MOX-Report No. 16/2021

Non intrusive reduced order modeling of parametrized PDEs by kernel POD and neural networks

Salvador, M.; Dede', L.; Manzoni, A.

NON INTRUSIVE REDUCED ORDER MODELING OF PARAMETRIZED PDES BY KERNEL POD AND NEURAL NETWORKS

Matteo Salvador
MOX, Department of Mathematics
Politecnico di Milano
P.zza Leonardo da Vinci 32, 20133 Milan, Italy
`matteo1.salvador@polimi.it`

Luca Dede'
MOX, Department of Mathematics
Politecnico di Milano
P.zza Leonardo da Vinci 32, 20133 Milan, Italy
`luca.dede@polimi.it`

Andrea Manzoni
MOX, Department of Mathematics
Politecnico di Milano
P.zza Leonardo da Vinci 32, 20133 Milan, Italy
`andrea1.manzoni@polimi.it`

ABSTRACT

We propose a nonlinear reduced basis method for the efficient approximation of parametrized partial differential equations (PDEs), exploiting kernel proper orthogonal decomposition (KPOD) for the generation of a reduced-order space and neural networks for the evaluation of the reduced-order approximation. In particular, we use KPOD in place of the more classical POD, on a set of high-fidelity solutions of the problem at hand to extract a reduced basis. This method provides a more accurate approximation of the snapshots' set featuring a lower dimension, while maintaining the same efficiency as POD. A neural network (NN) is then used to find the coefficients of the reduced basis by following a supervised learning paradigm and shown to be effective in learning the map between the time/parameter values and the projection of the high-fidelity snapshots onto the reduced space. In this NN, both the number of hidden layers and the number of neurons vary according to the intrinsic dimension of the differential problem at hand and the size of the reduced space. This adaptively built NN attains good performances in both the learning and the testing phases. Our approach is then tested on two benchmark problems, a one-dimensional wave equation and a two-dimensional nonlinear lid-driven cavity problem. We finally compare the proposed KPOD-NN technique with a POD-NN strategy, showing that KPOD allows a reduction of the number of modes that must be retained to reach a given accuracy in the reduced basis approximation. For this reason, the NN built to find the coefficients of the KPOD expansion is smaller, easier and less computationally demanding to train than the one used in the POD-NN strategy.

1 Introduction

Reduced order modeling (ROM) techniques represent a very efficient approach for the numerical approximation of problems involving the repeated solution of differential equations arising from engineering and applied sciences [10, 19, 27, 29, 33, 36, 37]. Their aim is to replace the original large-dimension numerical problem, which is typically called either high-fidelity approximation or full order model (FOM), by a reduced problem of substantially smaller dimension, to provide a fast and reliable approximation to the PDE solution for each newly queried parameter instance.

Depending on the context, there are several strategies to generate the reduced problem from the high-fidelity one. The approach followed by the reduced basis (RB) method consists in the projection of the high-fidelity problem upon a low-dimensional subspace made of specially selected basis functions, built from a set of high-fidelity solutions corresponding to suitably chosen parameters, e.g., through proper orthogonal decomposition (POD) [31]. The dimension of the reduced model should be in principle fairly lower than the FOM one. POD is an example of linear dimensionality reduction technique. Such a technique might show poor performances when dealing with highly nonlinear PDEs, because of the need of a huge number of POD modes to reach a desired accuracy of the reduced order approximation. Nonlinear dimensionality reduction techniques, such as kernel proper orthogonal decomposition (KPOD), have become popular also in the field of reduced order modeling for parametrized PDEs due to their better capability of spanning low-dimensional nonlinear trial manifolds [42, 43]. By exploiting the better representation power of nonlinear maps, these approaches condense better information from the underlying PDE keeping an even smaller reduced dimension than POD [38]. However, generating the reduced order approximation efficiently when dealing with nonlinear dimensionality reduction techniques might be far from being trivial. KPOD can be either used to perform the forward mapping, i.e. nonlinear dimensionality reduction, or the backward mapping, which is also called *pre-image* reconstruction [12]. Focusing on the forward mapping, KPOD projects the high-fidelity solutions by means of a nonlinear map to a high dimensional space (called *feature space*), where we are likely to obtain linear separability: then, by performing a dimensionality reduction, KPOD maps these data onto a reduced space that is in principle very similar to the space spanned by the most relevant modes in POD [38]. By employing a suitable *kernel trick* – as usually done when dealing with kernel principal component analysis (KPCA) [38] – we do not need to build the (very high-dimensional) feature space explicitly, finally relying on a suitably modified version of the snapshots correlation (or Gram) matrix, involving the evaluation of a bilinear kernel function instead of classical inner products. An important aspect, in this respect, is to find a suitable strategy to map the reduced solution back onto the high-fidelity space – a task that, in the case of POD, would simply entail the left-multiplication of the reduced approximation by the matrix collecting the vectors representing the POD modes [31]. In the case of KPOD, for instance, this task would involve the minimization of a discrepancy functional. The main motivations rely on the cost entailed by this stage, and the impossibility to provide a Galerkin projection to generate the (low-dimensional) reduced-order problem, which is solved for any new parameter instance.

For all these reasons, we propose to rely on neural network (NN) regression to determine, in a non intrusive way, the reduced order approximation for any new parameter instance, once a reduced basis has been built through KPOD. Thanks to their high representational power and flexibility [1, 8, 23, 44], neural networks are increasingly employed in the numerical approximation of PDEs [10, 19, 34, 35, 36]. Several recent works have shown possible applications of neural networks to parametrized PDEs – thanks to their approximation capabilities, their extremely favorable computational performances during online testing phases, and their relative easiness of implementation – both from a theoretical [2, 3, 4, 24, 39] and a computational standpoint [10, 19, 28, 36]. A priori characterizations of the NN architecture complexity in terms of the accuracy of the reconstructed solution have been reported in [24], showing that the NN complexity should scale with the intrinsic dimension of the PDE solution manifold, i.e. with the size of the reduced basis extracted from the PDE under investigation to properly capture its behavior. From a computational standpoint, feedforward NNs and autoencoders, have been employed in several ways to determine the reduced order approximation in a data-driven and less intrusive way (avoiding, e.g., the computational cost entailed by projection-based ROMs), but still relying on a linear trial manifold built, e.g., through POD. In [16, 17, 19, 37, 41] the solution of nonlinear, time-dependent ROMs, for any new parameter instance, has been replaced by the evaluation of NN-based regression models, built using a fixed size feedforward neural network. Few attempts have been made in order to model the reduced order manifold where the approximation is sought (avoiding, e.g., the linear superimposition of POD modes) through NNs, see, e.g., [14, 26]. In the framework of deep learning, variational autoencoders have been combined with a fixed size feedforward NN and eventually POD in [9, 10, 11] to build non intrusive ROMs similarly to [19], capturing however more details by using the intrinsic dimension of a differential problem for the reduced order approximation, that is, the number of parameters (plus one, to take into account the time coordinate) the solution depends on.

In this work, we propose a non-intrusive ROM technique that combines KPOD with an adaptively built NN, i.e. a NN where the number of layers and neurons is automatically adjusted to the complexity of the parametrized PDE at hand. This complexity is captured by the dimension of the reduced basis extracted from the FOM snapshots through KPOD. This latter technique is, at the same time, as efficient as the POD, yet capable of collecting most of the information in a small number of modes of the kernel matrix, thus performing data compression even more efficiently than POD.

The paper is structured as follows: in Section 2 we describe our ROM method involving a combined use of KPOD and NN, and we apply it to a general parametrized PDEs. In Section 3 we introduce the mathematical models on which our metholodology will be tested, i.e. wave equation and Navier-Stokes equations. In Section 4 we provide both space and time discretizations for the two mentioned FOM, which is essential to get high-fidelity snapshots. In Section 5 we

show the numerical results related to the KPOD-NN technique on the two test cases. Finally, we draw conclusions and possible future developments in Section 6.

2   Reduced basis methods for parametrized PDEs using neural networks

We describe in this section the proposed method, considering for the sake of generality the solution of a nonlinear, time-dependent parametrized PDE. Let us denote by $\Omega \subset \mathbb{R}^d$, $d \geq 1$ a given domain, and by $\mathcal{P} \subset \mathbb{R}^m$ the parameter set, with $m \geq 1$. For each $\boldsymbol{\mu} \in \mathcal{P}$, $\boldsymbol{x} \in \Omega$, $t \in [0, T]$, let us denote by $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}) : \mathbb{R}^{m+d+1} \to \mathbb{R}^l$ the scalar ($l = 1$) or vector ($l > 1$) field, solution of the following problem:

$$
\begin{cases}
\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}(t; \boldsymbol{\mu})\boldsymbol{u} + \boldsymbol{N}(\boldsymbol{u}, t; \boldsymbol{\mu}) = \boldsymbol{f}(t; \boldsymbol{\mu}) & \text{in } \Omega \times (0, T) \\
+\text{boundary conditions} & \text{in } \partial\Omega \times (0, T) \\
+\text{initial conditions} & \text{in } \Omega \times \{t = 0\}.
\end{cases}
\tag{1}
$$

Problem (1) can be either linear (if $\boldsymbol{N}(\boldsymbol{u}, t; \boldsymbol{\mu}) = \boldsymbol{0}$) or nonlinear (if $\boldsymbol{N}(\boldsymbol{u}, t; \boldsymbol{\mu}) \neq \boldsymbol{0}$); here $\boldsymbol{f}$ acts as a forcing term.

Our final goal is the efficient numerical approximation of problem (1) for different values of the parameters vector $\boldsymbol{\mu}$. Our starting point is a full order model (FOM) obtained by introducing the semi-discrete space approximation of (1), of the following form:

$$
\begin{cases}
\dfrac{\partial \boldsymbol{u}_h}{\partial t} + \boldsymbol{A}_h(t; \boldsymbol{\mu})\boldsymbol{u}_h + \boldsymbol{N}_h(\boldsymbol{u}_h, t; \boldsymbol{\mu}) = \boldsymbol{f}_h(t; \boldsymbol{\mu}) & \text{for } t \in (0, T) \\
+\text{initial conditions} & \text{for } t = 0.
\end{cases}
\tag{2}
$$

This latter problem is generated by introducing a suitable computational grid over the domain Ω, and an (e.g., finite element) approximation $\boldsymbol{u}_h(\boldsymbol{x}, t; \boldsymbol{\mu})$ of $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu})$ depending on a set of N_h degrees of freedom (DOFs); for this reason, with a slight abuse of notation, we denote by $\boldsymbol{u}_h(t; \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ the DOFs vector corresponding to the solution; $\boldsymbol{A}_h(t; \boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ defines the stiffness matrix obtained by the discretization of the (linear) operator $\boldsymbol{A}$, $\boldsymbol{N}_h(\boldsymbol{u}_h, t; \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is the vector obtained by discretizing the nonlinear operator $\mathbf{N}$, and $\boldsymbol{f}_h(t; \boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ corresponds to the discrete source term. We then consider a partition of the time interval $[0, T]$ into $N_T + 1$ equally spaced values $\{t_0 = 0, t_1, \ldots, t_j, \ldots, t_{N_T} = T\}$, and approximate the time-derivative using suitable finite difference schemes, finally yielding the fully discretized problem, which provides our FOM. For each $\boldsymbol{\mu} \in \mathcal{P}$, the FOM solution at time t_j is then denoted by $\boldsymbol{u}_h^j(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$, $j \in \{0, 1, \ldots, N_T\}$.

Several reduced order models (ROMs) for parametrized PDEs are built by sampling the parameter space and computing a set of *snapshots*, that is, FOM solutions for sampled values $\mathcal{P}_h = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_i, \ldots, \boldsymbol{\mu}_m\} \subset \mathcal{P}$ of the parameters. For the case at hand, the snapshots matrix collecting the computed FOM solutions includes both time and parameter dependencies, and reads:

$$
\mathbb{S} = [\boldsymbol{u}_h^0(\boldsymbol{\mu}_1) \,|\, \boldsymbol{u}_h^1(\boldsymbol{\mu}_1) \,|\, \ldots \,|\, \boldsymbol{u}_h^{N_T}(\boldsymbol{\mu}_1) \,|\, \ldots \,|\, \boldsymbol{u}_h^0(\boldsymbol{\mu}_m) \,|\, \boldsymbol{u}_h^1(\boldsymbol{\mu}_m) \,|\, \ldots \,|\, \boldsymbol{u}_h^{N_T}(\boldsymbol{\mu}_m)] \in \mathbb{R}^{N_h \times N_s},
\tag{3}
$$

with $N_s = m \times (N_T + 1)$. For the sake of simplicity, we suppose that $\mathbb{S}$ is built using all the snapshots in time generated by solving the FOM with a fixed time-step Δt, but eventually only a subset of those solutions can be considered, and, to make the notation lighter, we denote the snapshots matrix by:

$$
\mathbb{S} = [\mathbf{s}_1 \,|\, \ldots \,|\, \mathbf{s}_{N_s}].
$$

To build the proposed ROM strategy, we start from a projection-based framework, exploiting the reduced basis method for parametrized PDEs. We denote by $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\} \subset \mathbb{R}^{N_h}$, with $n \ll N_h$, a set of reduced basis functions spanning a low-dimensional subspace V_n, in which the reduced order approximation of the problem solution is sought. Such a subspace can be built in two different ways, either linearly, for instance through proper orthogonal decomposition (POD) [19, 41], or nonlinearly (see, e.g., [10, 11, 25]). We recall that, in the case of POD, the basis functions can be equivalently obtained either as left singular vectors of the snapshots matrix $\mathbb{S}$, or as eigenvectors (up to a normalization factor) of the Gram matrix $\mathbb{C} = \mathbb{S}^\top \mathbb{S}$.

In this paper, we propose to rely on KPOD [38], which is the nonlinear counterpart of proper orthogonal decomposition (POD), as an efficient and effective alternative to POD. Compared to POD, KPOD has the additional advantage to be able to deal with otherwise linearly inseparable data. In particular, KPOD projects, through a nonlinear map, the FOM snapshots to a high dimensional space, where we are likely to obtain linear separability (according to Cover's theorem [7]), as depicted in Figure 1.
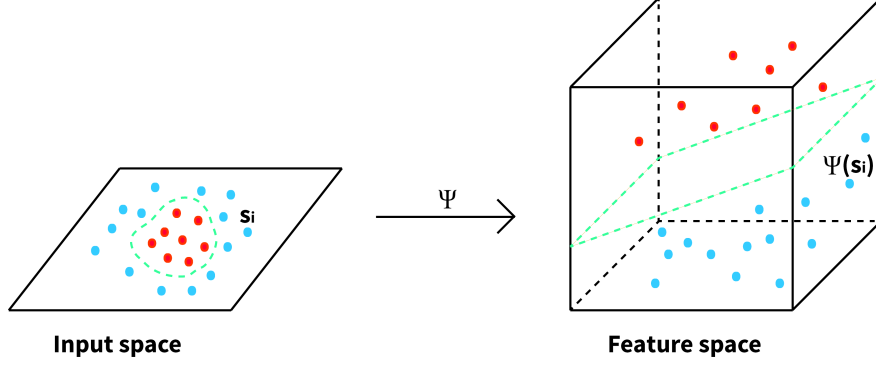
Figure 1: Example of application of the Cover's theorem. Linear inseparable data become separable if projected upon a higher dimensional space through a map $\boldsymbol{\Psi}(\cdot)$.

In principle, we should employ an arbitrary transformation $\boldsymbol{\Psi} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_Z}$ for some very large dimension $N_Z \gg N_h$. Such a transformation is meant to flatten the nonlinear solution manifold $\mathcal{S}_h = \{\boldsymbol{u}_h(t; \boldsymbol{\mu}), \ t \in [0, T], \ \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_h}$ where the FOM solution is sought. In other words, the manifold obtained as $\mathcal{Z}_h = \boldsymbol{\Psi}(\mathcal{S}_h) = \{\boldsymbol{U}_h(t; \boldsymbol{\mu}) = \boldsymbol{\Psi}(\boldsymbol{u}_h(t; \boldsymbol{\mu})), \ t \in [0, T], \ \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_Z}$ should be more readily approximated through linear subspaces. To this aim, we would need to apply POD to the $N_Z \times N_s$ matrix containing the transformed snapshots:

$$\mathbb{S}_{\boldsymbol{\Psi}} = [\boldsymbol{\Psi}(\mathbf{s}_1) \,|\, \ldots \,|\, \boldsymbol{\Psi}(\mathbf{s}_{N_s})] \in \mathbb{R}^{N_Z \times N_s}$$

However, we could also apply POD to the Gram matrix $\tilde{\mathbb{C}}_{\boldsymbol{\Psi}} = \mathbb{S}_{\boldsymbol{\Psi}}^\top \mathbb{S}_{\boldsymbol{\Psi}}$, which is of size $N_s \times N_s$ as $\mathbb{C} = \mathbb{S}^\top \mathbb{S}$. In the context of KPOD, the so-called *kernel trick* is applied, i.e. the matrix $\mathbb{S}_{\boldsymbol{\Psi}}$ is directly defined by introducing a bivariate symmetric form $\kappa : \mathbb{R}^{N_h} \times \mathbb{R}^{N_h} \to \mathbb{R}$, also referred to as *kernel function*, rather than the map $\boldsymbol{\Psi}$, to avoid the computation of the components of the matrix $\tilde{\mathbb{C}}_{\boldsymbol{\Psi}}$ according to the definition, that is:

$$(\tilde{\mathbb{C}}_{\boldsymbol{\Psi}})_{ij} = \sum_{k=1}^{N_Z} (\mathbb{S}_{\boldsymbol{\Psi}})_{ki} (\mathbb{S}_{\boldsymbol{\Psi}})_{kj} = (\boldsymbol{\Psi}(\mathbf{s}_i))^\top (\boldsymbol{\Psi}(\mathbf{s}_j)) \qquad i, j = 1, \ldots, N_s.$$

Thanks to the kernel function, we can instead define:

$$(\tilde{\mathbb{C}})_{ij} = \kappa(\mathbf{s}_i, \mathbf{s}_j) \qquad i, j = 1, \ldots, N_s,$$

which takes the name of kernel (similarity) matrix. A common choice is to use the squared exponential (or radial basis function) kernel function, which yields the following kernel matrix $\mathbb{K}$:

$$\mathbb{K}(\mathbf{s}_i, \mathbf{s}_j) = \exp\left(-\gamma \|\mathbf{s}_i - \mathbf{s}_j\|_2^2\right) \in \mathbb{R}^{N_s \times N_s} \qquad i, j = 1, \ldots, N_s. \tag{4}$$

Here $\gamma > 0$ is a hyperparameter of the model, whereas $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are two columns of the snapshots matrix $\mathbb{S}$, i.e. two different samples in the time domain and in the parameters space.

One of the advantages of using KPOD is that it does not work in the high-dimensional latent space directly. Indeed, it computes the projections of the FOM solutions onto the principal components, but not the principal components themselves (as done instead in POD, or PCA). To evaluate these projections, first, we compute the eigenvectors and eigenvalues of (4):

$$\mathbb{K}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k \qquad k = 1, \ldots, N_s, \tag{5}$$

where we denote by $\mathbb{W} = [\boldsymbol{w}_1 \,|\, \ldots \,|\, \boldsymbol{w}_{N_s}] \in \mathbb{R}^{N_s \times N_s}$ the matrix collecting the eigenvectors of $\mathbb{K}$, and by $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_{N_s}) \in \mathrm{R}^{N_s \times N_s}$ the matrix of the corresponding eigenvalues. Then, by analogy with POD, we compute and collect the projected vectors:

$$\boldsymbol{p}_k = \frac{1}{\sigma_k} \mathbb{S} \boldsymbol{w}_k \qquad k = 1, \ldots, N_s \tag{6}$$

in the matrix $\mathbb{P} = [\boldsymbol{p}_1 \,|\, \ldots \,|\, \boldsymbol{p}_{N_s}] \in \mathbb{R}^{N_h \times N_s}$, with $\sigma_k = \sqrt{\lambda_k}$. In POD, $\mathbb{P}$ would be an orthogonal matrix. Conversely, in KPOD, we finally have to perform a reduced QR factorization of $\mathbb{P}$ to obtain an orthonormal basis [32]:

$$\left[\mathbb{Q}, \tilde{\mathbb{R}}\right] = \text{QR}(\mathbb{P}), \tag{7}$$

4

where $\mathbb{Q} \in \mathbb{R}^{N_h \times N_{hs}}$ is an orthogonal matrix, whereas $\tilde{\mathbb{R}} \in \mathbb{R}^{N_{hs} \times N_s}$ is an upper triangular matrix, being $N_{hs} = \min(N_h, N_s)$.

We select n columns of $\mathbb{Q}$ to get the (orthonormal) reduced basis $\mathbb{V} = [\boldsymbol{q}_1|\boldsymbol{q}_2|...|\boldsymbol{q}_n] \in \mathbb{R}^{N_h \times n}$. The dimension n where the truncation occurs is determined as follows:

$$\frac{\sum_{k=n+1}^{N_s} \sigma_k^2}{\sum_{k=1}^{N_s} \sigma_k^2} \leq \hat{\varepsilon}, \tag{8}$$

being $\hat{\varepsilon} > 0$ a given tolerance and σ_k^2 the square of the kth singular value. We will always use $\hat{\varepsilon} = 10^{-12}$ in the numerical results.

The process is summarized in Algorithm 1. So far, we have just described the forward mapping, i.e. the nonlinear dimensionality reduction process. Due to the nonlinear nature of this technique, it is not trivial to map an element from the reduced space (with dimension n) towards its *pre-image* in the input space of the high-fidelity solutions (with dimension N_h). We refer to this mathematical problem as KPOD backward mapping.

Once $\mathbb{V}$ is built, we obtain the reduced coefficients by means of the linear projection $\mathbb{V}^T \boldsymbol{u}_h(t; \boldsymbol{\mu})$ onto V_n. We introduce now a function $\boldsymbol{\Phi}$ which represents the connection between the space of parameters, wherein we assimilate the time independent variable to a parameter, and the projected coefficients, i.e:

$$\boldsymbol{\Phi}: \mathcal{T} \times \mathcal{P} \subset \mathbb{R}^{m+1} \to \mathbb{R}^n \quad \text{s.t.} \quad \boldsymbol{\Phi}: (t; \boldsymbol{\mu}) \to \mathbb{V}^T \boldsymbol{u}_h(t; \boldsymbol{\mu}). \tag{9}$$

Similarly to what is done in [41] for the POD-NN method, we propose to rely on NN to determine the reduced order approximation for any new parameter instance. This NN is able to perform a nonlinear regression for the KPOD-NN method, to learn an approximation $\boldsymbol{\Phi}_{NN}$ of the map $\boldsymbol{\Phi}$. The evaluation of $\boldsymbol{\Phi}_{NN}$ for a certain $(t, \boldsymbol{\mu})$ provides a reduced solution:

$$\boldsymbol{u}_n^{NN}(t; \boldsymbol{\mu}) = \boldsymbol{\Phi}_{NN}(t; \boldsymbol{\mu}), \tag{10}$$

and consequently we come back to the high-fidelity dimension through:

$$\boldsymbol{u}_h^{NN}(t; \boldsymbol{\mu}) = \mathbb{V}\boldsymbol{\Phi}_{NN}(t; \boldsymbol{\mu}) = \mathbb{V}\boldsymbol{u}_n^{NN}(t; \boldsymbol{\mu}). \tag{11}$$

To the best of our knowledge, a novelty is that we consider a NN where the number of hidden layers scales as $\lceil \log(n) \rceil$ and the size of each hidden layer depends on the dimension n of the reduced basis. We also have a first (input) layer with $N_I = m + 1$ units (for the $(t; \boldsymbol{\mu})$ values) and a final (output) layer of size $N_O = n$ (for each single component of the reduced solution $\mathbb{V}^T \boldsymbol{u}_h(t; \boldsymbol{\mu})$). Our NN is fully connected, i.e. each neuron of a specific layer is connected to all the neurons of the next layer only [19]. The strength of these interactions is defined by weights $\boldsymbol{W}_i$ and biases [15] (as shown in Figure 2), whose values change during the training process. In this way the NN provides a suitable lower dimensional representation of the specific FOM that we want to approximate. We use a Parametric Rectified Linear Unit (PReLU) as activation function of the hidden layers. This function is known to be more flexible than ReLU and LeakyReLU while leading better NN approximations [30]. PReLU reads:

$$\text{PReLU}(x) = \begin{cases} \alpha x & \text{in } x < 0, \\ x & \text{in } x \geq 0, \end{cases} \tag{12}$$

where $\alpha > 0$ is automatically optimized during the training phase.

The architecture is synthesized in Table 1. A schematic view of the NN is sketched in Figure 2.

Algorithm 1 KPOD as nonlinear dimensionality reduction technique for ROM.

---

function $\mathbb{V}$=KPOD($\mathbb{S}, \hat{\varepsilon}, \gamma$)

    Assemble $\mathbb{K}(\mathbf{s}_i, \mathbf{s}_j) = \exp\left(-\gamma ||\mathbf{s}_i - \mathbf{s}_j||_2^2\right) \quad \text{for} \quad i, j = 1, ..., N_s$.
    Perform an eigenvalue-eigenvector decomposition: $\mathbb{K}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k \quad \text{for} \quad k = 1, ..., N_s$.
    Compute $\mathbb{P} = \left[\boldsymbol{p}_1|\boldsymbol{p}_2|...|\boldsymbol{p}_{N_s}\right] \in \mathbb{R}^{N_h \times N_s}$, where $\boldsymbol{p}_k = \frac{1}{\sqrt{\lambda_k}}\mathbb{S}\boldsymbol{w}_k \quad \text{for} \quad k = 1, \ldots, N_s$.

    Reduced QR factorization of matrix $\mathbb{P}$: $\left[\mathbb{Q}, \tilde{\mathbb{R}}\right] = \text{QR}(\mathbb{P})$.

    Select n $\boldsymbol{q}_k$ vectors according to (8).
    Assemble $\mathbb{V} = [\boldsymbol{q}_1|\boldsymbol{q}_2|...|\boldsymbol{q}_n]$.

end function

For the sake of NN construction, in supervised learning, let $\mathcal{D}_b = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_b}$ be a building set with input-output pairs, which is split in different subsets: a training (equivalently learning) dataset $\mathcal{D}_{tr} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_{tr}}$ and a validation
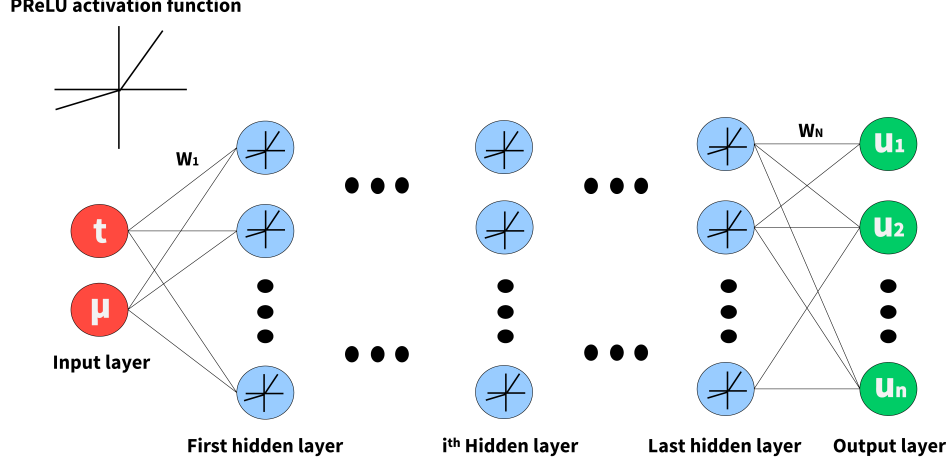
Figure 2: Example of fully connected feedforward neural network with PReLU activation functions, several hidden layers and an input layer made by time and one single parameter μ ($m = 1$). $\boldsymbol{W}_i$ indicates weights of the connections between the ith layer and the following one.

Layer	Number of neurons
Dense + PReLU activation (input)	$m + 1$
Dense + PReLU activation (hidden) 1	n
Dense + PReLU activation (hidden) 2	n
...	...
Dense + PReLU activation (hidden) $\lceil \log(n) \rceil$	n
Dense (output)	n

Table 1: NN architecture. n is the dimension of the reduced basis, whereas m is the length of the parameter vector. Glorot uniform initializer [13] is used to set the initial values of the weights of all layers.

one $\mathcal{D}_v = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_v}$, with $N_{tr} + N_v = N_b$. Inputs $\boldsymbol{x}_i$ are sampled from the space of parameters, whereas outputs $\boldsymbol{y}_i$ represent the corresponding reduced coefficients. We apply the so called K-fold cross-validation [18], where the building set is divided into K equal parts (here we use $K = 5$). In this respect, the network is trained K times, where $K - 1$ subdivisions of the building set act as learning set, whereas the last remaining subdivision is used to validate the model for a specific choice of the hyperparameters. Data located inside the building set are shuffled at the beginning of the K-fold procedure so that there is a high probability for the network to learn the reduced numerical solution along the entire time span $[0, T]$.

Thanks to K-fold cross-validation, we are able to perform both the training phase and the model selection. In particular, we investigate the effects of changing the number of layers and the growth/decay of the number of neurons of the layers in the network (e.g. linear, parabolic and hyperbolic). The selection of the best model among the different analyzed architectures is made through the comparison of their generalization error E_{gen}:

$$E_{gen} = \frac{1}{N_v} \sum_{k=1}^{K} \frac{\sqrt{\sum_{i=1}^{N_v} (\hat{\boldsymbol{y}}_i^k - \boldsymbol{y}_i)^2}}{\sqrt{\sum_{i=1}^{N_v} \boldsymbol{y}_i^2}}, \tag{13}$$

where $\hat{\boldsymbol{y}}_i^k$ is the prediction of the neural network on fold k associated with the output observation $\boldsymbol{y}_i$. We look for the network with the smallest generalization error. We trained different neural networks using input-output pairs coming from the benchmark problems that will be introduced in Section 3. As the generalization error was not significantly affected by the growth/decay strategy used to determine the number of neurons in a single layer, we selected a constant number of neurons per layer. The selected architecture is reported in Table 1, which provided - among the tested architectures - the lowest generalization error. In order to adapt the network complexity to the dimension n, we scaled the number of layers as $\lceil \log(n) \rceil$.

6

Once the architecture is fixed and a NN is built and trained, we test its performances, feeding the network with a new dataset (called test set) $\mathcal{D}_{test} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N_{test}}$, which is made by unseen observations. In this way, we assess the learning capability of the network. All the abovementioned operations are implemented using the Keras Python Deep Learning library [5]. Our loss function is the discrete relative L^2 norm:

$$\mathcal{C}_j = \frac{\sqrt{\sum_{i=1}^{N_j}(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2}}{\sqrt{\sum_{i=1}^{N_j} \boldsymbol{y}_i^2}} \quad \text{with} \quad j \in \{tr, v\}, \tag{14}$$

The network is trained over different steps, which are called epochs, considering each time a small amount of observations of the training set, which are referred to as batches; here we consider $N_{batch} = 10$. A small value of batch size is known to avoid sharp local minima in the optimization process, improving the accuracy of the network [21]. We use the Adam stochastic optimizer [22] with AMSGrad variant, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, to find the optimal values of weights and bias. The initial learning rate lr is set to 0.01 for the wave equation and to 0.1 for the lid-driven cavity problem. This lets the stochastic gradient descent method better explore the landscape of local minima during the very first epochs of training. We consider a regularization of the weights so that the network does not overfit and is able to explore more possibilities in the training process. The final formulation of the loss function is reported here:

$$\mathcal{C}_j = \frac{\sqrt{\sum_{i=1}^{N_j}(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2}}{\sqrt{\sum_{i=1}^{N_j} \boldsymbol{y}_i^2}} + \Theta \sum_{l=1}^{N_l} \sum_o \sum_k w_{l,o,k}^2 \quad \text{with} \quad j \in \{tr, v\}, \tag{15}$$

where $\Theta = 0.01$ acts as regularization parameter, $w_{l,o,k}$ is the oth weight of the lth layer connected to the kth neuron of the following layer of the NN. The entire implementation of the offline and online stages of the KPOD-NN method is described in Algorithm 2. This method is non-intrusive, because it does not need to access system operators to perform the projection and the online evaluation of the feedforward neural network is independent of the high-fidelity numerical scheme. For the sake of simplicity, we present an approach in which the size of the reduced space plays a role only in the definition of the NN architecture. The underlying idea is that the higher the dimension of the reduced space the higher should be the complexity of the NN. Still, a dependence from the dimension n can be eventually introduced in other parameters of the NN, such as the regularization term Θ, the configuration of the optimizer, the total number of epochs and the batch size.

Algorithm 2 KPOD-NN ROM method for unsteady PDEs.

---

function $[\mathbb{V}, \boldsymbol{\Phi}_{NN}]$=KPOD-NN_OFFLINE($\mathcal{P}, \Omega, T, m, N_T, \hat{\varepsilon}, \gamma$)

    Generate $[t_0 = 0, t_1, ..., t_j, ..., t_{N_T} = T]$ and $\mathcal{P}_h = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_i, ..., \boldsymbol{\mu}_m\} \subset \mathcal{P}$.

    Compute high-fidelity numerical solutions $\boldsymbol{u}_h^j(\boldsymbol{\mu}_i)$, $j = 0, ..., N_T$, $i = 1, ..., m$.

    Set $\mathbb{S} = [\boldsymbol{u}_h^0(\boldsymbol{\mu}_1), \boldsymbol{u}_h^1(\boldsymbol{\mu}_1), ..., \boldsymbol{u}_h^{N_T}(\boldsymbol{\mu}_1), \boldsymbol{u}_h^0(\boldsymbol{\mu}_2), ..., \boldsymbol{u}_h^{N_T}(\boldsymbol{\mu}_m)]$.

    $\mathbb{V}$=KPOD($\mathbb{S}, \hat{\varepsilon}, \gamma$).

    Build and train the NN for the calculation of $\boldsymbol{\Phi}_{NN}$.

end function

function $\mathbb{V}$=KPOD-NN_ONLINE($(t_{new}; \boldsymbol{\mu}_{new}), \mathbb{V}, \boldsymbol{\Phi}_{NN}$)

    Evaluate the output $\boldsymbol{u}_n^{NN}(t_{new}; \boldsymbol{\mu}_{new}) = \boldsymbol{\Phi}_{NN}(t_{new}; \boldsymbol{\mu}_{new})$ of the trained NN for the input $(t_{new}; \boldsymbol{\mu}_{new})$.

    Map the solution to the high-fidelity space as $\boldsymbol{u}_h^{NN}(t_{new}; \boldsymbol{\mu}_{new}) = \mathbb{V}\boldsymbol{\Phi}_{NN}(t_{new}; \boldsymbol{\mu}_{new})$.

end function

---

3 Benchmark problems

We apply the method described in Section 2 to the following two differential problems, namely the wave equation and the Navier-Stokes equations. The first problem is set in 1D, whereas the second one will be solved in 2D.

3.1 Wave equation

The wave equation is a second-order hyperbolic linear PDE and reads:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = c^2 \dfrac{\partial^2 u}{\partial x^2} & \text{in } \Omega \times (0,T), \\ u(x,0) = A_0 \exp^{-\frac{(x-x_0)^2}{2\sigma^2}} & \text{in } \Omega \times \{0\}, \\ u(0,t) = 0 & t \in (0,T), \\ u(L,t) = 0 & t \in (0,T), \end{cases} \tag{16}$$

with $\Omega = (0,L)$. $A_0 \in [0.5,1]$, $x_0 \in [\frac{L}{3}, \frac{2L}{3}]$, $\sigma \in [0.5,1]$ and $c \in \mathbb{R}$, with $L = 4\pi$ and $T = 52$. We assign homogeneous Dirichlet boundary conditions at $x = 0$ and $x = L$. The initial condition, given by a Gaussian pulse, triggers the propagation of two waves in opposite directions. We consider intensity A_0, position x_0 and variability σ of the initial pulse as input parameters of the neural network, i.e. $u = u(x,t; A_0, x_0, \sigma)$.

3.2 Navier-Stokes equations

The Navier-Stokes equations model the flow of a viscous fluid, either incompressible or compressible, in a certain domain $\Omega \subset \mathbb{R}^2$ and in the time interval $(0,T)$. The unknowns in primitive variables are velocity and pressure, i.e $(\boldsymbol{u}, p)$. The corresponding dimensionless strong form reads [40]:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p - 2\nabla \cdot \left(\dfrac{1}{\mathbb{R}e} D(\boldsymbol{u}) \right) = \boldsymbol{f} & \text{in } \Omega \times (0,T), \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega \times (0,T), \\ \boldsymbol{u}(0) = \boldsymbol{u}_0 & \text{in } \Omega \times \{0\}, \\ \boldsymbol{\sigma}_f \boldsymbol{n} = \boldsymbol{h} & \text{on } \Gamma_N \times (0,T), \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \Gamma_D \times (0,T), \end{cases} \tag{17}$$

where $\mathbb{R}e$ is the Reynolds number, $D(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$ is the strain tensor, and $\boldsymbol{\sigma}_f = -pI + 2\mathbb{R}e^{-1}D(\boldsymbol{u})$ is the Cauchy stress tensor, being I the identity tensor. $\boldsymbol{f} : \Omega \times (0,T) \to \mathbb{R}^2$ indicates the body forces, $\boldsymbol{u}_0 : \Omega \to \mathbb{R}^2$. $\boldsymbol{h}$ and $\boldsymbol{g}$ denote respectively the vector fields for the Neumann boundary condition on Γ_N and the Dirichlet boundary condition on Γ_D, where $\Gamma_N, \Gamma_D \subseteq \partial\Omega$ with $\overline{\Gamma_D \cup \Gamma_N} = \partial\Omega$ and $\mathring{\Gamma}_D \cap \mathring{\Gamma}_N = \emptyset$. $\boldsymbol{n}$ indicates the outward directed unit vector normal to Γ_N.

The weak formulation of (17) reads [40]: given $V_g = \{\boldsymbol{v} \in [H^1(\Omega)]^2 : \boldsymbol{v}|_{\Gamma_D} = \boldsymbol{g}\}$, $V = \{\boldsymbol{v} \in [H^1(\Omega)]^2 : \boldsymbol{v}|_{\Gamma_D} = \boldsymbol{0}\}$, $Q = L^2(\Omega)$, $W_g = \{\boldsymbol{v} \in V_g : \nabla \cdot \boldsymbol{v} = 0\} \subset V_g$ and $W = \{\boldsymbol{v} \in V : \nabla \cdot \boldsymbol{v} = 0\} \subset V$ spaces of divergence-free functions of V_g and V respectively, find $\boldsymbol{u} = \boldsymbol{u}(t) \in W_g$ $\forall t \in (0,T)$ such that:

$$\begin{cases} m\left(\dfrac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{v} \right) + a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) + c(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) = F(\boldsymbol{v}) + H(\boldsymbol{v}) & \forall \boldsymbol{v} \in W, \\ b(\boldsymbol{u}, q) = 0 & \forall q \in Q, \\ \boldsymbol{u}(0) = \boldsymbol{u}_0 & \text{in } \Omega, \end{cases} \tag{18}$$

where $m\left(\dfrac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{v} \right) = \displaystyle\int_\Omega \dfrac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v} \, d\Omega$, $a(\boldsymbol{u}, \boldsymbol{v}) = \dfrac{2}{\mathbb{R}e} \displaystyle\int_\Omega D(\boldsymbol{u}) : D(\boldsymbol{v}) \, d\Omega$, $b(\boldsymbol{v}, p) = -\displaystyle\int_\Omega p \nabla \cdot \boldsymbol{v} \, d\Omega$, $c(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}) = \displaystyle\int_\Omega ((\boldsymbol{w} \cdot \nabla)\boldsymbol{u}) \cdot \boldsymbol{v} \, d\Omega$, while $F(\boldsymbol{v}) = \displaystyle\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, d\Omega$ and $H(\boldsymbol{v}) = \displaystyle\int_{\Gamma_N} \boldsymbol{h} \cdot \boldsymbol{v} \, d\Gamma$.

The Navier-Stokes equations can be rewritten in the streamfunction formulation [40]. We introduce the quotient space $X = \{[\phi] : \phi \in H^2(\Omega)\}$ of scalar functions in $H^2(\Omega)$ that differ up to a constant and, under the hypothesis of simply connected domain, we consider the unique potential $\phi \in X$ such that $\boldsymbol{v} = \boldsymbol{curl}\phi$, where $\boldsymbol{curl}(\cdot)$ operator is defined as follows:

$$\boldsymbol{curl} : X \to [H^1(\Omega)]^2 \quad \boldsymbol{curl}\phi = \left(\dfrac{\partial \phi}{\partial y}, -\dfrac{\partial \phi}{\partial x} \right) \tag{19}$$

The streamfunction weak formulation reads: given $\boldsymbol{\Phi}_g = \{\psi \in X : \boldsymbol{curl}\psi|_{\Gamma_D} = \boldsymbol{g}\}$ and $\boldsymbol{\Phi} = \{\psi \in X : \boldsymbol{curl}\psi|_{\Gamma_D} = \boldsymbol{0}\}$, find $\phi = \phi(t) \in \boldsymbol{\Phi}_g$ $\forall t \in (0,T)$ such that:

$$\begin{cases} \overline{m}\left(\dfrac{\partial \phi}{\partial t}, \psi \right) + \overline{a}(\phi, \psi) + \overline{c}(\phi, \phi, \psi) = \overline{F}(\psi) + \overline{H}(\psi) & \forall \psi \in \boldsymbol{\Phi}, \\ \phi(0) = \phi_0 & \text{in } \Omega, \end{cases} \tag{20}$$
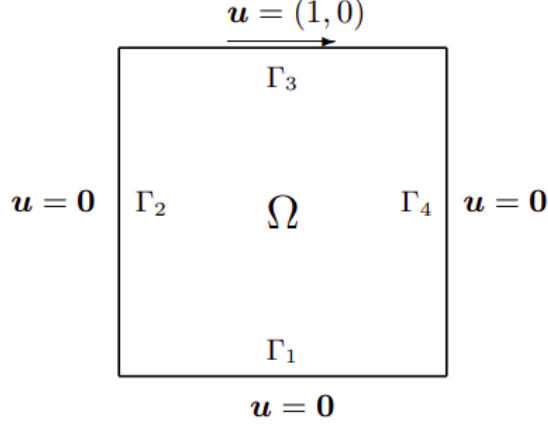
Figure 3: Lid-driven cavity problem: data and settings. $\Omega = (0,1)^2$, $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 = \partial\Omega$.

where $\overline{m}\left(\dfrac{\partial\phi}{\partial t}, \psi\right) = \displaystyle\int_\Omega \boldsymbol{curl}\dfrac{\partial\phi}{\partial t} \cdot \boldsymbol{curl}\psi\, d\Omega$, $\overline{a}(\phi,\psi) = \dfrac{2}{\mathbb{R}e}\displaystyle\int_\Omega D(\boldsymbol{curl}\phi) : D(\boldsymbol{curl}\psi)\, d\Omega$, $c(\phi,\phi,\psi) = \displaystyle\int_\Omega ((\boldsymbol{curl}\phi \cdot \nabla)\boldsymbol{curl}\phi) \cdot \boldsymbol{curl}\psi\, d\Omega$, while $\overline{F}(\psi) = \displaystyle\int_\Omega \boldsymbol{f} \cdot \boldsymbol{curl}\psi\, d\Omega$ and $\overline{H}(\psi) = \displaystyle\int_{\Gamma_N} \boldsymbol{h} \cdot \boldsymbol{curl}\psi\, d\Gamma$. The initial condition is chosen in such a way that $\boldsymbol{curl}\phi_0 = \boldsymbol{u}_0$. By construction the velocity field $\boldsymbol{u} = \boldsymbol{curl}\phi$ is divergence free.

We solve the lid-driven cavity benchmark problem on a square domain $\Omega = (0,1)^2$ as done in [40]. In Figure 3, we depict the boundary conditions assigned to $\partial\Omega$. We consider one single parameter, i.e. the Reynolds number $\mathbb{R}e \in [100, 5000]$, using the streamfunction formulation of the Navier-Stokes equations in 2D. We look for the steady state of the Navier-Stokes equations, i.e. we always take the snapshot at the final time $t = T = 30$ only. Indeed, we apply the KPOD-NN reduction on $\phi = \phi(\boldsymbol{x}, T; \mathbb{R}e)$.

4 Numerical discretization

In this section we discretize in space and time problems (16) and (20).

4.1 Wave equation

We use the finite difference method [32] to discretize the wave equation. We consider a partition $[x_0 = 0, x_1, ..., x_N = L]$ of $N+1$ equally spaced points distributed in the computational domain $\Omega = (0, L)$, and a set $[t_0 = 0, t_1, ..., t_{N_T} = T]$ of $N_T + 1$ equally spaced times in $[0, T]$. We employ second-order centered finite differences to approximate the derivatives of the PDE, leading to the following formulation:

$$
\begin{cases}
\dfrac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\Delta t^2} = c^2 \dfrac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{\Delta x^2} & \forall i = 1, ..., N-1 \quad \forall j = 0, ..., N_T - 1, \\
u_i^0 = A_0 \exp^{-\frac{(x_i - x_0)^2}{2\sigma^2}} & \forall i = 1, ..., N-1, \\
u_0^j = 0 & \forall j = 0, ..., N_T, \\
u_N^j = 0 & \forall j = 0, ..., N_T,
\end{cases}
\tag{21}
$$

where $\Delta t = T/N_T$ and $\Delta x = L/N$. We rearrange the first equation in (21) to obtain u_i^{j+1}:

$$
u_i^{j+1} = C(u_{i+1}^j + u_{i-1}^j) + 2(1 - C)u_i^j - u_i^{j-1},
\tag{22}
$$

with $C = c^2 \dfrac{\Delta t^2}{\Delta x^2}$. We remind that the value of $c\dfrac{\Delta t}{\Delta x}$ must be less or equal than 1 to satisfy the Courant-Friedrichs-Lewy condition, which ensures the stability of the numerical scheme [32].

4.2 Navier-Stokes equations

Following the approach proposed in [40], we provide the space discretization of (20) by means of the NURBS-based Isogeometric Analysis (IGA) Galerkin method. We employ the generalized-α scheme for its time approximation [6, 20].

Given an exact representation of the square domain Ω through B-splines functions, we introduce the finite dimensional space of B-splines in the physical domain [40], say $\mathcal{V}_h$, and we define $\boldsymbol{\Phi}_h = \boldsymbol{\Phi} \cap \mathcal{V}_h$ and $\boldsymbol{\Phi}_{g,h} = \boldsymbol{\Phi}_g \cap \mathcal{V}_h$, being $N_s = dim(\boldsymbol{\Phi})$ the dimension and $\{\mathcal{R}_i\}_{i=1}^{N_s}$ the set of basis functions. The IGA approximation of (20) reads: find $\phi_h(t) \in \boldsymbol{\Phi}_{g,h} \ \forall t \in (0, T)$ such that:

$$\begin{cases} \overline{m}\left(\dfrac{\partial \phi_h}{\partial t}, \psi_h\right) + \overline{a}(\phi_h, \psi_h) + \overline{c}(\phi_h, \phi_h, \psi_h) = \overline{F}(\psi_h) + \overline{H}(\psi_h) & \forall \psi_h \in \boldsymbol{\Phi}_h, \\ \phi_h(0) = \phi_{h,0} & \text{in } \Omega. \end{cases} \tag{23}$$

We can rewrite (24) in residual form: find $\phi_h(t) \in \boldsymbol{\Phi}_{g,h} \ \forall t \in (0, T)$ such that:

$$\begin{cases} R_h\left(\psi_h, \dfrac{\partial \phi_h}{\partial t}, \phi_h\right) = 0 & \forall \psi_h \in \boldsymbol{\Phi}_h, \\ \phi_h(0) = \phi_{h,0} & \text{in } \Omega, \end{cases} \tag{24}$$

with $R_h\left(\psi_h, \dfrac{\partial \phi_h}{\partial t}, \phi_h\right) = \overline{m}\left(\dfrac{\partial \phi_h}{\partial t}, \psi_h\right) + \overline{a}(\phi_h, \psi_h) + \overline{c}(\phi_h, \phi_h, \psi_h) - \overline{F}(\psi_h) - \overline{H}(\psi_h)$. We define the vector of discrete residuals whose components are the residuals $R_h\left(\cdot, \dfrac{\partial \phi_h}{\partial t}, \phi_h\right)$ evaluated in the NURBS basis functions $\mathcal{R}_i \ \ i = 1, ..., N_s$ for the function space $\boldsymbol{\Phi}_h$, i.e. $\boldsymbol{R}\left(\dfrac{\partial \phi_h}{\partial t}, \phi_h\right) = \left\{R_h\left(\mathcal{R}_i, \dfrac{\partial \phi_h}{\partial t}, \phi_h\right)\right\}_{i=1}^{N_s}, \forall t \in (0, T)$. Moreover, we introduce $\boldsymbol{\phi} = \boldsymbol{\phi}(t) = \{\phi_i\}_{i=1}^{N_s}$ and $\dfrac{\partial \boldsymbol{\phi}}{\partial t} = \dot{\boldsymbol{\phi}} = \dfrac{\partial \boldsymbol{\phi}}{\partial t}(t) = \left\{\dfrac{\partial \phi_i}{\partial t}\right\}_{i=1}^{N_s}$, which are the vectors of control variables $\forall t \in (0, T)$ for the function ϕ_h and its time derivative $\dfrac{\partial \phi_h}{\partial t}$ respectively. Consider $[t_0 = 0, t_1, ..., t_{N_T} = T]$ of $N_T + 1$ equally spaced times in $[0, T]$. In this framework, we perform the time discretization by means of the generalized-α method [6, 20, 40]: at time t_j, given $\dot{\boldsymbol{\phi}}_j$ and $\boldsymbol{\phi}_j$, find $\dot{\boldsymbol{\phi}}_{j+1}, \boldsymbol{\phi}_{j+1}, \dot{\boldsymbol{\phi}}_{j+\alpha_m}, \boldsymbol{\phi}_{j+\alpha_f}$ such that:

$$\begin{cases} \boldsymbol{R}\left(\dot{\boldsymbol{\phi}}_{j+\alpha_m}, \boldsymbol{\phi}_{j+\alpha_f}\right) = \boldsymbol{0}, \\ \dot{\boldsymbol{\phi}}_{j+\alpha_m} = \dot{\boldsymbol{\phi}}_j + \alpha_m\left(\dot{\boldsymbol{\phi}}_{j+1} - \dot{\boldsymbol{\phi}}_j\right), \\ \boldsymbol{\phi}_{j+\alpha_f} = \boldsymbol{\phi}_j + \alpha_f(\boldsymbol{\phi}_{j+1} - \boldsymbol{\phi}_j), \\ \boldsymbol{\phi}_{j+1} = \boldsymbol{\phi}_j + \Delta t\dot{\boldsymbol{\phi}}_j + \delta \Delta t\left(\dot{\boldsymbol{\phi}}_{j+1} - \dot{\boldsymbol{\phi}}_j\right), \end{cases} \tag{25}$$

where $\delta, \alpha_m, \alpha_f \in \mathbb{R}_0^+$ are chosen on the basis of accuracy and stability considerations and Δt is the fixed time step.

For further details about the numerical scheme and IGA we refer to [40].

5 Numerical results

We present some numerical results of wave equation and Navier-Stokes equations. For the first test case we compute all the variables at the mesh nodes, i.e. at the vertices, considering a mesh with 256 elements. We use B–Splines basis functions of degree $p = 2$ on uniform mesh of size $h = 1/64$ (4096 elements) and $\Delta t = 0.1$ for the lid-driven cavity problem. Indeed, IGA generally permits to obtain accurate numerical solutions using a reduced number of mesh elements than the Finite Element Method (FEM) [40]. This leads to smaller datasets that ease both the NN training and testing phases regardless of the specific ROM technique.

We implemented a MATLAB code for the numerical simulations of the wave equation, whereas the lid-driven cavity problem is available as a test case in the C++ IGA library isoglib. We train the neural network using the Google Tesla K80 NVIDIA GPU.

We compute the KPOD-NN relative approximation error in the following way:

$$\varepsilon_{KPOD-NN}(n, t, \boldsymbol{\mu}) = \frac{||\boldsymbol{u}_h(t; \boldsymbol{\mu}) - \mathbb{V}\boldsymbol{u}_h^{NN}(t; \boldsymbol{\mu})||}{||\boldsymbol{u}_h(t; \boldsymbol{\mu})||}. \tag{26}$$
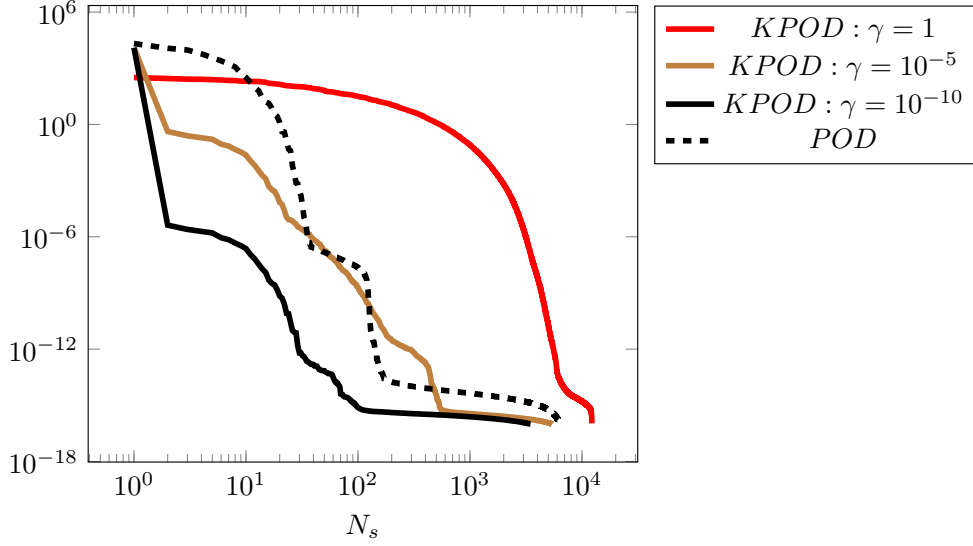
Figure 4: Comparison between the eigenvalues of the kernel matrix, coming from KPOD and different values for γ (continuous), and the eigenvalues of the Gram matrix $\mathbb{C}$, coming from POD (dashed), for wave equation.

This error is evaluated on a test dataset $\mathcal{D}_{test}$ made of N_{test} elements. The final estimator that we use is the average of all KPOD-NN relative approximation errors computed on N_{test} data:

$$\bar{\varepsilon}_{KPOD-NN}(n) = \frac{\sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}_{test}} \varepsilon_{KPOD-NN}(n,t,\boldsymbol{\mu})}{N_{test}} \tag{27}$$

5.1 Wave equation

In this section, we focus on the numerical results related to the wave equation. We denote the structure of both the training and test sets in Tables 2 and 3. Values of A_0, x_0 and σ in the building dataset are obtained by means of latin hypercube sampling [31]. We observe that, given a certain tolerance $\hat{\varepsilon}$ and a suitable value for γ, the reduced dimension that we get from KPOD ($n_{KPOD-NN} = 15$) is fairly lower than the one obtained from POD ($n_{POD-NN} = 109$). Moreover, the KPOD-NN size will be smaller than the POD-NN one and the former will be less expensive to train and to test in terms of both computational resources and total computational time.

Type of dataset	N_h	N_t	N_{A_0}	N_{x_0}	N_σ	$n_{KPOD-NN}$	n_{POD-NN}
Training set	256	100	5	5	5	15	109
Test set	256	100	1	1	1	-	-

Table 2: Building and test datasets for wave equation. These datasets have both the same mesh nodes and time steps, whereas values of A_0, x_0 and σ are different. We consider $\gamma = 10^{-10}$ to determine $n_{KPOD-NN}$.

Type of dataset	Parameter	Values
Training set	A_0	[0.5 0.641 0.721 0.821 1.0]
	x_0	[4.189 5.169 6.065 7.426 8.378]
	σ	[0.5 0.637 0.745 0.898 1.0]
Test set	A_0	0.75
	x_0	8.0
	σ	0.9

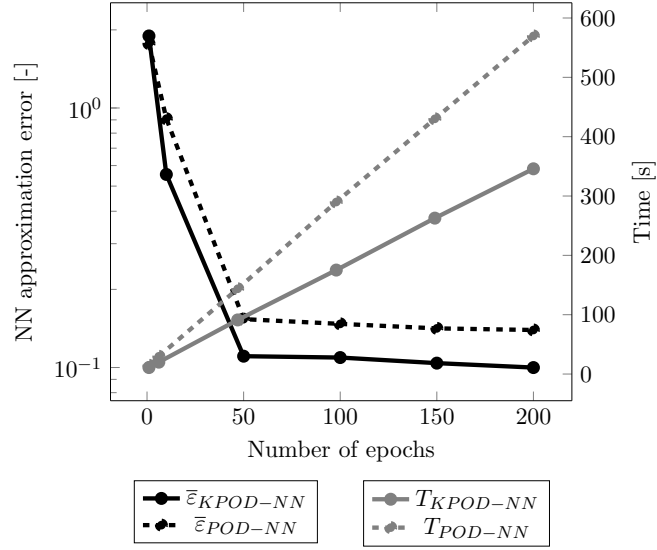Table 3: Composition of building and test sets in terms of parameter values.

Figure 5: NN approximation errors and NN training computational times related to KPOD-NN and POD-NN vs. number of epochs for wave equation ($n_{KPOD-NN} = 15$, $n_{POD-NN} = 109$, $\gamma = 10^{-10}$).
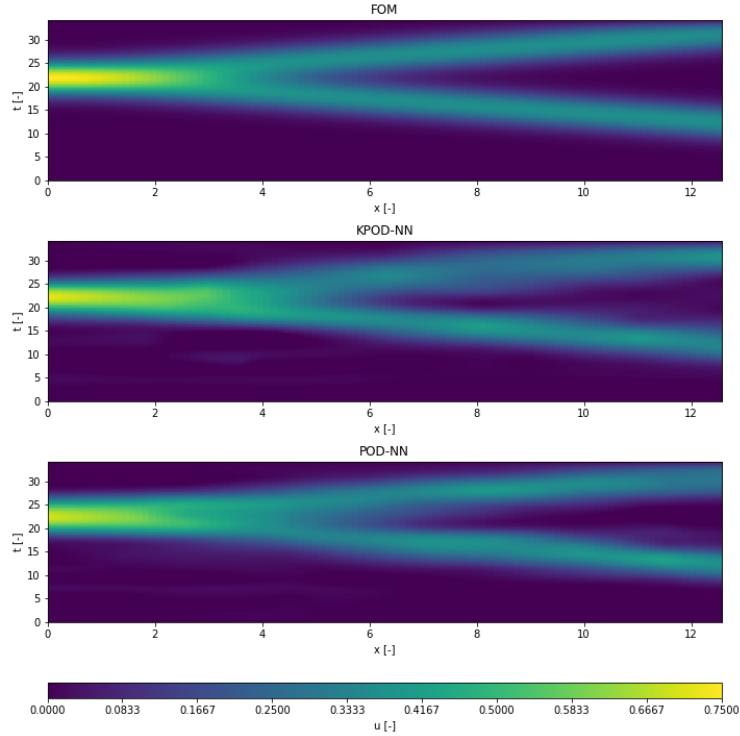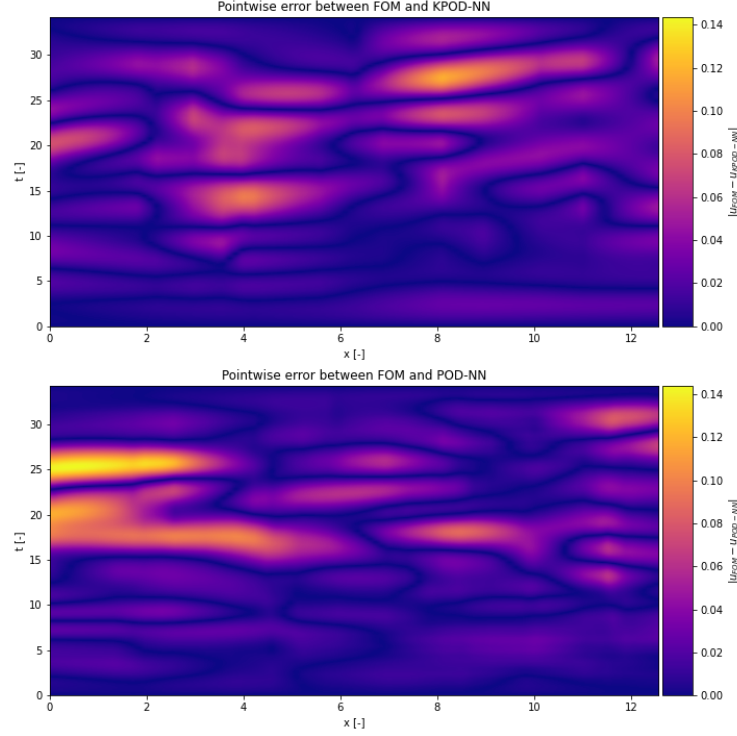

Figure 6: Numerical results on the test set $\mathcal{D}_{test}$ for wave equation. Comparison among FOM, KPOD-NN and POD-NN solutions. Both NN have been trained for $N_{epochs} = 100$, with $n_{KPOD-NN} = 15$ ($\gamma = 10^{-10}$) and $n_{POD-NN} = 109$.

In Figure 4 we highlight the role of γ. In particular, at least for this test case and for the range of values that we consider, the lower the value of parameter γ is, the higher the rate of decay is. We show the capability of the KPOD method to potentially collect the most significant modes and information in the really first eigenvalues of the kernel

Figure 7: Pointwise absolute error on the test set $\mathcal{D}_{test}$ for wave equation. Comparison among FOM, KPOD-NN and POD-NN solutions. Both NN have been trained for $N_{epochs} = 100$, with $n_{KPOD-NN} = 15$ ($\gamma = 10^{-10}$) and $n_{POD-NN} = 109$.
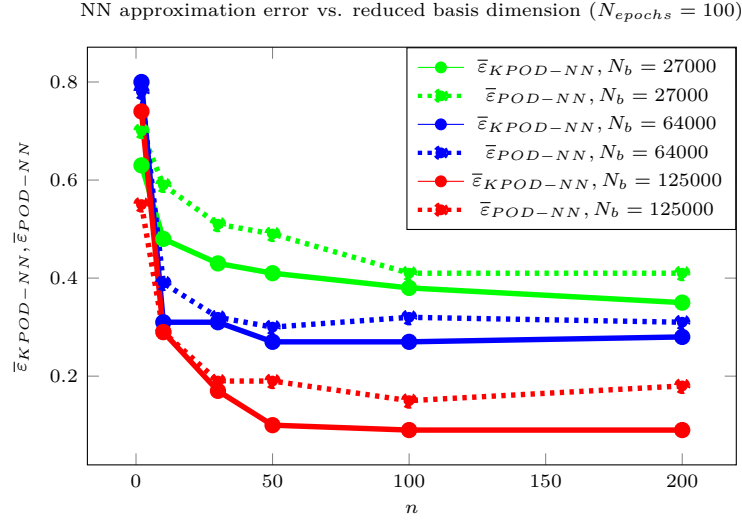
Figure 8: $\overline{\varepsilon}_{KPOD-NN}(n)$ and $\overline{\varepsilon}_{POD-NN}(n)$ errors vs. n for different choices of the dataset, which contain the numerical solutions of the wave equation for different parameters instances. $N_b = 125000$ corresponds to the building set introduced in Tables 2 and 3.

matrix. Moreover, the computational time that is needed to perform KPOD is comparable with the one of POD, due to the fact that we do not have to deal with high-dimensional data explicitly [38].

We depict in Figure 5 the behavior of both $\overline{\varepsilon}_{KPOD-NN}(n)$ and $\overline{\varepsilon}_{POD-NN}(n)$ errors with respect to N_{epochs}. We see that, providing a proper number of epochs, once convergence is reached, the approximation error related to KPOD-NN
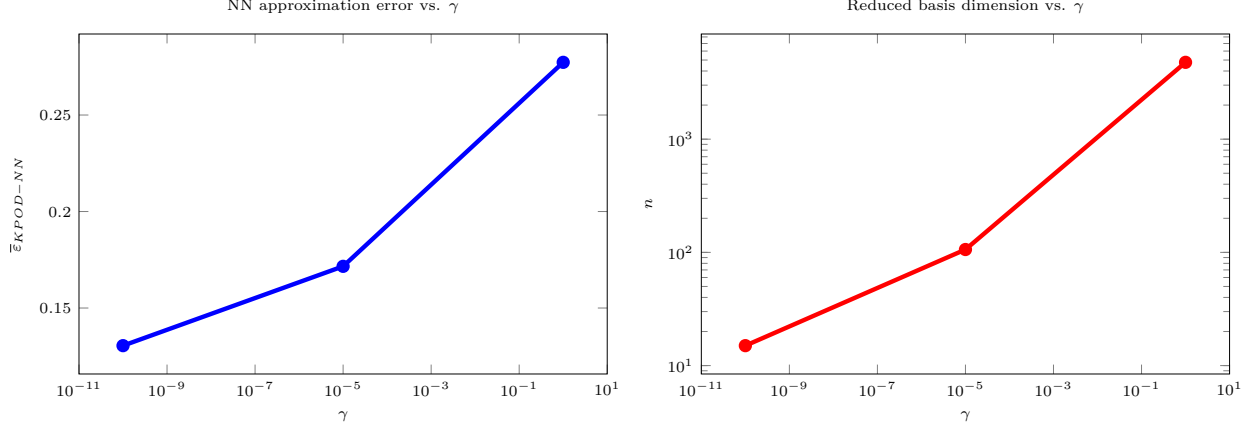
Figure 9: NN approximation error $\overline{\varepsilon}_{KPOD-NN}$ (left) and reduced basis dimension n (right) vs. γ for wave equation. All NN have been trained for $N_{epochs} = 100$.
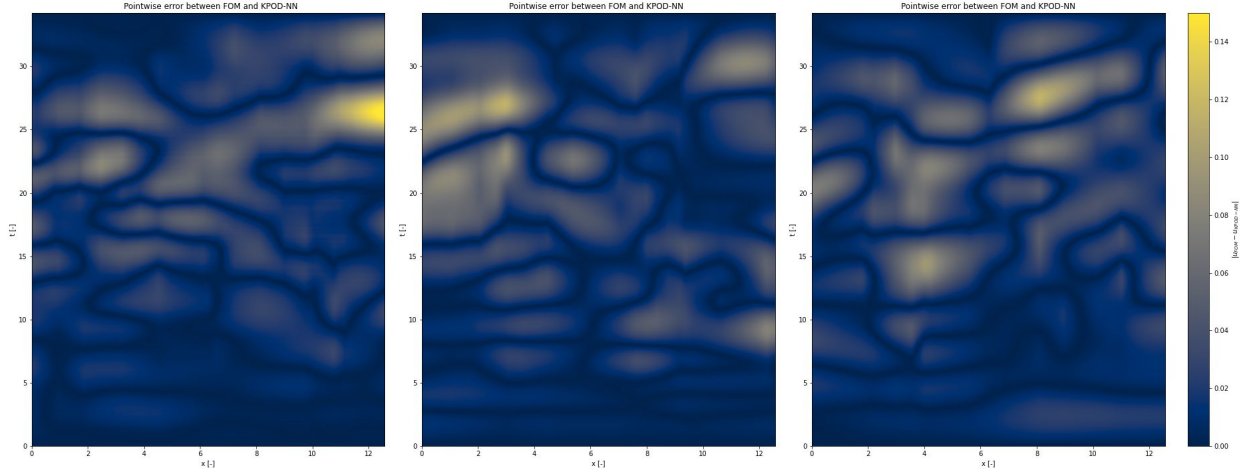

Figure 10: Pointwise absolute error on the test set $\mathcal{D}_{test}$ for wave equation. Comparison among KPOD-NN solutions for different values of γ. All NN have been trained for $N_{epochs} = 100$. We use $\gamma = 1$ (left, $n_{KPOD-NN} = 4772$), $\gamma = 10^{-5}$ (center, $n_{KPOD-NN} = 106$) and $\gamma = 10^{-10}$ (right, $n_{KPOD-NN} = 15$).

is slightly smaller than the one of POD-NN method, even if the size of the two NN is different. We also notice that the total time to train the KPOD-NN is approximately halved with respect the POD-NN one. By looking at the numerical results in Figure 6 on the test set D_{test} for all spatial and temporal coordinates, we see a good agreement between the FOM solution and the KPOD-NN one. This is also confirmed in Figure 7, where we show that also the pointwise difference in absolute value between the FOM solution and the ROM one is generally lower for the KPOD-NN method. In Figure 8 we see the plot related to NN approximation error vs. reduced dimension n for different building sets. We comment that, given a specific NN architecture and size, KPOD-NN has a higher representational power than POD-NN, and the smaller the training dataset is, the higher the plateau of the approximation error is. We stress that in this specific test case we are solving a linear PDE in 1D. For this reason, the advantages of KPOD over POD on the NN approximation errors are quite limited.

In Figure 9 we show both the behavior of the NN approximation error $\overline{\varepsilon}_{KPOD-NN}$ and reduced basis dimension n with respect to parameter γ. Given a fixed tolerance $\hat{\varepsilon}$, we notice that both $\overline{\varepsilon}_{KPOD-NN}$ and n scales monotonically with γ. We also depict in Figure 10 the pointwise absolute error related to KPOD-NN solutions for different choices of γ. We highlight that high values, such as $\gamma = 1$, are not favorable and might be source of high localized errors in space and time. Indeed in this scenario, given again a fixed tolerance $\hat{\varepsilon}$, the reduced dimension n could be potentially high and could lead to a bigger NN, which is more challenging to train. On the other hand, when lower values are considered (e.g. $\gamma = 10^{-5}$ or $\gamma = 10^{-10}$), KPOD modes decay faster and steeply while the corresponding NN is smaller and easier to train.
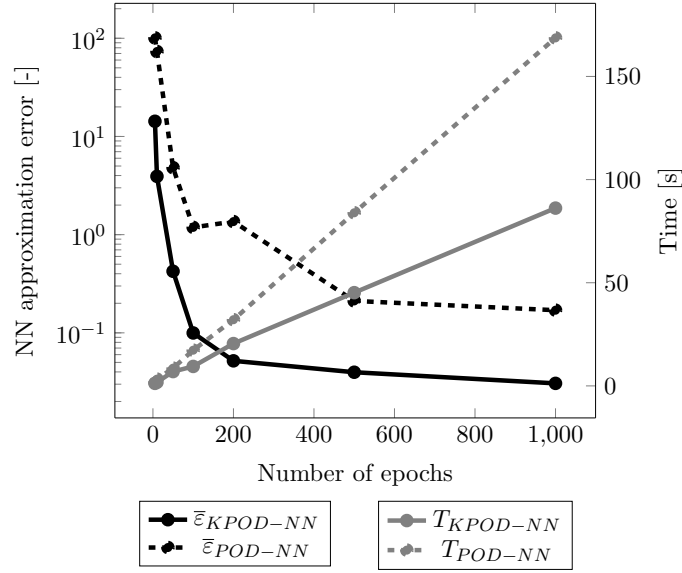
14

Figure 11: NN approximation errors and NN training computational times related to KPOD-NN and POD-NN vs. number of epochs for the lid driven cavity benchmark ($n_{KPOD-NN} = 12$, $n_{POD-NN} = 131$).

5.2 Lid-driven cavity benchmark

Here we present the numerical results for the lid-driven cavity problem. We generate the training set by means of FOM solutions, considering evenly spaced Reynolds numbers between 100 and 5000 with a fixed step equal to 10. The test set is sampled for $\mathbb{R}e \in [105, 4995]$ by using again a fixed step equal to 10. In Table 4 and Figure 11 we report some information related to the training phase of the NN when either KPOD or POD is employed on the same building set. Given a certain number of epochs and a tolerance $\hat{\varepsilon}$, the KPOD-NN approach leads to an approximation error that is one order of magnitude less than the POD-NN one. Moreover, the computational time is again halved for KPOD-NN and its reduced basis dimension is reduced by a factor 10. We highlight that the dimension of the reduced basis affects directly the number of NN parameters to train, i.e. the weights of the NN. This number is significantly smaller for KPOD-NN.

Technique	NN parameters	n	Number of epochs	NN approximation error	Computational time $[s]$
KPOD-NN	360	12	1000	$3.1 \cdot 10^{-2}$	86.17
POD-NN	69'954	131	1000	$1.7 \cdot 10^{-1}$	168.7

Table 4: Comparison of the KPOD-NN and POD-NN methods for the lid-driven cavity problem on the same building set, in terms of both $\overline{\varepsilon}_{KPOD-NN}(n)/\overline{\varepsilon}_{POD-NN}(n)$ (i.e. NN approximation error) and computational times to train the NN. We use $\gamma = 10^{-5}$ for KPOD.

Moreover, also in this case, we get a reduced value of n thanks to the better behavior of the eigenvalues of the kernel matrix with respect to the singular values of matrix $\mathbb{S}$, as can be seen in Figure 12. The computational times to train the NN coming from the KPOD approximation is strongly reduced with respect to the one related to POD. Testing times remain comparable and negligible. The same occurs for the computational times to perform KPOD and POD effectively.

In Figures 13 and 14 we compare the streamlines and the values assumed by the streamfunction ϕ for FOM and KPOD-NN for different Reynolds numbers $\mathbb{R}e$. We see that, considering the number and positions of the contour lines, the KPOD-NN method captures all vortexes properly. Indeed strong nonlinearities arises in the 2D lid-driven cavity problem, where there are both a discontinuous dependance of the solution with the Reynolds number and bifurcations in the development of vortices. For these reasons, in this second test case, we have stronger advantages in using a fully nonlinear technique, such as KPOD-NN.
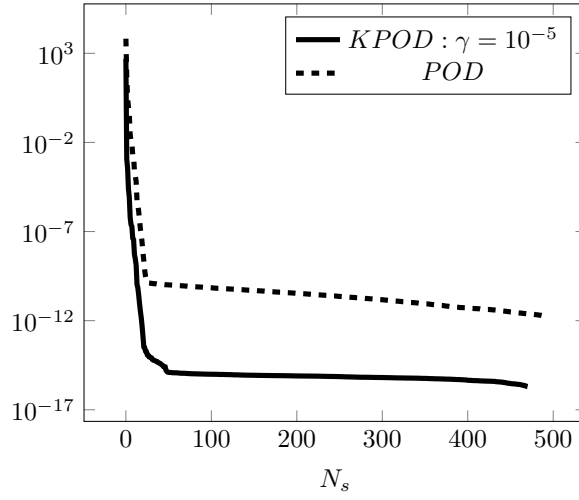
Figure 12: Comparison between the eigenvalues of the kernel matrix, coming from KPOD (continuous), and the eigenvalues of the Gram matrix $\mathbb{C}$, coming from POD (dashed), for the lid-driven cavity problem.
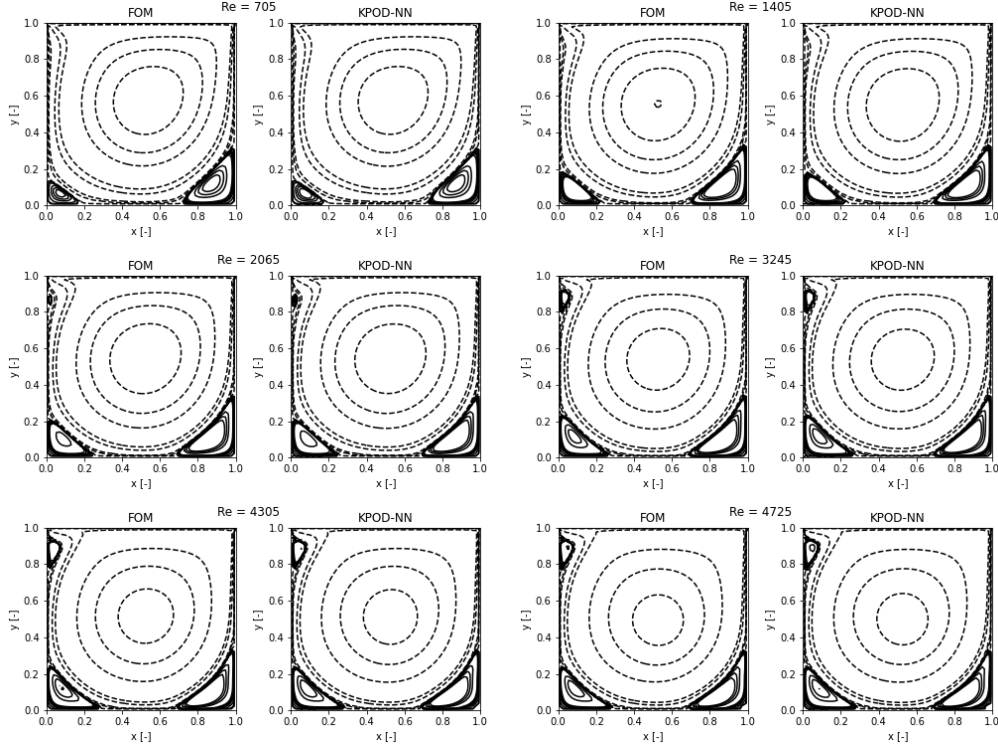

Figure 13: Streamlines computed on the test set $\mathcal{D}_{test}$ for the lid-driven cavity benchmark. Comparison between FOM and KPOD-NN solutions. The NN has been trained for $N_{epochs} = 1000$, with $n_{KPOD-NN} = 12$ ($\bar{\varepsilon}_{KPOD-NN} = 3.1 \cdot 10^{-2}$).
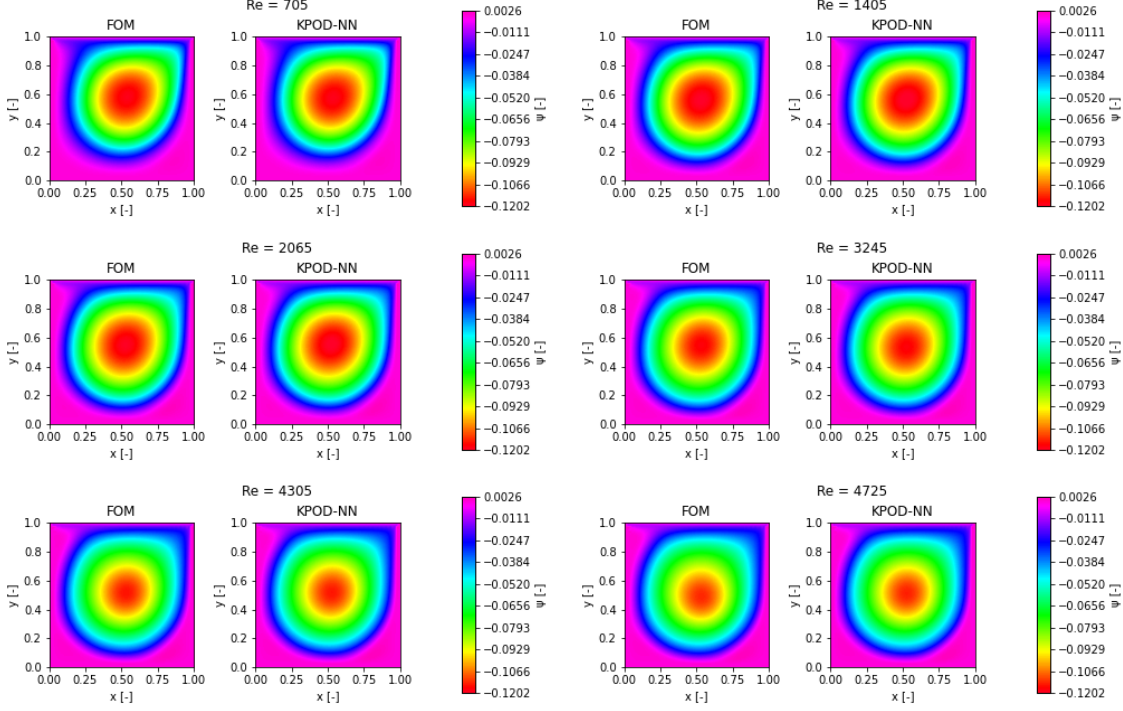
Figure 14: Streamfunction ϕ computed on the test set $\mathcal{D}_{test}$ for the lid-driven cavity benchmark. Comparison between FOM and KPOD-NN solutions. The NN has been trained for $N_{epochs} = 1000$, with $n_{KPOD-NN} = 12$ ($\bar{\varepsilon}_{KPOD-NN} = 3.1 \cdot 10^{-2}$).

6 Conclusions

In this work, we proposed a non-intrusive reduced order model technique that combines KPOD with an adaptively built NN, whose number of layers and number of neurons scale according to the dimension of the extracted reduced basis. The use of nonlinear dimensionality reduction in the first part of the algorithm permits to shrink the valuable information on the first modes. This operation strongly reduces the number of functions that must be collected to obtain a basis up to specific tolerance, in particular if this method is compared to linear dimensionality reduction techniques, such as POD.

We applied our methodology to parametrized parabolic and hyperbolic PDEs in both 1D and 2D settings, either in a linear (wave equation) or nonlinear (Navier-Stokes equations) context. We saw a good agreement between the FOM solution and the KPOD-NN solution in both cases. We also compared our method with the POD-NN one, by showing that the reduced coefficients coming from KPOD contain potentially more information than the one extracted by means of POD. This leads to a smaller NN approximation error on the test sets. Moreover, we remind that the NN size coming from KPOD is again by construction necessarily smaller and easier to train than the one given by POD. Both KPOD and POD involve the same computational resources, and the computational times spent by the two methods to compute the reduced basis is still similar. According to our tests, by fixing a priori a certain tolerance, KPOD-NN collects a number of modes which is at least 10 times smaller than POD-NN while leading to smaller NN approximation errors, which are reduced by approximately a factor of 1.2 for the wave equation and by approximately a factor of 10 for the lid-driven cavity problem, where nonlinearities arise. Moreover, the training costs for KPOD-NN are at least halved with respect to POD-NN ones. By using even lower values of γ, all the advantages of KPOD-NN over POD-NN are potentially even stronger.

A possible extension of this work could address 3D problems with a significantly higher number of DOFs. Another possible topic for this non-intrusive technique could be the application to multifield fully-coupled problems, such as the one arising in cardiac modeling: in this framework, the nonlinear PDEs related to electrophysiology, mechanics and fluid dynamics can be reduced in an independent manner with potentially different NNs. Finally, we notice that our KPOD-NN can also be embedded in pre-existing deep learning ROM frameworks to improve their efficiency and accuracy.

Acknowledgements

References

[1] A.R. Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39 (1993), pp. 930–945.

[2] J. Berner, P. Grohs, and A. Jentzen. "Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black-Scholes Partial Differential Equations". In: *SIAM Journal on Mathematics of Data Science* 2 (2020), pp. 631–657.

[3] K. Bhattacharya et al. "Model reduction and neural networks for parametric PDEs". In: *arXiv preprint arXiv:2005.03180* (2020).

[4] H. Bölcskei et al. "Optimal approximation with sparsely connected deep neural networks". In: *SIAM Journal on Mathematics of Data Science* 1 (2020), pp. 8–45.

[5] F. Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

[6] J. Chung and G.M. Hulbert. "A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method". In: *Journal of Applied Mechanics* 60 (1993), pp. 371–375.

[7] T. M. Cover. "Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition". In: *IEEE Transactions on Electronic Computers* EC-14 (1965), pp. 326–334.

[8] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2 (1989), pp. 303–314.

[9] S. Fresca and A. Manzoni. "POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition". In: *arXiv preprint arXiv:2101.11845* (Jan. 2021).

[10] S. Fresca, A. Manzoni, and L. Dede'. "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs". In: *Journal of Scientific Computing* (2021). Accepted for publication.

[11] S. Fresca et al. "Deep learning-based reduced order models in cardiac electrophysiology". In: *PLOS ONE* 15.10 (2020), pp. 1–32.

[12] A. García-González et al. "A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy". In: *arXiv preprint arXiv:2001.01958* (2020).

[13] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. Proceedings of Machine Learning Research. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[14] F. J. González and M. Balajewicz. "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems". In: *arXiv preprint arXiv:1808.01346* (2018).

[15] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[16] M. Guo and J. S. Hesthaven. "Data-driven reduced order modeling for time-dependent problems". In: *Computer Methods in Applied Mechanics and Engineering* 345 (2019), pp. 75–99.

[17] M. Guo and J. S. Hesthaven. "Reduced order modeling for nonlinear structural analysis using Gaussian process regression". In: *Computer Methods in Applied Mechanics and Engineering* 341 (2018), pp. 807–826.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003.

[19] J. S. Hesthaven and S. Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363 (2018), pp. 55–78.

[20] K.E. Jensen, Whiting C.H., and Hulbert G.M. "A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method". In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), pp. 305–319.

[21] N. S. Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[22] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[23] A.N. Kolmogorov. "On the representation of continuous functions of several veriables as superpositions of continuous functions of one variable and addition". In: *The Proceedings of the USSR Academy of Sciences* 114 (1957), pp. 953–956.

[24] G. Kutyniok et al. "A Theoretical Analysis of Deep Neural Networks and Parametric PDEs". In: *arXiv preprint arXiv:1904.00377* (Apr. 2019).

[25] J. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[26] K. Lee and K.T. Carlberg. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders". In: *J. Comput. Phys.* 404 (2020), p. 108973.

[27] R. Maulik et al. "Time-series learning of latent-space dynamics for reduced-order model closure". In: *Physica D: Nonlinear Phenomena* (2020), p. 132368.

[28] N.T. Mücke, S.M. Bohté, and C.W. Oosterlee. "Reduced Order Modeling for Parameterized Time-Dependent PDEs using Spatially and Memory Aware Deep Learning". In: *arXiv preprint arXiv:2011.11327* (2020).

[29] S. Pagani, A. Manzoni, and A. Quarteroni. "Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method". In: *Computer Methods in Applied Mechanics and Engineering* 340 (2018), pp. 530–558.

[30] D. Pedamonti. *Comparison of non-linear activation functions for deep neural networks on MNIST classification task*. 2018. arXiv: `1804.02763` [`cs.LG`].

[31] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations*. Springer, 2016.

[32] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2010.

[33] M. Raissi and G. Karniadakis. "Deep Multi-fidelity Gaussian Processes". In: *arXiv preprint arXiv:1604.07484* (2016).

[34] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *arXiv preprint arXiv:1711.10561* (2017).

[35] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations". In: *arXiv preprint arXiv:1711.10566* (2017).

[36] F. Regazzoni, L. Dede', and A. Quarteroni. "Machine learning for fast and reliable solution of time-dependent differential equations". In: *Journal of Computational Physics* 397 (2019), p. 108852.

[37] O. San and R. Maulik. "Neural network closures for nonlinear model order reduction". In: *Advances in Computational Mathematics* 44 (2018), pp. 1717–1750.

[38] B. Scholköpf, A. Smola, and K. Müller. "Kernel principal component analysis". In: *Advances in kernel methods - support vector learning*. MIT Press, 1999, pp. 327–352.

[39] Z. Shen, H. Yang, and S. Zhang. "Deep Network Approximation Characterized by Number of Neurons". In: *arXiv preprint arXiv:1906.05497* (2019).

[40] A. Tagliabue, L. Dede', and A. Quarteroni. "Isogeometric Analysis and error estimates for high order partial differential equations in fluid dynamics". In: *Computers & Fluids* 102 (2014), pp. 277–303.

[41] Q. Wang, J. S. Hesthaven, and D. Ray. "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem". In: *Journal of Computational Physics* 384 (2019), pp. 289–307.

[42] W.W. Xing, A.A. Shah, and P.B. Nair. "Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on Isomap". In: *Proceedings of the Royal Society A* 471 (2015), p. 20140697.

[43] W.W. Xing et al. "Manifold learning for the emulation of spatial fields from computational models". In: *Journal of Computational Physics* 326 (2016), pp. 666–690.

[44] T. Zhang. "Sequential greedy approximation for certain convex optimization problems". In: *IEEE Transactions on Information Theory* 49 (2003), pp. 682–691.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

13/2021 Ferro, N.; Perotto, S.; Cangiani, A.
An anisotropic recovery-based error estimator for adaptive discontinuous Galerkin methods

14/2021 Peli, R.; Menafoglio, A.; Cervino, M.; Dovera, L.; Secchi, P;
Physics-based Residual Kriging for dynamically evolving functional random fields

15/2021 Fumagalli, A.; Patacchini, F.S.
Model adaptation in a discrete fracture network: existence of solutions and numerical strategies

10/2021 Di Michele, F.; May, J.; Pera, D.; Kastelic, V.; Carafa, M.; Smerzini, C.; Mazzieri, I.; Rubino, I.
Spectral elements numerical simulation of the 2009 L'Aquila earthquake on a detailed reconstructed domain

11/2021 Antonietti,P.F.; Manzini, G.; Mazzieri, I.; Scacchi, S.; Verani, M.
The conforming virtual element method for polyharmonic and elastodynamics problems: a review

12/2021 di Cristofaro, D.; Galimberti, C.; Bianchi, D.; Ferrante, R.; Ferro, N.; Mannisi, M.; Perotto, S
Adaptive topology optimization for innovative 3D printed metamaterials

09/2021 Riccobelli, D.; Noselli, G.; DeSimone, A.
Rods coiling about a rigid constraint: Helices and perversions

08/2021 Antonietti, P. F.; Manuzzi, E.
Refinement of polygonal grids using Convolutional Neural Networks with applications to polygonal Discontinous Galerkin and Virtual Element methods

07/2021 Diquigiovanni, J.; Fontana, M.; Vantini, S.
The Importance of Being a Band: Finite-Sample Exact Distribution-Free Prediction Sets for Functional Data

06/2021 Negrini, G.; Parolini, N.; Verani, M.
A diffuse interface box method for elliptic problems