Demo *et al. Advanced Modeling and Simulation in Engineering Sciences(2023)10:12*
https://doi.org/10.1186/s40323-023-00249-9

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

A DeepONet multi-fidelity approach for residual learning in reduced order modeling

Nicola Demo[1], Marco Tezzele[1,2] and Gianluigi Rozza[1*]

*Correspondence: gianluigi.rozza@sissa.it
[1]Mathematics Area, mathLab, SISSA, Trieste, Italy
[2]Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX, USA

Abstract

In the present work, we introduce a novel approach to enhance the precision of reduced order models by exploiting a multi-fidelity perspective and DeepONets. Reduced models provide a real-time numerical approximation by simplifying the original model. The error introduced by the such operation is usually neglected and sacrificed in order to reach a fast computation. We propose to couple the model reduction to a machine learning residual learning, such that the above-mentioned error can be learned by a neural network and inferred for new predictions. We emphasize that the framework maximizes the exploitation of high-fidelity information, using it for building the reduced order model and for learning the residual. In this work, we explore the integration of proper orthogonal decomposition (POD), and gappy POD for sensors data, with the recent DeepONet architecture. Numerical investigations for a parametric benchmark function and a nonlinear parametric Navier-Stokes problem are presented.

Keywords: DeepONet, Multi-fidelity, POD, Gappy POD

Introduction

Multi-fidelity (MF) methods emerged as a solution to deal with complex models, which usually need a high computational budget to be solved [1]. Such a framework aims to exploit not only the so-called high-fidelity information, but also the response of low-fidelity models in order to increase the accuracy of the prediction. This feature plays a fundamental role, especially for outer loop applications such as uncertainty propagation and optimization, since it allows to achieve good accuracy without requiring evaluating the high-fidelity model (typically expensive) at every iteration. Thus, its employment is widespread for optimization purposes, and among all the contributions in literature, we highlight the successful application to naval engineering problems [2–4], to multiple fidelities modeling [5], and in the presence of uncertainty [6]. All these cases, as well as many others, build the correlation between the different fidelities by involving Gaussian process regression (GPR). Another approach with nonlinear autoregressive schemes is described in [7,8], whereas in [9] a possible extension for high-dimensional parameter spaces is investigated. Recently, an alternative to such a probabilistic framework is offered

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

by neural networks, where the mapping between the low-fidelity model and the high-fidelity one is learned by the network during the training procedure [10–12]. Among the different types of architecture, DeepONet [13,14] has been proposed to approximate operators and it has been successfully applied to MF problems in [15,16]. It has also been successfully used to create a fast PDE-constrained optimization method in [17]. Another type of architecture that has been successfully applied to multi-fidelity data is the Bayesian neural network [18], resulting in a framework robust to noisy measurements. We also highlight the employment of multi-fidelity techniques for uncertainty quantification. We cite [19,20] for a Bayesian framework capable to deal with model discrepancy using different fidelities, whereas we refer to [21] for an analysis of the trade-off between high- and low-fidelity data in a Monte Carlo estimation.

Reduced order modeling (ROM) [22–24] is a family of methods that aims at reducing the computational burden of evaluating complex models. Instead of combining data from heterogeneous models, ROM builds a simplified model, typically from some high-fidelity information. Also, in this case, the capabilities of ROM led to its diffusion in several industrial contexts [4,25,26], especially for optimization tasks [27–32] or inverse problems [33,34]. In the ROM community, proper orthogonal decomposition (POD) is one of the most employed methods to build the reduced model [35–39]. Given a limited set of high-fidelity data, POD is able to compute the reduced space of an arbitrary rank which optimally (in a least squares sense) represents the data. In the last years, its diffusion led to several variants including shifted POD [40,41], weighted POD [42,43], and gappy POD [44–47]. This latter exploit a compressive sensing approach [48–50], in order to use only a few information at certain locations of the domain (sensors) to compute the approximation. A generalization of gappy POD can be found in [51], where linear stochastic estimation allows the reconstruction of the linear map between the available data and the system state by an l_2 minimization. A novel approach where such a relation between sensor data and the reduced state is approximated in a nonlinear way employing neural networks can be found in [52].

In the present contribution, we explore the possibility of coupling these two methodologies, MF and ROM, to enhance the accuracy of the model. ROM indeed creates a simplified model from a few high-fidelity data. Such approximation can be considered the low-fidelity model, because of the projection error introduced by the ROM. In this context, MF could be adopted in order to find the correlation between the original model and the ROM one, resulting in a more precise prediction. We can therefore exploit twice the collected high-fidelity data: initially, it is used to build the reduced model, then again during the computation of the MF relation. From this point of view, the proposed improvement does not need any additional high-fidelity evaluations. Here we take into consideration the POD with interpolation or the gappy POD as low-fidelity modeling techniques and the DeepONet to learn the residual. POD with interpolation [53–56] is applied here for a completely data-driven approach, while gappy POD is used in order to make the pipeline applicable even for sensor data. The framework aims then to exploit the capability of POD models for linear prediction, adding the nonlinear term through the DeepONet, which can be viewed as a data-driven closure model. See [57] for another data-driven modelling approach to close ROMs, while for other recent works that propose nonlinear model order reduction, we cite [58–64].

The manuscript is organized as follows. In Sect. we present the end-to-end numerical pipeline, with a focus on POD, gappy POD, and DeepONets. We continue in Sect. by showing the numerical experiments, and finally we conclude with Sect. by summarizing the results and drawing some future extensions.

Methods

This section is devoted to present the numerical methods used within the proposed approximation scheme, together with the methods used for comparison. We describe their integration in order to provide a global overview, then we discuss in the following sections the algorithmic details.

Proper orthogonal decomposition (POD) is a widespread technique providing a linear model order reduction, particularly suited to deal with parametric problems [24,65,66]. Such a representation is computationally very cheap to acquire, however, it suffers from the linear limitations of POD that may decrease its accuracy, especially when dealing with nonlinear problems.

We are interested in efficiently computing a parametric field $\mathbf{u}(\mu)$ with $\mathbf{u} : P \to \mathcal{V}$, where P is the parametric space, $\mathcal{V}$ a generic norm equipped vector space with $\dim(\mathcal{V}) = n$. POD-based ROMs compute the approximation $\mathbf{u}_{\text{POD}}(\mu)$ such that:

$$\mathbf{u}_{\text{POD}}(\mu) \approx \mathbf{u}(\mu) = \mathbf{u}_{\text{POD}}(\mu) + \mathbf{r}(\mu), \tag{1}$$

where $\mathbf{r} : P \to \mathcal{V}$ is the projection error introduced by the model order reduction, which we assume here to be dependent on the parameter. In a classical POD framework, this residual $\mathbf{r}$ is usually neglected, due to its marginal contribution. In the present contribution, we aim instead to learn it by means of machine learning techniques, in order to improve the accuracy of the final prediction. Artificial neural networks (ANNs) can be used to model it, thanks to their general approximation capabilities, learning it by exploiting the snapshots already pre-computed to build the ROM. In particular, dealing with parametric problems, we exploit the DeepONet architecture to learn the residual. The light computational demand to infer the DeepONet enables a nonlinear but still real-time improvement of the POD model, at the cost of additional training during the offline phase.

The only input needed by the proposed methodology is the numerical solutions database $\{\mu_i, \mathbf{u}(\mu_i)\}_{i=1}^N$ computed by sampling the parameter space and exploiting any consolidated discretization method (e.g. finite element or finite volume method). These snapshots are combined in order to find the POD space, which can be used for intrusive or non-intrusive ROM. We explore in this contribution only the non-intrusive (data-driven) approach, while future works will study the application to POD-Galerkin contexts. We investigate two options for the non-intrusive ROM:

- POD with radial basis functions (POD-RBF) interpolation, which enables the prediction of new solutions (for new parameters) by means of the above-mentioned interpolation technique. In this case, the ROM takes as input the actual parameter providing as output the approximated solution.
- Gappy POD, which allows us to compute the approximated solution by providing only some sensor data thanks to a compressing strategy.

Once the ROM is built, we can exploit it to compute the low-fidelity representation of the original snapshots by passing the corresponding parameters (or sensor data). The high-

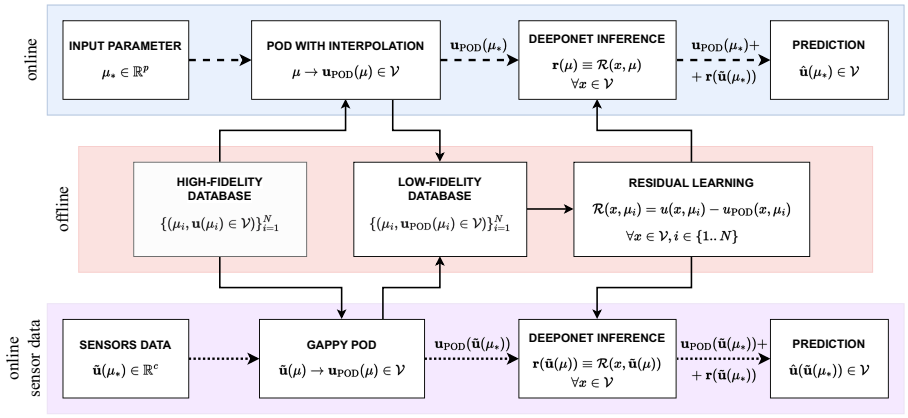

Fig. 1 Scheme for the multi-fidelity POD framework. The arrows indicate the relationship between the different methods. In the blue box the dashed arrows indicate the online phase when the input parameter is provided. In the purple box the dotted arrows indicate the information flow if only sensor data are provided. The central red frame emphasizes the computationally expensive offline phase

fidelity and low-fidelity databases are then used to learn the difference between them through the DeepONet network with the final aim of generalizing such residual even to unseen parameters and improving the final prediction. It is important to note that typically the space $\mathcal{V}$ is obtained by discretizing a generic $\mathbb{R}^d$ space. Depending on the complexity of the equation to solve and on the target accuracy, this kind of space can exhibit a high number of degrees of freedom.

Approximating the error over such a high-dimensional space with a neural network leads to two major issues: *i*) the number of the neurons in the last layer is equal to the number of degrees of freedom of the space $\mathcal{V}$, resulting in a model too large to treat; *ii*) the parameter-to-error relation becomes too complex to be efficiently learned. Thus we extract the spatial coordinates of the degrees of freedom of $\mathcal{V}$. Since we know the error (the difference between the original snapshots and the POD predictions) in any of these coordinates, we can arrange the data in the format $\{(x_i, \mu_j, \mathbf{r}(\mu_j)_i) \,|\, x_i \in \mathcal{V} \subset \mathbb{R}^d, \mu_j \in P, \mathbf{r}(\mu_j)_i \in \mathbb{R}\}$ where $i = 1, \ldots, n$ and $j = 1, \ldots, N$, to isolate the spatial and parametric dependency of the error. We can use such a dataset to learn the scalar error $r_{\text{net}} : \mathbb{R}^d \times P \to \mathbb{R}$ given the parametric and spatial coordinates. In this way, the network maintains a limited number of output dimensions, improving the identification of spatial recurrent patterns. The loss function which is minimized during the training procedure is then:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{r}_{\text{NN}}(\mu_i) + \mathbf{u}_{\text{POD}}(\mu_i) - \mathbf{u}(\mu_i)\|^2, \tag{2}$$

where $\mathbf{r}_{\text{NN}}(\mu)$ is not the high-dimensional output of a single network evaluation, but the array containing the result of the network inference for any spatial coordinates belonging to the discretized space such that $\mathbf{r}_{\text{NN}}(\mu) \equiv \left[r_{\text{net}}(x_1, \mu)\ r_{\text{net}}(x_2, \mu) \ldots r_{\text{net}}(x_n, \mu)\right]$, where $x_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$. In the case of gappy POD, it is important to note that the DeepONet takes as input the sensors data and not the actual parameters. We emphasize that the DeepONet training does not need any additional high-fidelity solutions besides those already collected for the POD space construction.

For the prediction of solutions for new parameters, the non-intrusive POD model and the DeepONet are finally queried, as sketched in Fig. 1. POD returns the low-fidelity

(linear) approximation by providing the test parameter or sensor data, while the neural network returns the nonlinear residual. In some sense, this pipeline aims to exploit the advantages of the consolidated POD model, but at the same time improves it by adding a nonlinear term. So it can be also seen as a closure model.

Proper orthogonal decomposition for low-fidelity modeling

POD is a consolidated technique widely used for model order reduction. In this section, we briefly introduce how to compute the POD modes, and we devote section to present the gappy POD variant in detail.

The method consists of the computation of the optimal reduced basis to represent the parametric solution manifold through a linear projection. Let $u_i \in \mathbb{R}^n$ be the discrete solution corresponding to the i-th parameter, and $U = [u_1, \dots, u_N] \in \mathbb{R}^{n \times N}$ be the snapshots matrix, whose columns are the solution vectors. We want to find a linear approximation such that:

$$u_i \approx \sum_{k=1}^{r} a_i^k \psi_k, \qquad \text{for } r \ll n, \text{ and for } i = 1, \dots, N, \tag{3}$$

where $\psi_i \in \mathbb{R}^n$ are the vectors comprising the reduced basis, the so-called *modes*, and $a_i := \left(a_i^1, a_i^2, \dots, a_i^r\right) \in \mathbb{R}^r$ are the coordinates of the corresponding solution at the reduced level, called modal coefficients or latent variables. These reduced variables are obtained by a projection of the solution snapshots onto the modes.

The POD modes can be obtained from the matrix U in different ways: by computing its singular value decomposition (SVD), or by decomposing its correlation matrix [67]. Moreover, all the modes have a corresponding singular value, which represents their energetic contribution. By arranging these modes in decreasing order (with respect to the singular values), we can express the original system with a hierarchical basis, from which we can discard the less meaningful modes. The energy criterion based on the singular values decay reads as

$$\frac{\sum_{j=1}^{r} \sigma_j}{\sum_{j=1}^{N} \sigma_j} > \epsilon, \tag{4}$$

where σ_j is the j-th singular value, and ϵ is a tolerance, usually set ≥ 0.99. In other words, by providing some samples of the solution manifold, POD is able to detect correlations between the data and reduce the dimensionality of these discrete solutions. This the approach becomes a fundamental tool for solving parametric partial differential equations (PDEs) in a many-query context, mainly due to the high-dimensional discrete spaces involved.

The POD space can be exploited in a Galerkin framework, by projecting the differential operators, or in a data-driven fashion by coupling it with an interpolation (or regression) technique. In this case, the database of reduced snapshots $\{\mu_i, a_i\}_{i=1}^N$ is used as input to build the mapping $\mathcal{I} : P \to \mathbb{R}^r$ such that $\mathcal{I}(\mu_i) = a_i$ for $i = 1, \dots, N$, which is used for interpolating the modal coefficients for any new parameter. Depending on the chosen regression technique, the equality could not hold in principle, and we have $\mathcal{I}(\mu_i) \approx a_i$. Finally, exploiting such a mapping, we have the possibility to query for the modal coefficients at any test parameter belonging to the space P and finally exploit the POD modes to map back the approximated solution in the original high-dimensional space.

Gappy POD for sensors data

The main assumption for using gappy POD is to have access to only some sensor data. These sensors are placed at specific locations, given by the projection matrix, or point measurement matrix, $C \in \mathbb{R}^{c\times n}$, with $c \ll n$, which contains 1 at measurements location and 0 elsewhere. Using the canonical basis vectors of $\mathbb{R}^n$ it takes the following form

$$C = \begin{bmatrix} e_{\gamma_1} & e_{\gamma_2} & \cdots & e_{\gamma_c} \end{bmatrix}^T, \tag{5}$$

for some indices $\gamma_i \in [1, \ldots, n]$, with $i \in [1, \ldots, c]$. The measurements $\tilde{u}_* \in \mathbb{R}^c$ of a generic full state vector $u_* \in \mathbb{R}^n$ are thus given by

$$\tilde{u}_* = Cu_*. \tag{6}$$

If we now consider a parametric framework we can collect the parameter–solution snapshot pairs $\{\mu_i, u_i\}_{i=1}^N$, where $\mu_i \in P \subset \mathbb{R}^p$, and $u_i \in \mathbb{R}^n$ is the corresponding full state. We arrange the snapshots by column in U as

$$U = \begin{bmatrix} | & | & | & | \\ u_1 & u_2 & \ldots & u_N \\ | & | & | & | \end{bmatrix}. \tag{7}$$

We take the r-rank SVD of the snapshots matrix U and compute the POD modes Ψ_r, so we can project the full states to their low-rank representation $a \in \mathbb{R}^{r\times N}$:

$$U = \Psi \Sigma V^T \approx \Psi_r \Sigma_r V_r^T, \qquad U \approx \Psi_r a. \tag{8}$$

In the classical POD setting, where we deal with the full snapshots, we would just use the modal coefficients matrix a to describe the solution manifold. For the gappy POD, instead, we have to consider the point measurement matrix. So, putting all together we have

$$\tilde{U} \approx (C\Psi_r)a, \tag{9}$$

where $\tilde{U}$ is the matrix containing the sensors measurements $\{\tilde{u}_i\}_{i=1}^N$ arranged by columns, as done for the snapshots matrix. For a generic snapshot $\tilde{u}_i$ we have:

$$\tilde{u}_i \approx C \sum_{k=1}^{r} a_i^k \psi_k, \tag{10}$$

where ψ_k are the columns of Ψ_r, and a_i^k are the modal coefficients, that is the i-th column of a. A possible solution to find the modal coefficients is to minimize the residual in a least-squares sense using the L^2 norm over the sensors locations which means considering the following quantity [68]

$$\int_{\mathrm{supp}[\tilde{u}_i]} \left(\tilde{u}_i - \sum_{k=1}^{r} \tilde{a}_i^k \psi_k \right)^2. \tag{11}$$

There are many ways in the literature to select the locations of the sensors: optimal sensor locations that improve the condition number of $C\Psi$ [45,69], which are robust to sensor noise, the sample maximal variance positions [70], or using information contained in secant vectors between data points [71], for example. In this work, we are going to use the sparse sensor placement optimization for reconstruction described in details in [69] and implemented in PySensors [72]. The main idea is to find C that minimizes the reconstruction error using the modes Ψ_r as in the following

$$C^* = \underset{C}{\mathrm{argmin}} \, \| U - \Psi_r (C\Psi_r)^\dagger \tilde{U} \|_2^2, \tag{12}$$

where the symbol † stands for the Moore-Penrose pseudoinverse.

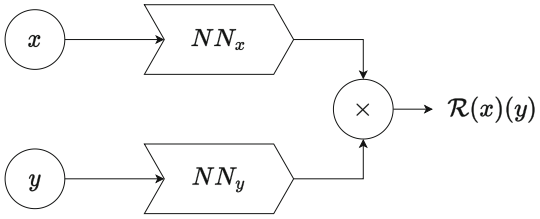

Fig. 2 The DeepONet scheme

DeepONet for residual learning

DeepONet [13] is a neural network architecture able to learn nonlinear operators. Referring to the original work for all the details, we emphasize its architecture composed by two separate networks whose final outputs are multiplied to obtain the final DeepONet outcome. The two networks, called *trunk* and *branch*, can be any available architecture —e.g. convolutional network, graph network—. In this work we consider feedforward networks (FFNs). The networks are trained simultaneously during the learning loop: the input is indeed divided into two independent components, $x \in \mathbb{R}^{N_x}$ and $y \in \mathbb{R}^{N_y}$, which feed the two networks NN_x and NN_y, respectively. The outputs $NN_x(x), NN_y(y) \in \mathbb{R}^{N_p}$ are finally multiplied to approximate the operator $\mathcal{G}$:

$$\mathcal{G}(x)(y) \approx \sum_{i=1}^{p} [NN_x(x)]_i [NN_y(y)]_i. \tag{13}$$

We underline that the choice of the two networks must satisfy the dimensional constraint: they have to produce outputs with the same number of components such that it is possible to compute their inner product. The scheme in Fig. 2 graphically summarizes the structure of the DeepONet.

In this work we adopt it to approximate the residual $\mathcal{R}(x)(\mu) = u(x, \mu) - u_{\text{POD}}(x, \mu)$ in a multi-fidelity approach. We can think at the mapping between the low-fidelity model (the POD/gappy POD) and the high-fidelity model as a parametric operator $\mathcal{R}(x)(\mu)$. This operator is numerically approximated by means of the DeepONet, using as dataset the low- and high-fidelity databases already computed. This architecture has demonstrated a great capability in fighting overfitting issues [13], allowing to generalize the residual even with a limited set of information

Numerical results

In this section we present the numerical results obtained by applying the proposed numerical framework to a simple algebraic problem and to a Navier–Stokes problem in a 2D domain. We are going to compare the proposed method with the POD model, the gappy POD model, and with the pure deep learning approach by using DeepONet, aiming for a fair comparison with two state-of-the-art techniques for (linear and nonlinear) data-driven modeling. All the computations are performed using PyTorch [73] for the artificial neural networks, EZyRB [74] for the POD with interpolation and gappy POD calculations. To solve the Navier-Stokes equations with the finite element method we use FEniCS [75].

Algebraic parametric problem

The first test case is a simple benchmark problem inspired by [76]. The high-fidelity parametric function $f^H : \Omega \times P \to \mathbb{R}$ is defined as

$$f^H(x;\mu) := \frac{1}{2}(\mu_1 x - 2)^2 \sin(12x - 4) + \sin(\mu_2 \cos(5x)), \tag{14}$$

where $x \in \Omega = [0, 1] \subset \mathbb{R}$, and $\mu = (\mu_1, \mu_2) \in P = [2, 15] \times [3, 20] \subset \mathbb{R}^2$. The first step is to compute the function value in some points in order to build the high-fidelity database. We use different sampling strategies for the spatial and parametric domain:

- we collect $n = 500$ equispaced samples $\{x_i^s\}_{i=1}^n$ in Ω;
- we collect 36 samples using the latin hypercube sampling, plus 4 additional samples at the corners of the domain, for a total of $N = 40$ points $\{\mu_i^s\}_{i=1}^N$ in P.

We thus compose the snapshots matrix, varying the parametric coordinates along the columns as follows:

$$\begin{bmatrix} f(x_1^s, \mu_1^s) & \dots & f(x_1^s, \mu_N^s) \\ \vdots & \ddots & \vdots \\ f(x_n^s, \mu_1^s) & \dots & f(x_n^s, \mu_N^s) \end{bmatrix} \in \mathbb{R}^{n \times N}. \tag{15}$$

Regarding the residual learning, we use the DeepONet model structured as follows: the *spatial* network (branch) is composed by 2 inner layers of 30 neurons each, with the softplus activation function, which is the smooth version of the Rectifier Linear Unit (ReLU) [77]; the *parametric* network (trunk) counts 2 inner layers with 30 neurons and the softplus function. The output layer has 30 neurons for both networks, without applying any additional function at this layer. The learning rate is equal to 0.005, the L^2-regularization factor is 0.0001.

We propose a comparison between the MF approach, POD, and DeepONet in terms of accuracy on test parameters with a fixed input database of solutions. We use different POD spaces in the comparison by selecting an increasing energetic threshold for the modes selection, aiming to analyze the difference in the error by varying the accuracy of the original POD model before getting improved by MFDeepONet[1]. We emphasize that no preprocessing or data centering is performed on the snapshots matrix, resulting in the first mode representing a large amount of energy. This corresponds to the minimal tolerance (0.99) in the experiments below. Regarding the DeepONet architecture, we employ the one described above also to learn the target function without the MF setting, such that the network learns the actual unknown field instead of the residual. In this way, we want to investigate the benefit of using the two methodologies (POD and DeepONet) in a multi-fidelity fashion instead of only separately. We measure the relative error on an equispaced grid of 20×20 parametric points.

POD with energy threshold 0.99 For the POD model, we select an energy threshold $\epsilon = 0.99$ corresponding to $N = 1$ mode and radial basis function (RBF) interpolation to approximate the map between the parameters and the latent variables. The training for DeepONet and MFDeepONet lasts 10,000 epochs. Figure 3 shows a quantitative comparison of the three investigated techniques, presenting the relative error in the whole parametric domain, the high-fidelity samples, and the error distribution. The last plot (bottom right corner) graphically shows the technique which best performs in all the tested parameters.

[1] For the remaining of this work, with *MFDeepONet* we are going to refer to the proposed technique.

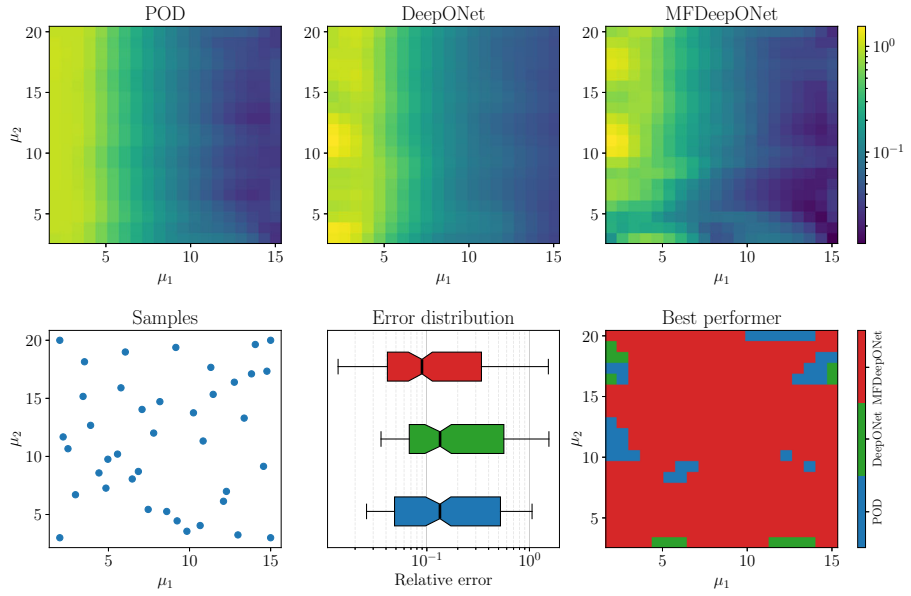

Fig. 3 Comparison between POD (0.99 energy threshold), DeepONet, and multi-fidelity DeepONet. From top to bottom, we have the relative error in the parametric domain, the location of the high-fidelity samples in the parametric domain, the relative error distribution, and the best performers

In this experiment, the proposed methodology outperforms both POD and DeepONet. The relative error distribution suggests that mixing the techniques helps in terms of accuracy. Indeed, even if the error shows a greater variance, the MFDeepONet is able on average to achieve the best precision among the tested methods, resulting the better approach in almost all the parametric domain. We can also note that a direct correlation between the samples location and the error distribution is not visible, confirming the DeepONet capabilities in terms of generalization and making the proposed framework effective also during the testing phase.

POD with energy threshold 0.999 In this experiment, we replicate the previous settings with the exception of the new energy threshold for POD modes and a higher number of epochs for the machine learning models (DeepONet and MFDeepONet). Here we increase it to $\epsilon = 0.999$ ($N = 6$ modes), addressing a more accurate original model, and balancing it with longer training.

Figure 4 illustrates the error obtained after a 20,000 epochs training. The results of the previous experiments are confirmed, even if with a lower overall benefit. The error distribution in the parametric space illustrates again how the MF enhancement combines the original methods: the regions of the parametric space where the methods work better are merged using MFDeepONet, resulting in a globally more accurate model. However, using a more precise POD model (as low-fidelity) reduces the benefits of the MF approach, even with the higher number of epochs.

Gappy POD Here we propose the same experiments as before, this time in a sensor data scenario. Here we use 5 sensor locations and a rank truncation equal to 10. The involved neural networks are trained in this case for 50,000 epochs.

Figure 5 summarizes the accuracy of the three tested methods, which are gappy POD, DeepONet, and MFDeepONet. The error distribution demonstrates that the multi-fidelity approach performs statistically better than the other methods. Looking at the competition

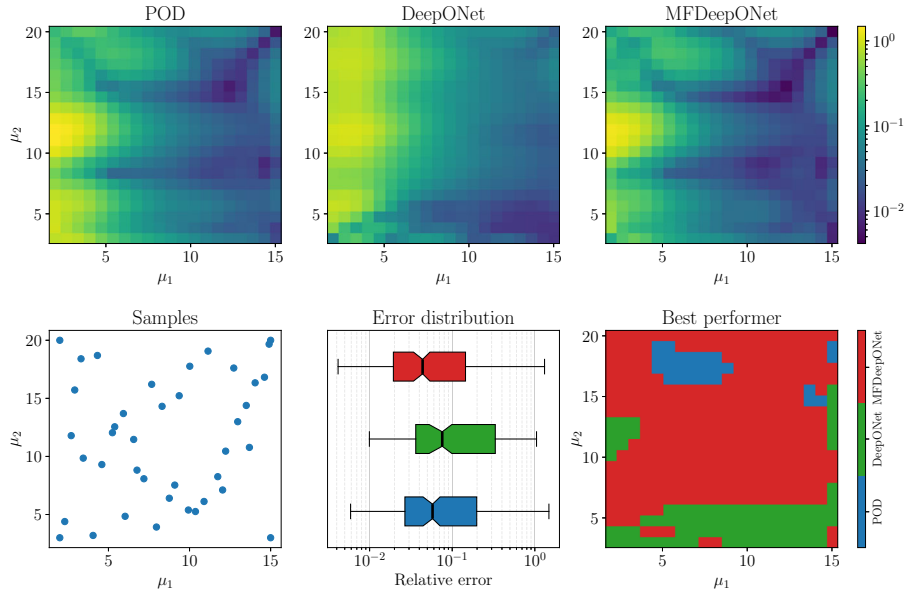

Fig. 4 Comparison between POD (0.999 energy threshold), DeepONet, and multi-fidelity DeepONet. From top to bottom, we have the relative error in the parametric domain, the location of the high-fidelity samples in the parametric domain, the relative error distribution, and the best performers

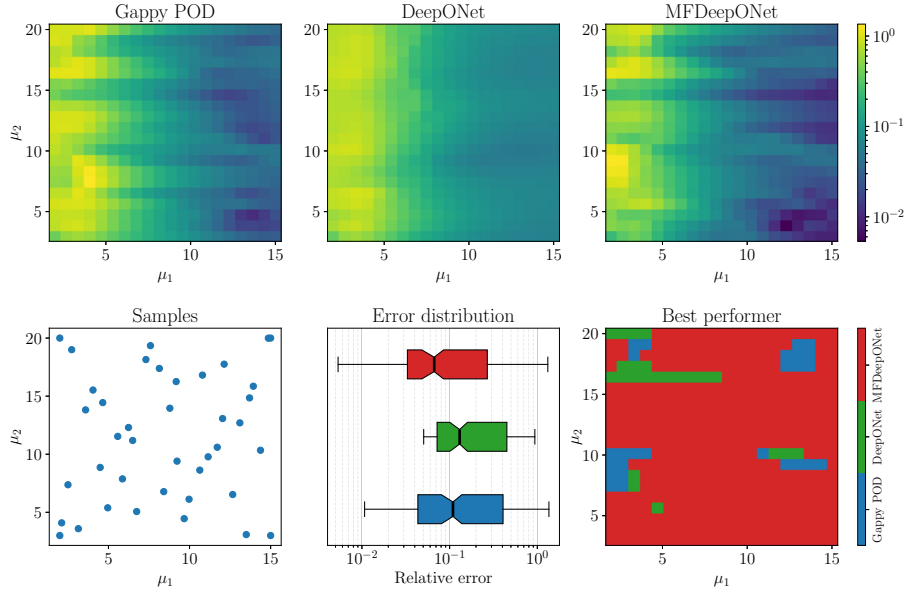

Fig. 5 Comparison between gappy POD, DeepONet, and multi-fidelity DeepONet. From top to bottom, we have the relative error in the parametric domain, the location of the high-fidelity samples in the parametric domain, the relative error distribution, and the best performers

between the techniques, we can also note that the multi-fidelity approach reaches the best accuracy in almost the whole parametric domain, even if at the boundaries there is a precision decrease. Such an issue could be mitigated by exploiting a better sampling strategy for the high-fidelity data.

The plots in Fig. 6 provide the comparison in the spatial domain at four test parameters. The statistical results are confirmed in these examples, with the multi-fidelity approach

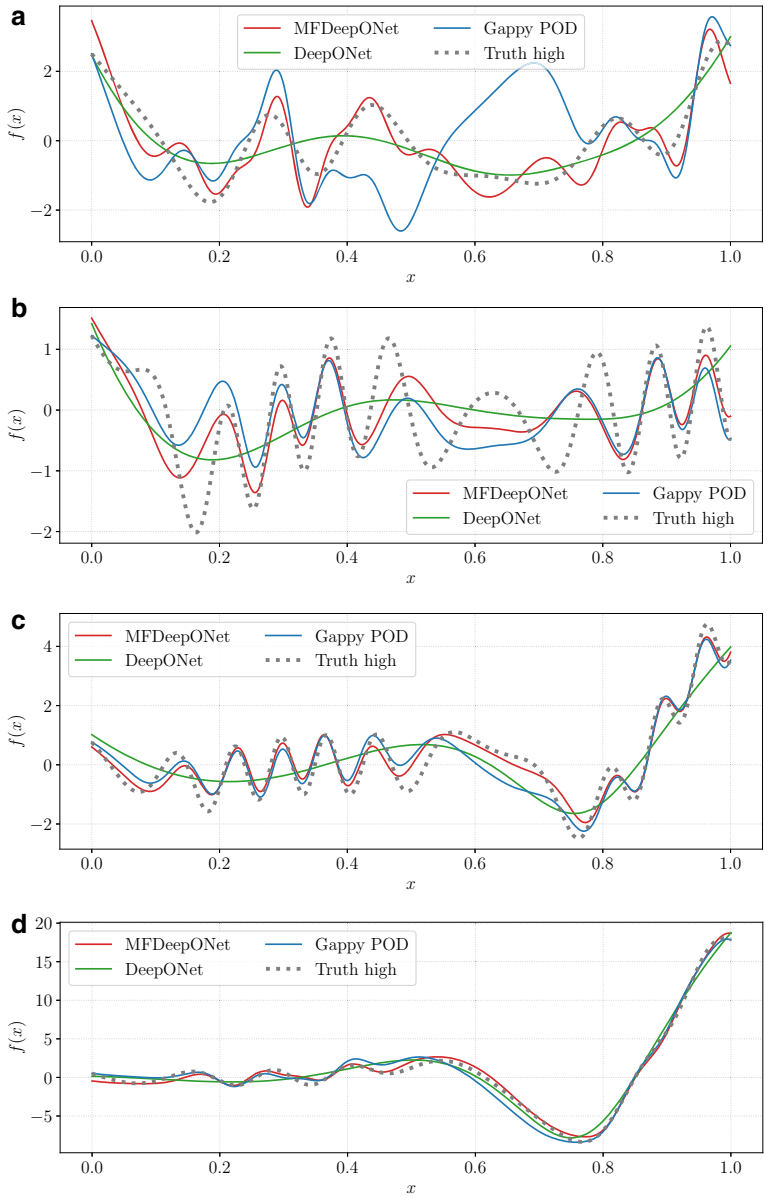

Fig. 6 Examples of prediction using gappy POD, DeepONet, and multi-fidelity DeepONet at different test parameters. From top to bottom we have: $\mu = [4, 8]$, $\mu = [3, 16]$, $\mu = [5, 18]$, and $\mu = [8, 11]$

that is able to predict most of the oscillations that the target function exhibits, contrarily to the single-fidelity approaches.

Navier Stokes problem

In the second numerical experiment, we test the accuracy of the proposed method for solving a parametric nonlinear PDE: the incompressible Navier–Stokes equation on a 2D domain. The numerical setting is inspired by [78].

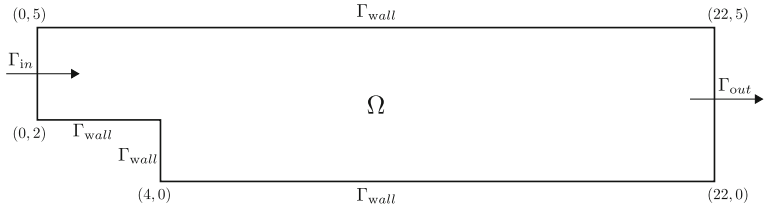

Fig. 7 Domain description

We define the parametric vector field $u : \Omega \times P \to \mathbb{R}^2$ and the parametric scalar field $p : \Omega \times P \to \mathbb{R}$ such that:

$$\begin{cases} \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0 & \text{in } \Omega, \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega, \\ \mathbf{u} = \mu \left\{ \frac{1}{2.25}(x_1 - 2)(5 - x_1), 0 \right\} & \text{on } \Gamma_{\text{in}}, \\ \mathbf{u} = 0 & \text{on } \Gamma_{\text{wall}}, \\ \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} = 0 & \text{on } \Gamma_{\text{out}}, \end{cases} \tag{16}$$

where $x = (x_0, x_1) \in \Omega \subset \mathbb{R}^2$ and $\mu \in P = [1, 80]$. The L-shape spatial domain Ω, together with the boundaries, is sketched in Fig. 7. For this test case, the parametric solution is computed numerically by means of finite element discretization. The spatial domain has been tessellated into 1639 non-overlapping elements, and for stability we apply the Taylor-Hood $P2-P1$ scheme. The high-fidelity dataset is composed of 20 equispaced parametric samples in P, arranged in the snapshots matrix $U \in \mathbb{R}^{n \times N}$ with $N = 20$ and $n = 1639$.

The DeepONet structure for this problem is the following:

- the spatial network (branch) is composed by 3 hidden layers of 50 neurons each;
- the parameter network (trunk) is composed by 3 hidden layers of 20 neurons each.

Also in this case, the last layer of the networks has the same number of neurons, 20. The activation function used in all the hidden layers is the Parametric ReLU (PReLU) [79], with the learning rate equal to 0.003 and the L^2-regularization factor equal to 0.001. The learning phase lasted 2.5×10^4 epochs. The accuracy of the MF approach is compared to the gappy POD and to the standard DeepONet, with the same architecture (single-fidelity). The relative error is evaluated over 500 testing points, randomly sampled in the parametric space.

POD with energy threshold 0.99 As before, we start with a relatively poor POD model, using $N = 1$ mode selected by the energetic criterion. RBF is employed also here to approximate the solution manifold at the reduced level. The number of epochs is fixed at 10,000 for the deep learning training.

Figure 8 shows the plot of the mean relative error over the spatial domain for all the test parameters, reporting also the location of the samples in the parameter space. As for the previous experiment, the proposed technique is able to keep a higher precision in the entire domain, without showing a visible correlation between the location of the high-fidelity data and the error trend, demonstrating its robustness in terms of possible overfitting. Employing the DeepONet architecture to learn the residual (between the POD and high-fidelity models) rather than the target function results in a more efficient learning procedure, capable to ourperform the single-fidelity approaches in the entire parametric space here considered.

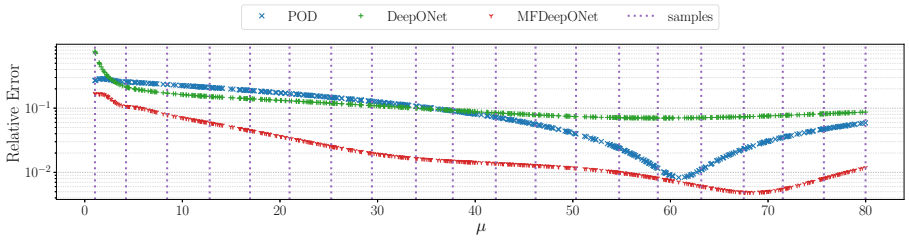

Fig. 8 Comparison between POD (0.99 energy threshold), DeepONet, and multi-fidelity DeepONet in terms of relative error in the parametric domain. The vertical dotted lines indicate the location of the high-fidelity samples

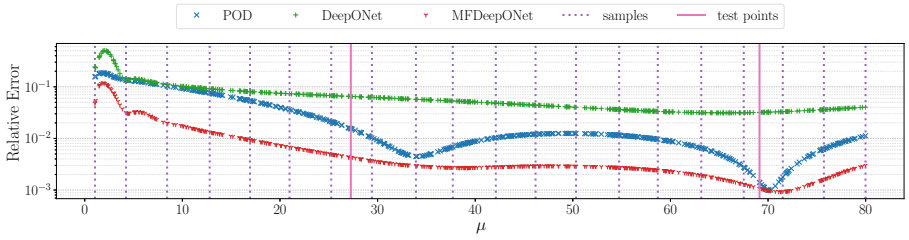

Fig. 9 Comparison between POD (0.999 energy threshold), DeepONet, and multi-fidelity DeepONet in terms of relative error in the parametric domain. The vertical dotted lines indicate the location of the high-fidelity samples. The 2 dashed vertical lines indicate the test parameters represented in Figs. 10 and 11

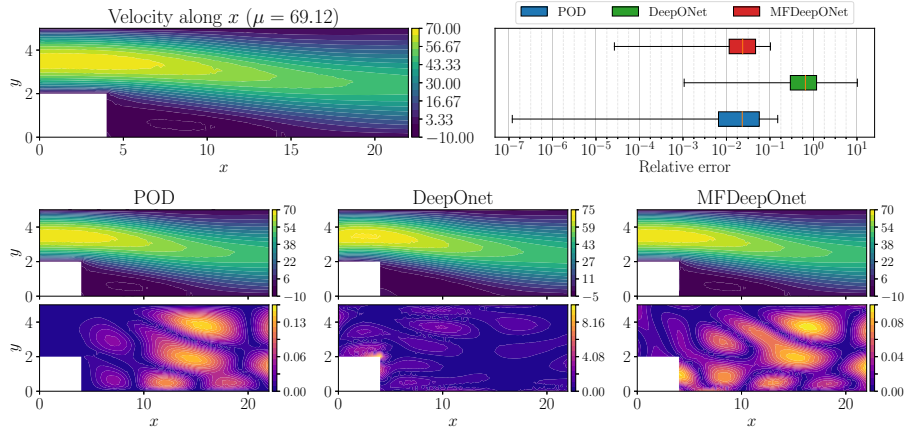

Fig. 10 Representation over the spatial domain of the velocity (along x) in the Navier–Stokes testcase for $\mu = 69.12$. The approximation computed by POD (0.999 energy threshold), DeepONet, and multi-fidelity DeepONet is shown at the bottom together with the relative error. The distribution of the error is summarized in the box plot

POD with energy threshold 0.999 As for the previous test case, we repeat the same experiment with a more accurate POD model. Here we use $N = 3$, raising the training time to 20,000 epochs.

The trend showed in the previous investigations is confirmed, as depicted in Fig. 9. The MFDeepONet method is able to produce a more accurate prediction in all the testing points, with no visible correlation with the training data. For a fair comparison, we also investigated the predicted field in the only point of the parametric domain where the MFDeepONet shows a slightly higher error with respect to the POD model (whereas the standard DeepONet performs poorly there).

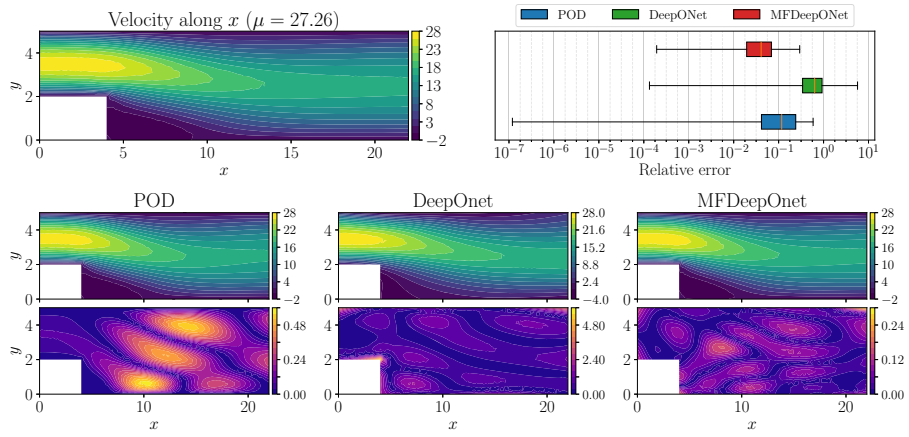

Fig. 11 Representation over the spatial domain of the velocity (along x) in the Navier–Stokes testcase for $\mu = 39.95$. The approximation computed by POD (0.999 energy threshold), DeepONet, and multi-fidelity DeepONet is shown at the bottom together with the relative error. The distribution of the error is summarized in the box plot

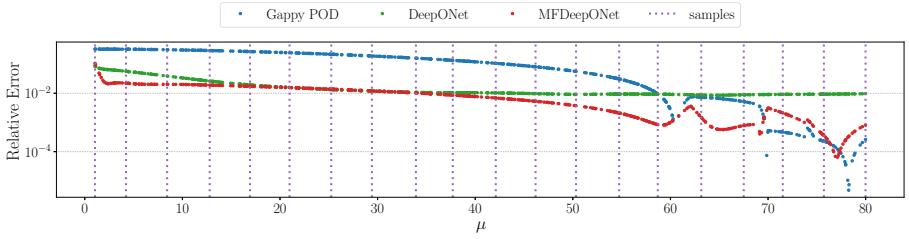

Fig. 12 Comparison between gappy POD, DeepONet, and multi-fidelity DeepONet in terms of relative error in the parametric domain. The vertical dotted lines indicate the location of the high-fidelity samples

Figure 10 shows the x-component of the velocity field for the parameter $\mu = 69.12$ obtained by the three methods, with a statistical summary of the relative error. The MF approach shows here a smaller spatial variance, even if on average performs equally to the POD model. Looking instead at a different parametric coordinate (Fig. 11), the benefits of the proposed approach become clear. The considerations regarding the variance of the error are still valid, but the solution for $\mu = 39.95$ shows a remarkable improvement in the accuracy over the testing points.

Gappy POD The last numerical experiment focuses on the Navier–Stokes model, for which sensor data are used by the gappy POD for the low-fidelity approximation. Here we use 7 sensor locations and a rank truncation equal to 8. We trained the DeepONet and the MFDeepONet for 50,000 epochs.

Figure 12 reports the relative test error measured in all the test points. In this case, the standard DeepONet, is able to outperform the POD model in a large region of the parametric domain, with a relative error that remains close to 0.01. Gappy POD is able to reach the best precision in a few test points, but also here the MF approach is the best compromise in terms of global accuracy, even if it is actually less precise than the POD model for high parameter values ($\mu > 70$).

Table 1 The mean relative error computed in all the experiments

		POD	**DeepONet**			**MFDeepONet**		
			10k	**20k**	**50k**	**10k**	**20k**	**50k**
Testcase #1	POD rank = 0.99	0.324	0.270	0.265	**0.217**	0.293	0.247	0.308
	POD rank = 0.999	0.203	0.270	0.265	0.217	0.196	0.193	**0.127**
	POD rank = 0.9999	0.098	0.270	0.265	0.217	**0.093**	0.104	0.178
Testcase #2	POD rank = 0.99	0.105	0.116	0.068	0.072	0.030	**0.022**	0.030
	POD rank = 0.999	0.033	0.116	0.068	0.072	0.025	0.023	**0.009**
	POD rank = 0.9999	0.011	0.116	0.068	0.072	0.011	**0.009**	**0.009**
	Gappy POD	**DeepONet**			**MFDeepONet**			
		10k	**20k**	**50k**	**10k**	**20k**	**50k**	
Testcase #1	0.260	0.419	0.380	0.278	0.218	0.217	**0.197**	
Testcase #2	0.135	0.035	0.033	0.017	0.029	0.010	**0.009**	

In bold the best results for each row

Summary discussion

This section is devoted to a summary discussion of the results obtained in the numerical investigations. For a fair comparison, we computed the mean relative test error[2] for each method, reporting the accuracy for different neural networks training times. In addition to the previous tests, we show in Table 1 the results obtained by employing a POD space whose modes are selected with an energetic threshold of $\epsilon = 0.9999$. The error charts for the missing cases, as well as some graphical representations of the parametric solutions, are reported in Appendix 4. The latter experiment aims to analyze the final accuracy when the low-fidelity POD is even more precise: the Mf approach is able to reach the best mean relative error, but its effectiveness is marginal, confirming the trend already defined in the previous tests. The combination of the POD model and DeepONet in the cascade fashion is able to reach the best accuracy in almost all the cases, but its improvement becomes marginal when the POD has good accuracy. Learning the residual however does not seem to affect the final outcome in a pejorative way, provided that the DeepONet is trained for a proper number of epochs. This is for sure a critical issue inherited by deep learning in general: we can indeed see that a longer training step does not always ensure better accuracy, producing instead over-fitting. On the practical side, the optimal settings of the network—e.g. training epochs, number of layers, type of activation function—need to be calibrated with a trial and error procedure or using more sophisticated approaches such as grid search. This calibration is out of the scope of this investigation where we want to formalize the novel framework, but surely sensitivity analysis regarding the hyper-parameters will be explored in future works. The generalization of the DeepONet, assisted also by the L_2-regularization imposed during the optimization, is able to improve accuracy over the entire parametric space, without showing a visible correlation between the location of the high-fidelity snapshots and the relative error spatial distribution.

To conclude, we highlight that the numerical experiments demonstrate a great improvement when the original POD model lacks accuracy, resulting in a great tool to treat problems where POD is not able to capture all the fluid characteristics, due to the complexity of the mathematical model or to the limited number of high-fidelity snapshots.

[2] We recall the test error is computed over a 20×20 regular grid for the algebraic problem, and at 460 random sample for the Navier–Stokes problem.

Conclusions and future perspectives

In this work, we introduced a novel approach to enhance POD-based reduced order models thanks to a residual learning procedure by DeepONet. It operates by building from a limited set of data an initial low-fidelity approximation exploiting established reduced order modeling techniques. Then it learns the difference between this low-fidelity representation and the original model through the artificial neural networks, that will be inferred to predict the solution at unseen parameters. We emphasize that such an enhancement neither needs any additional evaluation of the original model nor the knowledge of the high-fidelity model, resulting in a generic data-driven improvement at a fixed computational budget. This framework has demonstrated its effectiveness in two different test-cases: a univariate parametric function and a Navier–Stokes problem on a 2-dimensional domain, showing a higher precision in both experiments with respect to the use of single-fidelities. We highlight that in these experiments the number of considered POD modes is voluntarily kept small, simulating a POD model with poor accuracy.

The present work illustrates the pipeline for POD and gappy POD for the construction of the low-fidelity model and the DeepONet architecture for residual learning. Due to its modularity, the framework is general, admitting in principle to replace the low-fidelity models with different ones. Possible future extensions should investigate adaptive samplings and sensor placement exploiting the proposed numerical framework.

Appendix

This section presents additional plots for the POD energy threshold equal to 0.9999 case, for the algebraic function (Fig. 13) and for the parametric Navier-Stokes problem (Figs. 14 and 15).

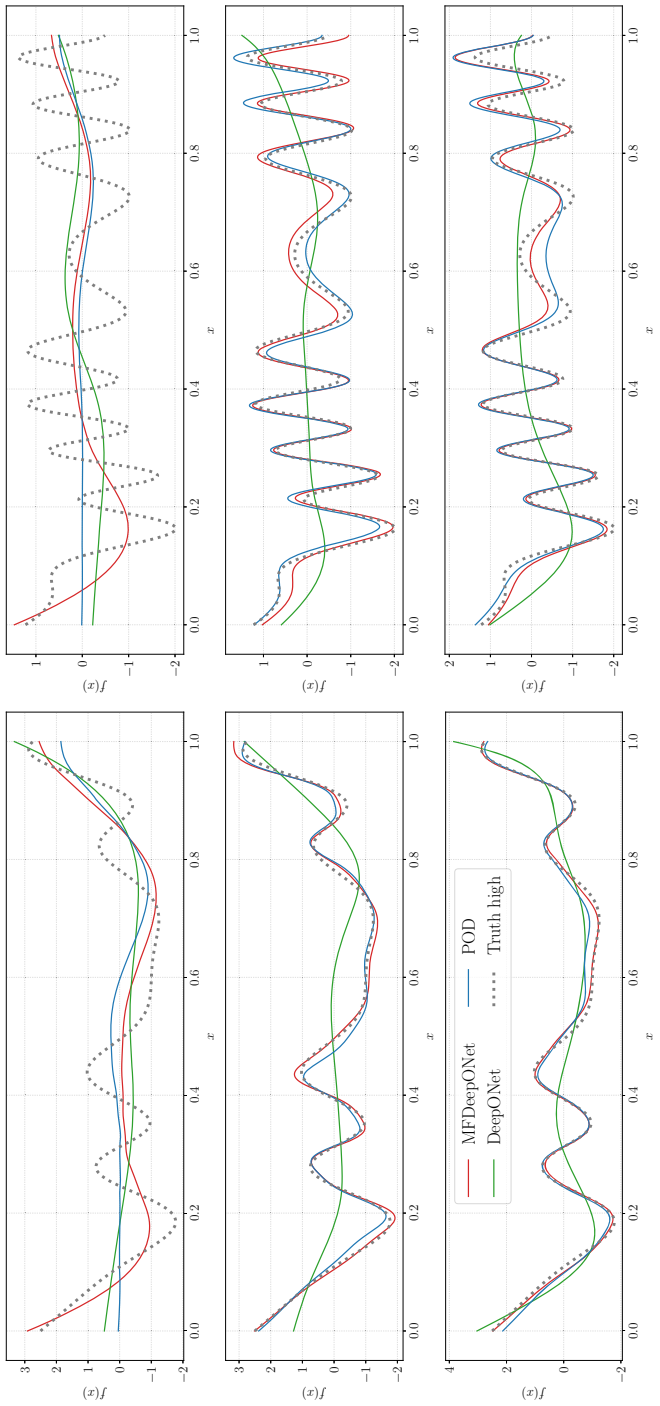

Fig. 13 Predictions at two different test parameters using gappy POD, DeepONet, and multi-fidelity DeepONet with different configurations, varying the number of epochs and the POD energy threshold. The left column shows the results for $\mu = [4, 8]$, while in the right one we have $\mu = [3, 16]$. *Top row*: POD energy threshold equal to 0.99, 10,000 epochs. *Center row*: POD energy threshold equal to 0.999, 50,000 epochs. *Bottom row*: POD energy threshold equal to 0.9999, 50,000 epochs

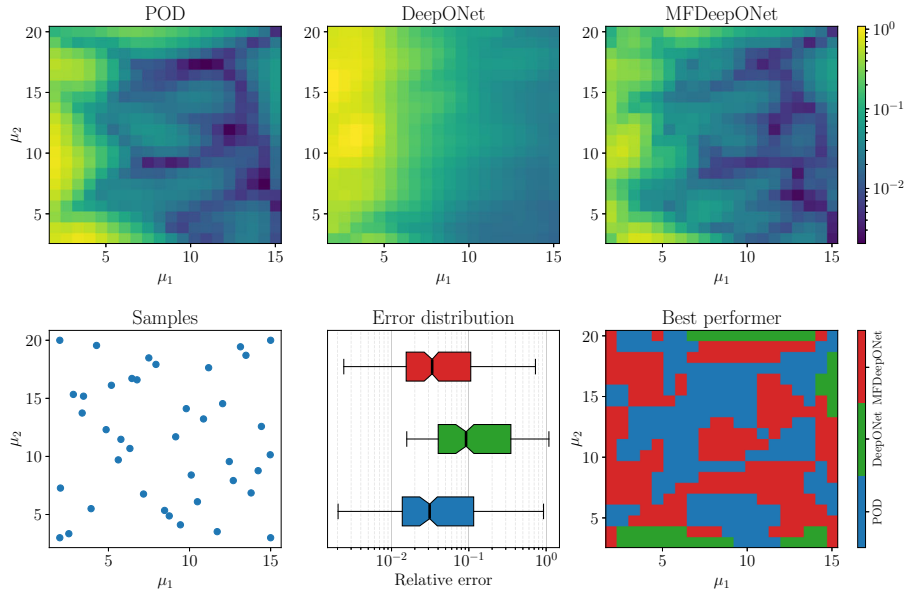

Fig. 14 Comparison between POD (0.9999 energy threshold), DeepONet, and multi-fidelity DeepONet. From top to bottom we have the relative error in the parametric domain, the location of the high-fidelity samples in the parametric domain, the relative error distribution, and the best performers

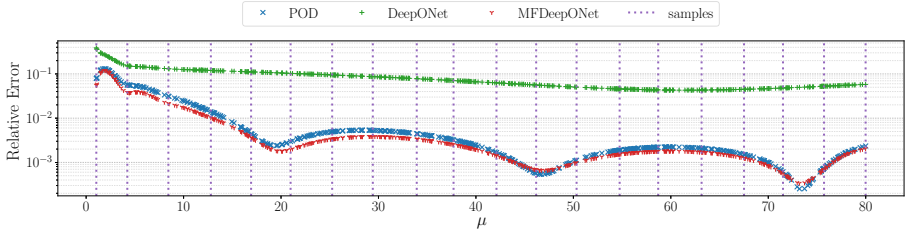

Fig. 15 Comparison between POD (0.9999 energy threshold), DeepONet, and multi-fidelity DeepONet in terms of relative error in the parametric domain. The vertical dotted lines indicate the location of the high-fidelity samples

Acknowledgements
Not applicable.

Author contributions
ND: Conceptualization, methodology, software, Writing—original draft, visualization. MT: Conceptualization, methodology, software, writing—original draft. GR: Writing—Review and editing, supervision, funding acquisition.

Funding
This work was partially supported by an industrial Ph.D. Grant sponsored by Fincantieri S.p.A. (IRONTH Project), by the MIT-FVG project "Multi-disciplinary Ship Design by Reduced Order Models and Machine Learning", and partially funded by European Union Funding for Research and Innovation—Horizon 2020 Program—in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 "Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics" P.I. Professor Gianluigi Rozza.

Availability of data and materials
All the computations were done using open source Python packages cited in the manuscript.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 10 April 2023 Accepted: 7 July 2023
Published online: 26 July 2023

References

1. Peherstorfer B, Willcox KE, Gunzburger M. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Rev. 2018;60(3):550–91. https://doi.org/10.1137/16M1082469.
2. Bonfiglio L, Perdikaris P, Vernengo G, de Medeiros JS, Karniadakis G. Improving SWATH seakeeping performance using Multi-Fidelity Gaussian Process and Bayesian Optimization. J Ship Res. 2018;62(4):223–40. https://doi.org/10.5957/JOSR.11170069.
3. Bonfiglio L, Perdikaris P, Brizzolara S, Karniadakis G. Multi-fidelity optimization of super-cavitating hydrofoils. Comput Methods Appl Mech Eng. 2018;332:63–85. https://doi.org/10.1016/j.cma.2017.12.009.
4. Tezzele M, Fabris L, Sidari M, Sicchiero M, Rozza G. A multi-fidelity approach coupling parameter space reduction and non-intrusive POD with application to structural optimization of passenger ship hulls. Int J Numer Methods Eng. 2023;124(5):1193–210. https://doi.org/10.1002/nme.7159.
5. Forrester AI, Sóbester A, Keane AJ. Multi-fidelity optimization via surrogate modelling. Proc Royal Soc A Math Phys Eng Sci. 2007;463(2088):3251–69. https://doi.org/10.1098/rspa.2007.1900.
6. Ng LW, Willcox KE. Multifidelity approaches for optimization under uncertainty. Int J Numer Methods Eng. 2014;100(10):746–72. https://doi.org/10.1002/nme.4761.
7. Perdikaris P, Raissi M, Damianou A, Lawrence ND, Karniadakis GE. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. Proc Royal Soc A. 2017;473(2198):20160751. https://doi.org/10.1098/rspa.2016.0751.
8. Raissi M, Perdikaris P, Karniadakis GE. Inferring solutions of differential equations using noisy multi-fidelity data. Journal of Computational Physics. 2017;335:736–46. https://doi.org/10.1016/j.jcp.2017.01.060.
9. Romor F, Tezzele M, Mrosek M, Othmer C, Rozza G. Multi-fidelity data fusion through parameter space reduction with applications to automotive engineering. arXiv preprint arXiv:2110.14396 (Submitted, 2021).
10. Zhang X, Xie F, Ji T, Zhu Z, Zheng Y. Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. Comput Methods Appl Mech Eng. 2021;373: 113485. https://doi.org/10.1016/j.cma.2020.113485.
11. Meng X, Karniadakis GE. A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems. J Comput Phys. 2020;401: 109020. https://doi.org/10.1016/j.jcp.2019.109020.
12. Guo M, Manzoni A, Amendt M, Conti P, Hesthaven JS. Multi-fidelity regression using artificial neural networks: efficient approximation of parameter-dependent output quantities. Comput Methods Appl Mech Eng. 2022;389: 114378. https://doi.org/10.1016/j.cma.2021.114378.
13. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nature Machine Intelligence. 2021;3(3):218–29.
14. Lin G, Moya C, Zhang Z. B-DeepONet: An enhanced Bayesian DeepONet for solving noisy parametric PDEs using accelerated replica exchange SGLD. J Comput Phys. 2023;473: 111713.
15. Lu L, Pestourie R, Johnson SG, Romano G. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. arXiv preprint arXiv:2204.06684 2022.
16. Howard AA, Perego M, Karniadakis GE, Stinis P. Multifidelity deep operator networks. arXiv preprint arXiv:2204.09157 2022.
17. Wang S, Bhouri MA, Perdikaris P. Fast PDE-constrained optimization via self-supervised operator learning. arXiv preprint arXiv:2110.13297 2021.
18. Meng X, Babaee H, Karniadakis GE. Multi-fidelity Bayesian neural networks: algorithms and applications. J Comput Phys. 2021;438: 110361. https://doi.org/10.1016/j.jcp.2021.110361.
19. Hart J, Waanders BvB. Hyper-differential sensitivity analysis with respect to model discrepancy: mathematics and computation. arXiv preprint arXiv:2210.09037 2022.
20. Hart J, Waanders BvB. Hyper-differential sensitivity analysis with respect to model discrepancy: Calibration and optimal solution updating. arXiv preprint arXiv:2210.09044 2022.
21. Farcas 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. arXiv preprint arXiv:2211.10835 2022.
22. Benner P, Ohlberger M, Patera A, Rozza G, Urban K. Model reduction of parametrized systems. MS&A series, vol. 17. Springer, Berlin; 2017.
23. Chinesta F, Huerta A, Rozza G, Willcox K. Model reduction methods. In: Stein E, de Borst R, Hughes TJR, editors. Encyclopedia of computational mechanics. 2nd ed. Hoboken: Wiley; 2017. p. 1–36.
24. Rozza G, Stabile G, Ballarin F. Advanced reduced order methods and applications in computational fluid dynamics. Soc Indu Appl Math. 2022. https://doi.org/10.1137/1.9781611977257.
25. Morelli UE, Barral P, Quintela P, Rozza G, Stabile G. A numerical approach for heat flux estimation in thin slabs continuous casting molds using data assimilation. Int J Numer Methods Eng. 2021;122(17):4541–74.
26. Tezzele M, Demo N, Stabile G, Mola A, Rozza G. Enhancing CFD predictions in shape design problems by model and parameter space reduction. Adv Model Simul Eng Sci. 2020. https://doi.org/10.1186/s40323-020-00177-y.
27. Benner P, Sachs E, Volkwein S. Model order reduction for PDE constrained optimization. Trends in PDE constrained optimization, 2014;303–326.
28. Amsallem D, Zahr M, Choi Y, Farhat C. Design optimization using hyper-reduced-order models. Struct Multidiscipl Optim. 2015;51(4):919–40. https://doi.org/10.1007/s00158-014-1183-y.
29. Zahr MJ, Farhat C. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. Int J Numer Methods Eng. 2015;102(5):1111–35. https://doi.org/10.1002/nme.4770.
30. Tezzele M, Salmoiraghi F, Mola A, Rozza G. Dimension reduction in heterogeneous parametric spaces with application to naval engineering shape design problems. Adv Model Simul Eng Sci. 2018;5(1):25. https://doi.org/10.1186/s40323-018-0118-3.
31. Demo N, Tezzele M, Rozza G. A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems. SIAM J Sci Comput. 2021;43(3):831–53. https://doi.org/10.1137/20M1345219.
32. Demo N, Tezzele M, Mola A, Rozza G. Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing. J Marine Sci Eng. 2021;9(2):185. https://doi.org/10.3390/jmse9020185.

33. Ghattas O, Willcox K. Learning physics-based models from data: perspectives from inverse problems and model reduction. Acta Numerica. 2021;30:445–554. https://doi.org/10.1017/S0962492921000064.
34. Ivagnes A, Demo N, Rozza G. Towards a machine learning pipeline in reduced order modelling for inverse problems: neural networks for boundary parametrization, dimensionality reduction and solution manifold approximation. arXiv preprint arXiv:2210.14764 2022.
35. Pichi F, Strazzullo M, Ballarin F, Rozza G. Finite Element-Based Reduced Basis Method in Computational Fluid Dynamics. In: Rozza, G., Stabile, G., Ballarin, F. (eds.) Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics. CS&E Series, pp. 13–58. SIAM Press, 2022. Chap. 2. https://doi.org/10.1137/1.9781611977257.ch2.
36. Qian E, Kramer B, Peherstorfer B, Willcox K. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. Physica D: Nonlinear Phenomena. 2020;406: 132401. https://doi.org/10.1016/j.physd.2020.132401.
37. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders WHA, Silveira LM (eds.): Volume 1: System- and Data-Driven Methods and Algorithms. De Gruyter, Berlin, Boston 2021. https://doi.org/10.1515/9783110498967.
38. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders WHA, Silveira LM, editors. Snapshot-based methods and algorithms, vol. 2. Berlin, Boston: De Gruyter; 2021. https://doi.org/10.1515/9783110671490.
39. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders WHA, Silveira LM, editors. Applications, vol. 3. Boston: De Gruyter; 2021. https://doi.org/10.1515/9783110499001.
40. Papapicco D, Demo N, Girfoglio M, Stabile G, Rozza G. The neural network shifted-proper orthogonal decomposition: a machine learning approach for non-linear reduction of hyperbolic equations. Comput Methods Appl Mech Eng. 2022;392: 114687. https://doi.org/10.1016/j.cma.2022.114687.
41. Reiss J, Schulze P, Sesterhenn J, Mehrmann V. The shifted proper orthogonal decomposition: a mode decomposition for multiple transport phenomena. SIAM J Sci Comput. 2018;40(3):1322–44.
42. Carere G, Strazzullo M, Ballarin F, Rozza G, Stevenson R. A weighted POD-reduction approach for parametrized PDE-constrained optimal control problems with random inputs and applications to environmental sciences. Comput Math Appl. 2021;102:261–76.
43. Venturi L, Ballarin F, Rozza G. A weighted POD method for elliptic PDEs with random inputs. J Sci Comput. 2019;81(1):136–53. https://doi.org/10.1007/s10915-018-0830-7.
44. Everson R, Sirovich L. Karhunen-Loève procedure for Gappy data. JOSA A. 1995;12(8):1657–64. https://doi.org/10.1364/JOSAA.12.001657.
45. Willcox K. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. Comput Fluids. 2006;35(2):208–26. https://doi.org/10.1016/j.compfluid.2004.11.006.
46. Bui-Thanh T, Damodaran M, Willcox K. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. AIAA J. 2004;42(8):1505–16. https://doi.org/10.2514/1.2159.
47. Mainini L, Willcox K. Surrogate modeling approach to support real-time structural assessment and decision making. AIAA J. 2015;53(6):1612–26. https://doi.org/10.2514/1.J053464.
48. Bright I, Lin G, Kutz JN. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. Phys Fluids. 2013;25(12): 127102. https://doi.org/10.1063/1.4836815.
49. Brunton SL, Tu JH, Bright I, Kutz JN. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. SIAM J Appl Dyn Syst. 2014;13(4):1716–32. https://doi.org/10.1137/130949282.
50. Kutz JN, Sargsyan S, Brunton SL. Leveraging sparsity and compressive sensing for reduced order modeling. In: Benner, P., Ohlberger, M., Patera, A., Rozza, G., Urban, K. (eds.) Model Reduction of Parametrized Systems. MS&A, vol. 17, pp. 301–315. Springer, Cham 2017. https://doi.org/10.1007/978-3-319-58786-8_19.
51. Adrian RJ. On the role of conditional averages in turbulence theory. In: Zakin JL, Patterson GK (eds.) Turbulence in liquids: Proceedings of the 4th Biennial Symposium on Turbulence in Liquids, pp. 323–332. University of Missouri–Rolla; 1975.
52. Nair NJ, Goza A. Leveraging reduced-order models for state estimation using deep learning. J Fluid Mech. 2020. https://doi.org/10.1017/jfm.2020.409.
53. Wang Y, Yu B, Cao Z, Zou W, Yu G. A comparative study of pod interpolation and pod projection methods for fast and accurate prediction of heat transfer problems. Int J Heat Mass Transfer. 2012;55(17–18):4827–36. https://doi.org/10.1016/j.ijheatmasstransfer.2012.04.053.
54. Tezzele M, Demo N, Stabile G, Rozza G. nonintrusive data-driven reduced order models in computational fluid dynamics. In: Rozza, G., Stabile, G., Ballarin, F. (eds.) Advanced reduced order methods and applications in computational fluid dynamics. CS&E Series. SIAM Press, 2022. Chap. 9. https://doi.org/10.1137/1.9781611977257.ch9.
55. Gadalla M, Cianferra M, Tezzele M, Stabile G, Mola A, Rozza G. On the comparison of LES data-driven reduced order approaches for hydroacoustic analysis. Comput Fluids. 2021;216: 104819. https://doi.org/10.1016/j.compfluid.2020.104819.
56. Demo N, Tezzele M, Rozza G. A non-intrusive approach for reconstruction of POD modal coefficients through active subspaces. Comptes Rendus Mécanique de l'Académie des Sciences. 2019;347(11):873–81. https://doi.org/10.1016/j.crme.2019.11.012.
57. Xie X, Mohebujjaman M, Rebholz LG, Iliescu T. Data-driven filtered reduced order modeling of fluid flows. SIAM J Sci Comput. 2018;40(3):834–57. https://doi.org/10.1137/17M1145136.
58. Amsallem D, Zahr MJ, Farhat C. Nonlinear model order reduction based on local reduced-order bases. Int J Numer Methods Eng. 2012;92(10):891–916. https://doi.org/10.1002/nme.4371.
59. Alla A, Kutz JN. Nonlinear model order reduction via dynamic mode decomposition. SIAM J Sci Comput. 2017;39(5):778–96. https://doi.org/10.1137/16M105930.
60. Kramer B, Willcox KE. Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition. AIAA J. 2019;57(6):2297–307. https://doi.org/10.2514/1.J057791.
61. San O, Maulik R. Neural network closures for nonlinear model order reduction. Adv Comput Math. 2018;44:1717–50. https://doi.org/10.1007/s10444-018-9590-z.
62. Geelen R, Wright S, Willcox K. Operator inference for non-intrusive model reduction with quadratic manifolds. Comput Methods Appl Mech Eng. 2023;403: 115717. https://doi.org/10.1016/j.cma.2022.115717.

63. Meneghetti L, Shah N, Girfoglio M, Demo N, Tezzele M, Lario A, Stabile G, Rozza G. A Deep Learning Approach to Improving Reduced Order Models. In: Rozza, G., Stabile, G., Ballarin, F. (eds.) Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics. CS&E Series. SIAM Press; 2022. Chap. 20. https://doi.org/10.1137/1.9781611977257.ch20.
64. Little C, Farhat C. Nonlinear Projection-Based Model Order Reduction in the Presence of Adaptive Mesh Refinement. In: AIAA SCITECH 2023 Forum 2023. https://doi.org/10.2514/6.2023-2682.
65. Manzoni A, Negri F, Quarteroni A. Dimensionality reduction of parameter-dependent problems through proper orthogonal decomposition. Ann Math Sci Appl. 2016;1(2):341–77.
66. Cueto E, Chinesta F, Huerta A. Model order reduction based on proper orthogonal decomposition. Separated representations and PGD-based model reduction: fundamentals and applications, 2014;1–26.
67. Volkwein S. Proper orthogonal decomposition: theory and reduced-order modelling. Lecture Notes Univ Konstanz. 2013;4(4):1–29.
68. Brunton SL, Kutz JN. Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge: Cambridge University Press; 2019.
69. Manohar K, Brunton BW, Kutz JN, Brunton SL. Data-driven sparse sensor placement for reconstruction: demonstrating the benefits of exploiting known patterns. IEEE Control Syst Mag. 2018;38(3):63–86. https://doi.org/10.1109/MCS.2018.2810460.
70. Yildirim B, Chryssostomidis C, Karniadakis G. Efficient sensor placement for ocean measurements using low-dimensional concepts. Ocean Model. 2009;27(3–4):160–73. https://doi.org/10.1016/j.ocemod.2009.01.001.
71. Otto SE, Rowley CW. Inadequacy of linear methods for minimal sensor placement and feature selection in nonlinear systems: a new approach using secants. J Nonlinear Sci. 2022;32(5):1–51. https://doi.org/10.1007/s00332-022-09806-9.
72. de Silva BM, Manohar K, Clark E, Brunton BW, Kutz JN, Brunton SL. PySensors: a Python package for sparse sensor placement. J Open Source Softw. 2021;6(58):2828. https://doi.org/10.21105/joss.02828.
73. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. PyTorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems 2019;32:8024–8035. Curran Associates, Inc.
74. Demo N, Tezzele M, Rozza G. EZyRB: Easy Reduced Basis method. J Open Source Softw. 2018;3(24):661. https://doi.org/10.21105/joss.00661.
75. Logg A, Mardal K-A, Wells G. Automated solution of differential equations by the finite element method: The FEniCS Book, vol. 84. Berlin: Springer; 2012.
76. Benamara T, Breitkopf P, Lepot I, Sainvitu C. Multi-fidelity extension to non-intrusive proper orthogonal decomposition based surrogates. In: Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016), 2016:4129–4145.
77. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011:315–323. JMLR Workshop and Conference Proceedings.
78. Ballarin F, Manzoni A, Quarteroni A, Rozza G. Supremizer stabilization of POD-Galerkin approximation of parametrized steady incompressible Navier-Stokes equations. Int J Numer Methods Eng. 2015;102(5):1136–61. https://doi.org/10.1002/nme.4772.
79. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, 2015:1026–1034.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.