthogonal decomposition (pod) for flow around an oscillating cylinder. J. Fluids Struct. **26**(2), 292–311 (2010). https://doi.org/10.1016/j.jfluidstructs.2009.10.006
63. Antil, H., Heinkenschloss, M., Sorensen, D.C.: Application of the discrete empirical interpolation method to reduced order modeling of nonlinear and parametric systems. In: Quarteroni, A., Rozza, G.

(eds.) Reduced Order Methods for Modeling and Computational Reduction, pp. 101–136. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-02090-7_4
64. Yin, M., Charon, N., Brody, R., Lu, L., Trayanova, N., Maggioni, M.: Dimon: Learning solution operators of partial differential equations on a diffeomorphic family of domains (2024). arXiv preprint arXiv:2402.07250. https://doi.org/10.48550/arXiv.2402.07250
65. Taddei, T.: A registration method for model order reduction: data compression and geometry reduction. SIAM J. Sci. Comput. **42**(2), 997–1027 (2020). https://doi.org/10.1137/19M1271270
66. Barwey, S., Shankar, V., Viswanathan, V., Maulik, R.: Multiscale graph neural network autoencoders for interpretable scientific machine learning. J. Comput. Phys. **495**, 112537 (2023). https://doi.org/10.1016/j.jcp.2023.112537
67. Franco, N.R., Fresca, S., Tombari, F., Manzoni, A.: Deep learning-based surrogate models for parametrized pdes: Handling geometric variability through graph neural networks. Chaos: Interdiscipl. J. Nonlin. Sci. **33**(12), 1 (2023). https://doi.org/10.1063/5.0170101
68. Gladstone, R.J., Rahmani, H., Suryakumar, V., Meidani, H., D'Elia, M., Zareei, A.: Gnn-based physics solver for time-independent pdes (2023). arXiv preprint arXiv:2303.15681. https://doi.org/10.48550/arXiv.2303.15681
69. Lin, F.-C., Ritzwoller, M.H., Snieder, R.: Eikonal tomography: surface wave tomography by phase front tracking across a regional broad-band seismic array. Geophys. J. Int. **177**(3), 1091–1110 (2009). https://doi.org/10.1111/j.1365-246X.2009.04105.x
70. Ma, T., Zhang, Z.: Calculating ray paths for first-arrival travel times using a topography-dependent eikonal equation solver. Bull. Seismol. Soc. Am. **104**(3), 1501–1517 (2014). https://doi.org/10.1785/0120130172
71. Ghanem, R.G., Spanos, P.D.: Stochastic Finite Elements: a Spectral Approach. Springer, New York (2003). https://doi.org/10.107/978-1-4612-3094-6
72. Mokrỳ, P.: Iterative method for solving the eikonal equation. In: Optics and Measurement International Conference 2016, vol. 10151, pp. 263–268 (2016). https://doi.org/10.1117/12.2257326. SPIE
73. Hesthaven, J.S., Ubbiali, S.: Non-intrusive reduced order modeling of nonlinear problems using neural networks. J. Comput. Phys. **363**, 55–78 (2018). https://doi.org/10.1016/j.jcp.2018.02.037
74. Vitullo, P., Colombo, A., Franco, N.R., Manzoni, A., Zunino, P.: Nonlinear model order reduction for problems with microstructure using mesh informed neural networks. Finite Elem. Anal. Des. **229**, 104068 (2024). https://doi.org/10.1016/j.finel.2023.104068
75. Guo, M., Hesthaven, J.S.: Reduced order modeling for nonlinear structural analysis using gaussian process regression. Comput. Methods Appl. Mech. Eng. **341**, 807–826 (2018). https://doi.org/10.1016/j.cma.2018.07.017
76. Peherstorfer, B., Willcox, K., Gunzburger, M.: Optimal model management for multifidelity monte carlo estimation. SIAM J. Sci. Comput. **38**(5), 3163–3194 (2016)
77. Farcaş, I.-G., Peherstorfer, B., Neckel, T., Jenko, F., Bungartz, H.-J.: Context-aware learning of hierarchies of low-fidelity models for multi-fidelity uncertainty quantification. Comput. Methods Appl. Mech. Eng. **406**, 115908 (2023)
78. Gühring, I., Kutyniok, G., Petersen, P.: Error bounds for approximations with deep relu neural networks in w s, p norms. Anal. Appl. **18**(05), 803–859 (2020)
79. Adcock, B., Dexter, N.: The gap between theory and practice in function approximation with deep neural networks. SIAM J. Math. Data Sci. **3**(2), 624–655 (2021)
80. Pagliantini, C.: Dynamical reduced basis methods for hamiltonian systems. Numer. Math. **148**(2), 409–448 (2021)
81. Matray, V., Néron, D., Feyel, F., Amlani, F.: Geometry-agnostic model reduction with gnn-generated reduced pod bases and boosted pgd enrichment for (non) linear structural elastodynamics. Comput. Methods Appl. Mech. Eng. **448**, 118357 (2026)
82. Zieliński, P., Hesthaven, J.S.: Discovery of slow variables in a class of multiscale stochastic systems via neural networks. J. Nonlin. Sci. **32**(4), 51 (2022). https://doi.org/10.1007/s00332-022-09808-7
83. Amsallem, D., Farhat, C.: An online method for interpolating linear parametric reduced-order models. SIAM J. Sci. Comput. **33**(5), 2169–2198 (2011). https://doi.org/10.1137/100813051

Authors and Affiliations

Nicola Rares Franco[1] · Andrea Manzoni[1] · Paolo Zunino[1] · Jan S. Hesthaven[2]

✉ Nicola Rares Franco
nicolarares.franco@polimi.it

Andrea Manzoni
andrea1.manzoni@polimi.it

Paolo Zunino
paolo.zunino@polimi.it

Jan S. Hesthaven
jan.hesthaven@kit.edu

[1] MOX, Department of Mathematics, Politecnico di Milano, P.zza Leonardo da Vinci, 32, Milan 20133, Italy

[2] Karlsruhe Institute of Technology (KIT), Kaiserstrasse 12, Karlsruhe 76131, Germany