



# Machine learning for fast and reliable solution of time-dependent differential equations

F. Regazzoni<sup>a,\*</sup>, L. Dedè<sup>a</sup>, A. Quarteroni<sup>a,b</sup>

<sup>a</sup> MOX - Dipartimento di Matematica, Politecnico di Milano, P.zza Leonardo da Vinci 32, 20133 Milano, Italy

<sup>b</sup> Mathematics Institute, École Polytechnique Fédérale de Lausanne, Av. Piccard, CH-1015 Lausanne, Switzerland (Professor Emeritus)



## ARTICLE INFO

### Article history:

Received 17 January 2019

Received in revised form 21 July 2019

Accepted 22 July 2019

Available online 26 July 2019

### Keywords:

Machine learning

Differential equations

Model order reduction

System identification

Artificial neural networks

Data-driven modeling

## ABSTRACT

We propose a data-driven Model Order Reduction (MOR) technique, based on Artificial Neural Networks (ANNs), applicable to dynamical systems arising from Ordinary Differential Equations (ODEs) or time-dependent Partial Differential Equations (PDEs). Unlike model-based approaches, the proposed approach is non-intrusive since it just requires a collection of input-output pairs generated through the high-fidelity (HF) ODE or PDE model. We formulate our model reduction problem as a maximum-likelihood problem, in which we look for the model that minimizes, in a class of candidate models, the error on the available input-output pairs. Specifically, we represent candidate models by means of ANNs, which we train to learn the dynamics of the HF model from the training input-output data. We prove that ANN models are able to approximate every time-dependent model described by ODEs with any desired level of accuracy. We test the proposed technique on different problems, including the model reduction of two large-scale models. Two of the HF systems of ODEs here considered stem from the spatial discretization of a parabolic and an hyperbolic PDE respectively, which sheds light on a promising field of application of the proposed technique.

© 2019 Elsevier Inc. All rights reserved.

## 1. Introduction

The numerical simulation of time-dependent mathematical models is often needed in applied sciences. The increasing demand of more complex and reliable mathematical models may lead to an unbearable demand for computational resources, either in terms of computational power or memory storage. Very often, in several applications, numerical simulations of a given model need to be performed multiple times (multi-query) and for many different inputs, either for sensitivity-analysis, optimization, control or uncertainty-quantification purposes, or to deal with multiscale problems. In computational medicine and meteorology e.g., complex mathematical models, although precise and reliable, may be useless for predictive purposes if these cannot be solved nearly in real-time.

This strongly motivates the development of *reduced models*, that is computationally tractable, lower dimensional mathematical models which can be solved with a smaller effort (both in terms of time and computational resources), yet reproducing with a good approximation the results of the *high-fidelity* (HF) model ([1–4]).

\* Corresponding author.

E-mail address: francesco.regazzoni@polimi.it (F. Regazzoni).

In this paper, we focus on time-invariant dynamical systems, whose behavior is determined by a time-dependent input  $\mathbf{u}(t) \in \mathbb{R}^{N_u}$  and endowed with an output  $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ . Let us consider the following general<sup>1</sup> form for our HF model:

$$\begin{cases} \dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{X}(0) = \mathbf{X}_0 \end{cases} \quad (1)$$

$$\mathbf{y}(t) = \mathbf{G}(\mathbf{X}(t)), \quad t \in (0, T],$$

with  $\mathbf{F}: \mathbb{R}^N \times \mathbb{R}^{N_u} \rightarrow \mathbb{R}^N$ ,  $\mathbf{G}: \mathbb{R}^N \rightarrow \mathbb{R}^{N_y}$ , where  $\mathbf{X}(t)$  represents the HF state of the system and can be either finite-dimensional, for ODE models (i.e.  $\mathbf{X}(t) \in \mathbb{R}^N$ ), or infinite-dimensional, e.g. for PDE models. We notice that (1) is not the most general form that we can face: indeed the equation can be stated in implicit form. However we stick to form (1) just to illustrate the concept.

Most of Model Order Reduction (MOR) methods for time-dependent problems provide a reduced model in the following form

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (2)$$

$$\tilde{\mathbf{y}}(t) = \mathbf{g}(\mathbf{x}(t)), \quad t \in (0, T],$$

where the reduced-order state  $\mathbf{x}(t)$  belongs to a lower dimensional space  $\mathbb{R}^n$  (typically with  $n \ll N$ ) and the functions  $\mathbf{f}: \mathbb{R}^n \times \mathbb{R}^{N_u} \rightarrow \mathbb{R}^n$  and  $\mathbf{g}: \mathbb{R}^n \rightarrow \mathbb{R}^{N_y}$  can be evaluated with a smaller computational effort than  $\mathbf{F}$  and  $\mathbf{G}$  in Eq. (1). Notice that, in several cases, the knowledge of the evolution of the full-order state  $\mathbf{X}(t)$  is not essential since the user may be interested in just one or a few output quantities, represented by  $\mathbf{y}(t)$ . Should the reduced model (2) be able to reliably reproduce the input-output map  $\mathbf{u} \rightarrow \mathbf{y}$ , it could replace the HF model (1), hopefully with a considerable gain in terms of computational resources and simulation time. We here denote by  $\tilde{\mathbf{y}}(t)$  the output of the reduced model, which generally differs from that of the HF model  $\mathbf{y}(t)$ .

MOR can be *model-based* or *data-driven* ([5–9]). In model-based MOR, Eq. (1) is the starting point to derive its reduced version (2). With data-driven approaches instead, the reduced model is built upon a collection of input-output pairs, through which the dynamics of the system is inferred. The advantages of model-based approaches are that, often, the reduced model inherits structural properties (e.g. stability) from the HF model; moreover, the underlying HF system structure provides the base for deriving error estimates, hence error certification. Unfortunately, an equation in the form of (1) is not always available to express the dynamics of the HF model, which may be accessible only through input-output data. This is the case, for instance, when the system dynamics is available only through experimental measurements, either in the time domain or in the frequency domain (i.e. through samples of the transfer function, in the case of a linear system), or when the HF system is accessible through the simulations of a black-box software library. On the other hand, even when the HF model is available, the implementation of model-based MOR into existing codes may not be straightforward. Data-driven MOR, instead, thanks to its black-box approach, is intrinsically non-intrusive and can thus be applied even when the HF model is not directly accessible.

### 1.1. Model-based MOR

The most popular approach to model-based MOR for dynamical systems consists in *projection-based* methods ([5,2,1,6]). In this framework, the full-order state space  $\mathbb{R}^N$  is approximated by a lower-dimensional subspace  $\text{span}(\mathbf{V})$ , where  $\mathbf{V} \in \mathbb{R}^{N \times n}$  is the matrix whose columns are the basis of the subspace. The full-order state is approximated as  $\mathbf{X}(t) \simeq \mathbf{V}\mathbf{x}(t)$  and the HF model equation is projected in the Galerkin (or Petrov-Galerkin) sense, by left multiplying it by  $\mathbf{V}^T$  (or by another matrix  $\mathbf{W}^T$ , respectively):

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{W}^T \mathbf{F}(\mathbf{V}\mathbf{x}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{x}(0) = \mathbf{W}^T \mathbf{X}_0 \end{cases} \quad (3)$$

$$\tilde{\mathbf{y}}(t) = \mathbf{G}(\mathbf{V}\mathbf{x}(t)), \quad t \in (0, T].$$

This is equivalent to imposing orthogonality of the HF residual to  $\text{span}(\mathbf{V})$  (or  $\text{span}(\mathbf{W})$ ). Various projection-based methods differ on the selection procedure of the bases for  $\mathbf{V}$  and  $\mathbf{W}$ .

When the HF model (1) is linear in the state  $\mathbf{X}$ , the Moment-Matching approach (or Padé approximation) consists in building  $\mathbf{V}$  and  $\mathbf{W}$  in such a way that the associated transfer function interpolates, up to a desired order, the first few moments of the full-order transfer function for a given frequency ([10–12]). The Balanced-Truncation approach applies in

<sup>1</sup> The non time-invariant case (i.e.  $\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}(t), \mathbf{u}(t), t)$ ) can be written in the form (1) by introducing a further dependent variable  $X_{N+1}$ , representing time, with equation  $\dot{X}_{N+1}(t) = 1$  and  $X_{N+1}(0) = 0$ . Moreover, Eq. (1) also includes parametric differential equations, that is the case when  $\mathbf{u}(t)$  is constant in time and is regarded as a parameter.

the linear case too: it neglects the states corresponding to the smallest Hankel singular values, which measure the relevance of each state in terms of both reachability and observability ([13,6]). Proper Orthogonal Decomposition (POD), originally dating back to [14] and developed by Sirovich in 1987 (see [15]), is tightly related to Principal Component Analysis (PCA, see [16]) and to Karhunen-Loève expansion, also known as Hotelling transform in stochastic process theory ([17]). The POD approach consists in collecting a set of *snapshots* of the full-order state  $\mathbf{X}$  (i.e. solutions computed at different times for different input values) and in building a basis by selecting the left singular vectors of the snapshot matrix corresponding to the largest singular values, computed through Singular Value Decomposition (SVD) of this snapshot matrix. Such basis is optimal as it minimizes the least-squares error of snapshot reconstruction ([6]). POD gained a huge popularity in the PDE framework: the POD Galerkin projection-based approach represents one of the two classical forms of the Reduced Basis (RB) method, the other one being based on the greedy algorithm to generate the snapshots (see e.g. [18,19,4,20,21]).

The effectiveness of projection-based MOR relies on the offline/online decomposition. Such decoupling is straightforward in the case of linear models with affine dependence of  $\mathbf{F}$  on the input  $\mathbf{u}$  since the algebraic structures (matrices and vectors) of the reduced model can be precomputed *offline* (i.e. during the construction of the reduced model) by projection of the full-order algebraic structures ([4,2]); the latter do not need to be accessible in the *online* phase (i.e. during the numerical simulation of the reduced model). In the nonlinear case and/or with non-affine input dependence the offline and online phases cannot be decoupled in principle, and the full-order right-hand side  $\mathbf{F}$  should be evaluated at each time step of the online phase, thus preventing the reduction of the HF model complexity. To overcome this issue, the nonlinear dependence on the state and the input is replaced by affine approximations by employing techniques such as the empirical interpolation method, EIM ([22,23]), the discrete empirical interpolation method, DEIM ([24,25]), its matrix version MDEIM ([26]) or the gappy POD reconstruction ([27]). In [28] the authors proposed a projection-based MOR technique, where the system is projected into a nonlinear manifold, determined by convolutional autoencoders from deep learning.

Another class of model-based MOR techniques is that of hierarchical surrogates, that is to say models derived from the HF one under simplified physical assumptions, simplified geometries or coarser computational grids. While this approach is ubiquitous, we limit to mention a few applications, e.g. [29–32]. Most methods in this category are dependent on the class of differential problems describing the phenomenon or are tailored to a specific model.

## 1.2. Data-driven MOR

In the Loewner framework ([33]) a linear reduced model is derived from transfer function measurements at a collection of *interpolation points*, either in left or right *tangential directions* (i.e. by left or right multiplying the transfer function matrix by the vector corresponding to the tangential direction). The reduced model matrices and arrays are computed in such a way that the reduced model transfer function interpolates the full-order one at the interpolation points and in the tangential directions (see [34,35]). In [9] the Loewner framework has been extended to derive reduced models from time-domain data, instead of frequency-domain data. Although the Loewner framework only applies to linear models, it has recently been extended to bilinear (i.e. linear independently in the state and the input) models ([36]), quadratic-bilinear models ([37]) and to analytic nonlinear models with affine input dependence, by rewriting models with analytic state nonlinearities as quadratic-bilinear models (however increasing the size of the original model, see [38]). The orthonormal vector fitting (OVF) method, suitable for linear systems, is another frequency-domain approach. Starting from transfer function data samples, OVF approximates the transfer function through orthonormal rational functions ([39,40]).

Kriging, originally developed in geosciences ([41]) and also known as Gaussian Process (GP) regression, is widely used to perform MOR for nonlinear models in the steady-state case ([42,43]). It is a regression method that employs, as prior for the outcome of a function, a GP whose covariance function depends on so-called hyperparameters, tuned according to a maximum likelihood principle. This technique has been extended to the dynamical case in [44] under the name of dynamic mapping kriging (DMK); here, by considering the discrete time version of the evolution equation (2) (i.e.  $\mathbf{x}^{k+1} = \mathbf{f}(\mathbf{x}^k, \mathbf{u}^k)$ ), DMK performs kriging on the function  $\mathbf{f}$  starting from sample data for several  $(\mathbf{x}, \mathbf{u})$  pairs. In [45] the authors propose, under the name of Sparse Identification of Nonlinear Dynamics (SINDy), a technique to infer a model for a dynamical system starting from measurements of  $(\mathbf{x}(t), \mathbf{u}(t), \dot{\mathbf{x}}(t))$  tuples. This technique, under the assumption that  $\mathbf{f}$  depends on few combinations of the inputs (such as linear combinations, products, and trigonometric functions), seeks a sparse solution for the coefficients of a predetermined collection of linear and nonlinear terms. Despite DMK and SINDy can be applied to nonlinear systems, both these techniques require access to the full-order state of the model and do not perform any reduction in the state dimension.

The method proposed in [46] stands at the interplay between model-based and data-driven MOR, by combining the RB method with GP regression. In particular the authors build a reduced basis by POD of a collection of snapshots collected by the HF solution of time-dependent parametric differential equations; then, by GP regression, they approximate the map from the parameters and the time instant to the reduced basis coefficients of the solution. This method, however, is restricted to parametric differential equations (i.e. when  $\mathbf{u}(t)$  is constant in time) and cannot be easily extended to the case of models with time-dependent inputs.

### 1.3. Learning models from data

Data-driven MOR, due to its black-box nature, can also be applied when the HF model for the state  $\mathbf{X}$  is not accessible, or may not fit in known families of mathematical models: that is, when one is unable or may not be interested in explicitly building a model. This is the case when a physical system is accessible through measurements and one tries to *identify* the underlying law generating the input-output pairs. This task is commonly known in the field of control theory as *System Identification*, SI (see e.g. [47,48]). Models in the form (2) are known in the SI field as internal-dynamics or state-space models, whereas the most commonly treated form in the SI field is that of external-dynamics models (see NARX/NARMAX models, [49]), i.e. discrete time models in the form

$$\hat{\mathbf{y}}^{k+1} = \eta(\mathbf{y}^k, \mathbf{y}^{k-1}, \dots, \mathbf{y}^{k-p}, \mathbf{u}^k, \mathbf{u}^{k-1}, \dots, \mathbf{u}^{k-q}),$$

where the prediction for the next output  $\hat{\mathbf{y}}^{k+1}$  depend on the value of the previous  $p + 1$  output measurements  $\{\hat{\mathbf{y}}^j\}_{j=k-p}^k$  and the previous  $q + 1$  inputs  $\{\hat{\mathbf{u}}^j\}_{j=k-q}^k$ . However, models in this family are designed for online identification and, most of all, for online predictions: the model must be fed with the output measured at previous steps, so that the *true* output should be available not only at the identification (or training/offline) stage, but also at the prediction (online) stage. Instead, we are looking here for a model that can be used in a stand-alone way in the online phase, once an offline training phase has been carried out. The use of ANNs in the context of nonlinear SI is quite popular ([50,51,49]), even if their application is limited, up to our knowledge, to online identification and prediction of discrete time systems.

In some recent works the authors have developed learning machines, either based on GP ([52,53]) or ANNs ([54–56]), for data-driven solution and *data-driven discovery* of PDEs. However, the learning machine must have knowledge of the form of the equations that generated the observed data. This technique can be applied to linear or nonlinear parametric PDEs, where the parameters of the PDE (e.g. diffusion, reaction coefficients, etc.) are unknown.

In [57] the authors make use of ANNs to perform data-driven discovery of nonlinear dynamical systems. In particular, they train the ANN to minimize the residuals of a given multi-step time-stepping scheme (such as Adams-Bashforth or BDF schemes) on a collection of available snapshots of the full-order state  $\{\mathbf{X}(t^k)\}_{k=1}^M$ . However, this approach, like DMK and SINDy techniques, requires  $\mathbf{X}$  to be accessible and, moreover, does not perform any dimensionality reduction of the state space.

In [58] the authors use ANNs in combination with projection-based MOR for time-dependent models, to approximate the effect of the discarded modes on the retained ones. In [59,60] the authors use machine-learning techniques to model the error of projection-based reduced model of parametrized nonlinear dynamical systems.

### 1.4. Original contributions and outline

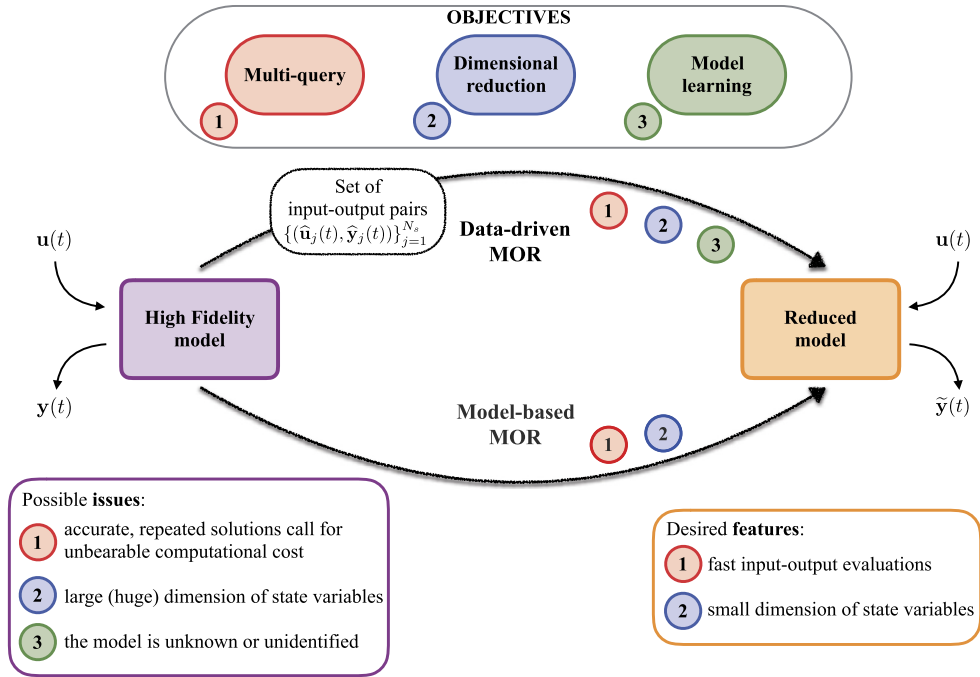
In this work, we address the problem of data-driven MOR for nonlinear dynamical systems (which can be interpreted as a nonlinear SI problem), where we suppose to have no direct access either to the HF model (that is  $\mathbf{F}$  and  $\mathbf{G}$  in Eq. (1)), nor to full-order state observations  $\mathbf{X}(t)$ , but only to input-output pairs  $(\mathbf{u}(t), \mathbf{y}(t))$ . This task is remarkably hard since we aim at the same time at (i) reconstructing the internal state of the system through its reduced description  $\mathbf{x}(t)$  without the possibility of observing the true internal state of the system  $\mathbf{X}(t)$  and (ii) finding a model for the dynamics of  $\mathbf{x}(t)$  itself. We notice that the reconstruction of the system state through  $\mathbf{x}(t)$  is not the final goal, but it is just instrumental to reconstruction the input-output map  $\mathbf{u} \rightarrow \mathbf{y}$ .

We proceed by setting the problem in an abstract form, where we look for the best-approximation of the HF model into a class of simpler models (i.e. the class of reduced models with a prescribed level of complexity). This is an optimization problem, where the unknown is the model itself. Noticeably, we need to carefully select the class of candidate models and to find a suitable representation for the models to define an optimization algorithm for solving the best-approximation problem. Because of their ability to approximate any continuous function with a desired level of accuracy (see [61]) and to learn from data, we represent the right-hand side  $\mathbf{f}$  in (2) through an ANN, that we train to learn the underlying physics from input-output pairs.

In this paper we introduce a general framework which serves multiple purposes (see Fig. 1):

- Building surrogates of time-dependent differential models, which allow for fast evaluations and are suitable for multi-query problems.
- Reducing the dimensionality of the state variables of time-dependent differential models (e.g. when a mathematical model needs to be solved virtually in each point of a computational domain and the overall memory storage must be contained).
- Learning mathematical models from available input-output pairs.

This paper is structured as follows. In Sec. 2 we present our strategy, by rephrasing the model reduction problem in terms of an optimization problem, for which we define the objective functional. Then, in Sec. 3, we present the strategy employed to numerically find a solution of the optimization problem. In particular, we show: (i) how the unknowns of the



**Fig. 1.** Model-based and data-driven MOR. The HF model can have three undesired features: ① it may be computationally demanding, ② it can have a high state dimension, ③ it may not be accessible. A reduced model can be derived either directly from the HF model (model-based MOR), or from input-output pairs generated by the HF model (data-driven MOR). The latter approach allows to learn a model, in the case when the HF model is not accessible (see ③). In both cases, the reduced model represents a surrogate of the HF model allowing for fast evaluations, lowering the computational burden in multi-query applications (see ①), and features a low dimensional state (see ②).

problem are discretized; (ii) how the time discretization is performed; (iii) the optimization algorithm employed. Finally, in Sec. 4, we show the obtained results and we analyze and critically discuss them.

## 2. Model reduction strategy

We start by giving a definition of *model* (in the way this is meant in this paper): an object which maps time-dependent inputs into time-dependent outputs.

### 2.1. The dynamical model

By *model* we intend a general framework including either a physical model (like a natural phenomenon or an engineering process) or a mathematical or numerical model, which associates a time-dependent output to a time-dependent input. We consider a limited time interval  $[0, T]$  and we denote by  $U \subset \mathbb{R}^{N_u}$  and by  $Y \subset \mathbb{R}^{N_y}$  the sets where the input  $\mathbf{u}: [0, T] \rightarrow U$  and the output  $\mathbf{y}: [0, T] \rightarrow Y$  take values, respectively. We name as *experiment* the action of inputting  $\mathbf{u}(t)$  to the model and recording the corresponding output  $\mathbf{y}(t)$  and by *sample* we refer to the couple  $(\mathbf{u}, \mathbf{y})$ . For the sake of abstraction, we make the following minimal assumptions on the model:

- (A1) **Time invariance:** by denoting with  $\mathbf{y}(t)$  the output obtained by starting at time  $t_0$  an experiment with input  $\mathbf{u}(t)$ , the output of another experiment started at time  $t_1 \geq t_0$  with input  $\mathbf{u}(t - (t_1 - t_0))$  is  $\mathbf{y}(t - (t_1 - t_0))$ . Hence in the following we will consider, without loss of generality, each experiment starting from the initial time  $t = t_0 = 0$ .
- (A2) **Existence of an initial state:** at time  $t = 0$ , for each experiment, the model is in the same initial state, that is it always responds in the same way to a prescribed input. Otherwise, the map  $\mathbf{u} \mapsto \mathbf{y}$  would not be well defined.
- (A3) **Causality principle:** The input-output relationship must be consistent with the arrow of time, that is the output of the model is allowed to depend only on previous values of the input and not on future values: given two inputs  $\mathbf{u}_1$  and  $\mathbf{u}_2$ , such that, for some  $t^*$ ,  $\mathbf{u}_1(t) = \mathbf{u}_2(t)$  for  $t \in [0, t^*]$ , the corresponding outputs  $\mathbf{y}_1$  and  $\mathbf{y}_2$  must satisfy  $\mathbf{y}_1(t) = \mathbf{y}_2(t)$  for  $t \in [0, t^*]$ .
- (A4) **No input-output direct dependence:** the output at time  $t$  depends just on the state of the system at the same time, but not (directly) on the input at time  $t$ . As a consequence, thanks to (A2), the output at time  $t = 0$  will be the same for each experiment. Notice that this assumption could be removed, by allowing  $\mathbf{g}$  in (2) to depend also on  $\mathbf{u}(t)$ . However, for the sake of simplicity, we will not consider the latter case in this work.

For simplicity, we consider the case when both the input and output are continuous functions in time. Therefore, the model can be seen as a map  $\varphi: \mathcal{U} \rightarrow \mathcal{Y}$  from the space of input signals  $\mathcal{U} = \mathcal{C}^0([0, T]; U)$  to the space of output signals  $\mathcal{Y} = \mathcal{C}^0([0, T]; Y)$ . Thanks to assumptions (A1) and (A2) the map  $\varphi$  is well defined. Moreover, assumption (A3) can be written as

$$\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U} \quad \forall t^* \in [0, T] \quad \mathbf{u}_1|_{[0, t^*]} = \mathbf{u}_2|_{[0, t^*]} \implies (\varphi \mathbf{u}_1)|_{[0, t^*]} = (\varphi \mathbf{u}_2)|_{[0, t^*]}, \quad (4)$$

where  $\varphi \mathbf{u}_1$  denotes the output  $\mathbf{y}_1 \in \mathcal{Y}$  of the model when the input is  $\mathbf{u}_1 \in \mathcal{U}$  (thus both  $\mathbf{u}_1$  and  $\varphi \mathbf{u}_1$  are functions of time) and  $(\varphi \mathbf{u}_1)|_{[0, s]}$  denotes the restriction of the output  $\mathbf{y}_1$  to the time interval  $[0, s]$ . On the other hand, assumption (A4) entails

$$\exists \mathbf{y}_0 \in Y \quad \text{s.t.} \quad \forall \mathbf{u} \in \mathcal{U} \quad (\varphi \mathbf{u})(0) = \mathbf{y}_0, \quad (5)$$

where  $(\varphi \mathbf{u})(0)$  denotes the output  $\mathbf{y} \in \mathcal{Y}$  of the model – corresponding to the time-dependent input  $\mathbf{u} \in \mathcal{U}$  – evaluated at time  $t = 0$  (i.e.  $\mathbf{y}(0)$ ). Thus, we define the set of all the models associated with the input and output sets  $\mathcal{U}$  and  $\mathcal{Y}$  as

$$\Phi = \{\varphi: \mathcal{U} \rightarrow \mathcal{Y} \quad \text{s.t.} \quad (4) \text{ and } (5) \text{ hold}\},$$

endowed with the norm of the supremum:

$$\|\varphi\|_\Phi = \sup_{\mathbf{u} \in \mathcal{U}} \|\varphi \mathbf{u}\|_{\mathcal{Y}} = \sup_{\mathbf{u} \in \mathcal{U}} \sup_{t \in [0, T]} \|(\varphi \mathbf{u})(t)\|_Y.$$

## 2.2. Building a reduced model

We perform  $N_s$  experiments with the HF model and we collect a set of  $N_s$  input-output pairs:

$$\{(\widehat{\mathbf{u}}_j, \widehat{\mathbf{y}}_j)\}_{j=1, \dots, N_s} \subset \mathcal{U} \times \mathcal{Y}. \quad (6)$$

Then we select a subset of candidate models, which we denote by  $\widehat{\Phi} \subseteq \Phi$ , and we consider the problem of finding the best-approximation of the HF model, in the least-squares sense, in the subset  $\widehat{\Phi}$ :

$$\varphi^* = \underset{\varphi \in \widehat{\Phi}}{\operatorname{argmin}} J(\varphi), \quad (7)$$

where the objective functional is given by

$$J(\varphi) = \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T |\widehat{\mathbf{y}}_j(t) - (\varphi \widehat{\mathbf{u}}_j)(t)|^2 dt. \quad (8)$$

When the measurements of the output  $\widehat{\mathbf{y}}_j$  are affected by Gaussian noise, the least-squares best-approximation corresponds to the maximum-likelihood estimation (see e.g. [62]).

The next step is the choice of the subset of candidate models  $\widehat{\Phi} \subseteq \Phi$ . A possible approach is that of directly approximating the input-output map  $\mathbf{u} \rightarrow \mathbf{y}$  in the time domain (i.e. from  $\mathcal{U}$  to  $\mathcal{Y}$ ), e.g. by means of an ANN which takes as an input a set  $\{\mathbf{u}(t_0), \mathbf{u}(t_1), \dots, \mathbf{u}(t_M)\}$  of values associated to a collection of time instants and returns the corresponding output values  $\{\mathbf{y}(t_0), \mathbf{y}(t_1), \dots, \mathbf{y}(t_M)\}$ . However, working on input and outputs as signals (i.e. in the spaces  $\mathcal{U}$  and  $\mathcal{Y}$ ) would clearly lead to a remarkably large-size problem, potentially making the learning process unaffordable because of its computational complexity. Thus, we pursue a different approach: by exploiting the structure of the elements in  $\Phi$ , which is based on assumptions (A1)–(A4), we restrict the investigation to input-output maps described by systems of ODEs. This dramatically reduces the size of the problem, as we show in the following.

## 2.3. Models described by systems of ODEs

We refer now to the specific class of models, in the framework of Sec 2.1, which are governed by a system of ODEs in the form of (2). Such class represents a subset of  $\Phi$ , as we will show. First, we notice that, given two functions  $\mathbf{f} \in \mathcal{F}_n := \{\mathbf{f} \in \mathcal{C}^0(\mathbb{R}^n \times U; \mathbb{R}^n)\}$ , which are Lipschitz continuous in  $\mathbf{x}$  uniformly in  $\mathbf{u}$  and  $\mathbf{g} \in \mathcal{G}_n := \mathcal{C}^0(\mathbb{R}^n; Y)$  and a vector  $\mathbf{x}_0 \in \mathcal{X}_n \equiv \mathbb{R}^n$  (where the subscript  $n$  stands for the number of internal reduced states), the system (2) identifies a unique map from  $\mathcal{U}$  to  $\mathcal{Y}$ . We denote by  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0}$  such map.

**Proposition 1.** For each  $\mathbf{f} \in \mathcal{F}_n$ ,  $\mathbf{g} \in \mathcal{G}_n$  and  $\mathbf{x}_0 \in \mathcal{X}_n$ , we have  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} \in \Phi$ , that is the input-output map represented by (2) is a model according to the definition of Sec. 2.1.



**Proof.** Thanks to the Picard-Lindelöf theorem, Eq. (2) has a unique solution. Thus, the map  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0}$  is well defined. Moreover, it is continuous from  $\mathcal{U}$  to  $\mathcal{Y}$ , thus the norm in  $\Phi$  is finite. Property (4) is easy to be checked, while (5) holds by setting  $\mathbf{y}_0 = \mathbf{g}(\mathbf{x}_0)$ .  $\square$

Moreover, given the triplet  $\widehat{\mathcal{F}} \subseteq \mathcal{F}_n$ ,  $\widehat{\mathcal{G}} \subseteq \mathcal{G}_n$  and  $\widehat{\mathcal{X}} \subseteq \mathcal{X}_n$ , we define the following subset of models:

$$\Phi^{\widehat{\mathcal{F}}, \widehat{\mathcal{G}}, \widehat{\mathcal{X}}} = \{\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} \in \Phi \text{ s.t. } \mathbf{f} \in \widehat{\mathcal{F}}, \mathbf{g} \in \widehat{\mathcal{G}}, \mathbf{x}_0 \in \widehat{\mathcal{X}}\} \subset \Phi. \quad (9)$$

We have the following result, which states that the expressive power of the class of models with  $n$  state variables grows as  $n$  increases.

**Proposition 2.** *The classes of models with  $n$  internal states are nested, that is:*

$$\Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n} \subseteq \Phi^{\mathcal{F}_m, \mathcal{G}_m, \mathcal{X}_m} \quad \forall n \leq m.$$

**Proof.** Given a model in the form Eq. (2) with  $n$  state variables, by adding  $m - n$  further variables which affect neither the dynamics of the other variables, nor the output, we obtain a model with  $m$  state variables, still representing the same input-output map of the previous model.  $\square$

Moreover, we have the following result.

**Proposition 3.** *Let  $U$  be compact. Suppose that the subsets  $\widehat{\mathcal{F}} \subseteq \mathcal{F}_n$  and  $\widehat{\mathcal{G}} \subseteq \mathcal{G}_n$  are such that the restrictions of their functions to compact sets are dense in the full spaces, that is for each compact set  $E \subset \mathbb{R}^n$  we have:*

$$\forall \varepsilon > 0 \quad \forall \mathbf{f} \in \mathcal{F}_n \quad \exists \widehat{\mathbf{f}} \in \widehat{\mathcal{F}} \quad \forall \mathbf{x} \in E, \mathbf{u} \in U \quad |\mathbf{f}(\mathbf{x}, \mathbf{u}) - \widehat{\mathbf{f}}(\mathbf{x}, \mathbf{u})| \leq \varepsilon, \quad (10)$$

$$\forall \varepsilon > 0 \quad \forall \mathbf{g} \in \mathcal{G}_n \quad \exists \widehat{\mathbf{g}} \in \widehat{\mathcal{G}} \quad \forall \mathbf{x} \in E \quad |\mathbf{g}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x})| \leq \varepsilon. \quad (11)$$

Then, the subset of models  $\Phi^{\widehat{\mathcal{F}}, \widehat{\mathcal{G}}, \mathcal{X}_n}$  is dense in the model space  $\Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n}$ , that is:

$$\forall \varepsilon > 0 \quad \forall \varphi \in \Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n} \quad \exists \widehat{\varphi} \in \Phi^{\widehat{\mathcal{F}}, \widehat{\mathcal{G}}, \mathcal{X}_n} \quad \text{s.t.} \quad \|\varphi - \widehat{\varphi}\|_{\Phi} \leq \varepsilon. \quad (12)$$

**Proof.** By definition, there exists  $\mathbf{f} \in \mathcal{F}_n$ ,  $\mathbf{g} \in \mathcal{G}_n$  and  $\mathbf{x}_0 \in \mathcal{X}_n$  such that  $\varphi = \varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0}$ . First, we show that the state  $\mathbf{x}$  of the HF model  $\varphi$  is bounded. We have:

$$\begin{aligned} \frac{d}{dt} \left( \frac{1}{2} |\mathbf{x}(t)|^2 \right) &= \dot{\mathbf{x}}(t) \cdot \mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \cdot \mathbf{x}(t) \leq |\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))| |\mathbf{x}(t)| \\ &\leq (|\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \mathbf{f}(\mathbf{x}_0, \mathbf{u}(t))| + |\mathbf{f}(\mathbf{x}_0, \mathbf{u}(t))|) |\mathbf{x}(t)|. \end{aligned}$$

By denoting by  $L_f$  the Lipschitz constant of  $\mathbf{f}$ , we have  $|\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \mathbf{f}(\mathbf{x}_0, \mathbf{u}(t))| \leq L_f |\mathbf{x}(t) - \mathbf{x}_0| \leq L_f (|\mathbf{x}(t)| + |\mathbf{x}_0|)$ . Moreover, as  $U$  is compact, we have  $|\mathbf{f}(\mathbf{x}_0, \mathbf{u}(t))| \leq M$  for some finite  $M \in \mathbb{R}$ . Therefore, we have:

$$\frac{d}{dt} \left( \frac{1}{2} |\mathbf{x}(t)|^2 \right) \leq L_f |\mathbf{x}(t)|^2 + (L_f |\mathbf{x}_0| + M) |\mathbf{x}(t)| \leq \left( L_f + \frac{1}{2} \right) |\mathbf{x}(t)|^2 + \frac{1}{2} (L_f |\mathbf{x}_0| + M)^2.$$

Using the Gronwall lemma we get:

$$|\mathbf{x}(t)| \leq \sqrt{(|\mathbf{x}_0|^2 + (L_f |\mathbf{x}_0| + M)^2 T) e^{(2L_f + 1)T}} =: R \quad \forall t \in [0, T].$$

Thanks to (11), there exists  $\widehat{\mathbf{g}} \in \widehat{\mathcal{G}}$  such that  $|\mathbf{g}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x})| \leq \frac{1}{2}\varepsilon$  for  $\mathbf{x} \in \overline{B_{2R}}$ , the closed ball of radius  $2R$ .

Moreover, being  $\mathbf{g}$  continuous, by the Heine-Cantor theorem, it is also uniformly continuous on the compact ball  $\overline{B_{2R}}$ , thus there exists some  $\varepsilon'$  such that for any  $\mathbf{x}_1, \mathbf{x}_2 \in \overline{B_{2R}}$ , if  $|\mathbf{x}_1 - \mathbf{x}_2| \leq \varepsilon'$ ; this implies that  $|\mathbf{g}(\mathbf{x}_1) - \mathbf{g}(\mathbf{x}_2)| \leq \frac{1}{2}\varepsilon$ .

We define the positive number:

$$\varepsilon'' = \min \left\{ \varepsilon', \frac{1}{2}R \right\} \left( T e^{(2L_f + 1)T} \right)^{-1/2}.$$

Thanks to (10), there exists  $\widehat{\mathbf{f}} \in \widehat{\mathcal{F}}$  such that  $|\mathbf{f}(\mathbf{x}, \mathbf{u}) - \widehat{\mathbf{f}}(\mathbf{x}, \mathbf{u})| \leq \varepsilon''$  for  $\mathbf{x} \in \overline{B_{2R}}$ ,  $\mathbf{u} \in U$ .

Consider the model  $\widehat{\varphi} = \varphi_{\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \mathbf{x}_0}$ , whose state is denoted by  $\widehat{\mathbf{x}}(t)$ . Let  $\mathbf{u} \in \mathcal{U}$  be a generic input signal. Now, we show that the trajectory of the state  $\widehat{\mathbf{x}}(t)$  is contained in  $B_{2R}$ . Suppose by contradiction that  $\widehat{\mathbf{x}}(s) = 2R$  for some  $s \in [0, T]$ ,  $\widehat{\mathbf{x}}(t) < 2R$  for  $t < s$ . Then, in the interval  $[0, s)$  the discrepancy between the two states  $\mathbf{z}(t) := \mathbf{x}(t) - \widehat{\mathbf{x}}(t)$  satisfies:

$$\begin{aligned}
\frac{d}{dt} \left( \frac{1}{2} |\mathbf{z}(t)|^2 \right) &= \dot{\mathbf{z}}(t) \cdot \mathbf{z}(t) = [\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \widehat{\mathbf{f}}(\widehat{\mathbf{x}}(t), \mathbf{u}(t))] \cdot \mathbf{z}(t) \\
&= |\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \mathbf{f}(\widehat{\mathbf{x}}(t), \mathbf{u}(t))| |\mathbf{z}(t)| + |\mathbf{f}(\widehat{\mathbf{x}}(t), \mathbf{u}(t)) - \widehat{\mathbf{f}}(\widehat{\mathbf{x}}(t), \mathbf{u}(t))| |\mathbf{z}(t)| \\
&\leq L_f |\mathbf{z}(t)|^2 + \varepsilon'' |\mathbf{z}(t)| \leq \left( L_f + \frac{1}{2} \right) |\mathbf{z}(t)|^2 + \frac{1}{2} (\varepsilon'')^2.
\end{aligned} \tag{13}$$

By Gronwall inequality we get

$$|\mathbf{x}(t) - \widehat{\mathbf{x}}(t)| \leq \varepsilon'' \sqrt{T e^{(2L_f+1)T}} \leq \frac{1}{2} R,$$

where the second inequality follows from the definition of  $\varepsilon''$ . The, we have  $|\widehat{\mathbf{x}}(t)| \leq |\mathbf{x}(t) - \widehat{\mathbf{x}}(t)| + |\mathbf{x}(t)| \leq \frac{3}{2} R$  for  $t \in [0, s)$ . Since  $|\widehat{\mathbf{x}}(s)| = 2R$ , we reach a contradiction and we have proved that the trajectory of  $\widehat{\mathbf{x}}(t)$  is contained into  $B_{2R}$ . Therefore, inequality (13) holds on the whole time interval  $[0, T]$  and we conclude:

$$|\mathbf{x}(t) - \widehat{\mathbf{x}}(t)| \leq \varepsilon'' \sqrt{T e^{(2L_f+1)T}} \leq \varepsilon' \quad t \in [0, T],$$

which entails  $|\mathbf{g}(\mathbf{x}(t)) - \mathbf{g}(\widehat{\mathbf{x}}(t))| \leq \frac{1}{2} \varepsilon$  for  $t \in [0, T]$ . Finally, we have

$$|(\boldsymbol{\varphi}\mathbf{u})(t) - (\widehat{\boldsymbol{\varphi}}\mathbf{u})(t)| = |\mathbf{g}(\mathbf{x}(t)) - \widehat{\mathbf{g}}(\widehat{\mathbf{x}}(t))| \leq |\mathbf{g}(\mathbf{x}(t)) - \mathbf{g}(\widehat{\mathbf{x}}(t))| + |\mathbf{g}(\widehat{\mathbf{x}}(t)) - \widehat{\mathbf{g}}(\widehat{\mathbf{x}}(t))| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \quad \square$$

Proposition 3 states that if the subsets  $\widehat{\mathcal{F}}$  and  $\widehat{\mathcal{G}}$  approximate in a suitable sense (see Eqs. (10)–(11)) the spaces  $\mathcal{F}_n$  and  $\mathcal{G}_n$  respectively, the class of candidate models  $\Phi^{\widehat{\mathcal{F}}, \widehat{\mathcal{G}}, \mathcal{X}_n}$  approximate the space of models  $\Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n}$  with arbitrary accuracy.

We notice that by setting  $\widehat{\Phi} = \Phi^{\widehat{\mathcal{F}}, \widehat{\mathcal{G}}, \widehat{\mathcal{X}}}$ , the abstract problem (7) reads:

$$\begin{cases} \min_{\mathbf{f} \in \widehat{\mathcal{F}}, \mathbf{g} \in \widehat{\mathcal{G}}, \mathbf{x}_0 \in \widehat{\mathcal{X}}} & \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T |\widehat{\mathbf{y}}_j(t) - \mathbf{g}(\mathbf{x}_j(t))|^2 dt \\ \text{s.t.} & \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)), \quad t \in (0, T], \quad j = 1, \dots, N_s \\ & \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1, \dots, N_s. \end{cases} \tag{14}$$

We are thus addressing a least-squares minimization problem where the design variables are the two functions  $\mathbf{f} \in \widehat{\mathcal{F}}$  and  $\mathbf{g} \in \widehat{\mathcal{G}}$  and the vector  $\mathbf{x}_0 \in \widehat{\mathcal{X}}$ .

#### 2.4. Non-uniqueness of the representation

**Remark 1.** Given a model  $\boldsymbol{\varphi}_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} \in \Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n}$ , its representation in terms of  $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$  may not be unique. Indeed, by taking any invertible and sufficiently regular map  $\mathbf{h}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ , let us consider the change of variables  $\widetilde{\mathbf{x}} = \mathbf{h}(\mathbf{x})$  and define

$$\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}, \mathbf{u}) = (\nabla \mathbf{h} \circ \mathbf{h}^{-1})(\widetilde{\mathbf{x}}) \mathbf{f}(\mathbf{h}^{-1}(\widetilde{\mathbf{x}}), \mathbf{u}), \quad \widetilde{\mathbf{g}}(\widetilde{\mathbf{x}}) = \mathbf{g}(\mathbf{h}^{-1}(\widetilde{\mathbf{x}})), \quad \widetilde{\mathbf{x}}_0 = \mathbf{h}(\mathbf{x}_0). \tag{15}$$

We have  $\boldsymbol{\varphi}_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} = \boldsymbol{\varphi}_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{x}}_0}$  (i.e. the input-output map represented by the two models is equivalent). As a particular case, for any  $\alpha \in \mathbb{R} \setminus \{0\}$ , we have that, with the transformation  $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}, \mathbf{u}) = \alpha \mathbf{f}(\widetilde{\mathbf{x}}/\alpha, \mathbf{u})$ ,  $\widetilde{\mathbf{g}}(\widetilde{\mathbf{x}}) = \mathbf{g}(\widetilde{\mathbf{x}}/\alpha)$ ,  $\widetilde{\mathbf{x}}_0 = \alpha \mathbf{x}_0$ , the triplets  $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$  and  $(\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{x}}_0)$  identify the same model.

Because of Remark 1, the best-approximation problem (7) might be ill-posed. Indeed, if the spaces  $\widehat{\mathcal{F}}$ ,  $\widehat{\mathcal{G}}$  and  $\widehat{\mathcal{X}}$  are wide enough to contain both  $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$  and – according to (15) – their equivalent counterparts  $(\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{x}}_0)$ , the solution of problem (7) may lose uniqueness in terms of its  $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$  representation. This is certainly an issue since non-uniqueness can deteriorate the performance of the optimization algorithm. Nevertheless, we can reduce the size of the spaces  $\widehat{\mathcal{F}}$ ,  $\widehat{\mathcal{G}}$  and  $\widehat{\mathcal{X}}$  by imposing specific constraints on the solution to choose a priori a representative solution for a given class of equivalent solutions. In such a way we can restrict the design space for the optimization problem without ruling out possible solutions, thus reducing its complexity. We now show two possible ways of performing this task.

##### 2.4.1. Partial disambiguation by constraining $\mathbf{x}_0$

Consider a model  $\boldsymbol{\varphi}_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} \in \Phi$ . Consider then the (invertible) state transformation  $\widetilde{\mathbf{x}} = \mathbf{h}_1(\mathbf{x}) = \mathbf{x} - \mathbf{x}_0$ . By applying the transformation (15), we get an equivalent model  $\boldsymbol{\varphi}_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{x}}_0}$ , where  $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}, \mathbf{u}) = \mathbf{f}(\widetilde{\mathbf{x}} + \mathbf{x}_0, \mathbf{u})$ ,  $\widetilde{\mathbf{g}}(\widetilde{\mathbf{x}}) = \mathbf{g}(\widetilde{\mathbf{x}} + \mathbf{x}_0)$  and  $\widetilde{\mathbf{x}}_0 = \mathbf{0}$ . Thanks to this property, when we look for the solution of the best-approximation problem (7) we suppose, without loss of generality that  $\mathbf{x}_0 = \mathbf{0}$ . This is equivalent to reducing the set of possible initial states to the singleton  $\widehat{\mathcal{X}} = \{\mathbf{0}\}$ , or equivalently, to minimize the cost functional  $J$  under the constraint  $\mathbf{x}_0 = \mathbf{0}$ . The statement of the best-approximation problem improves since the number of design variables decreases (the design variables are now just  $\mathbf{f}$  and  $\mathbf{g}$ ) and we have disambiguated among a number of equivalent solutions, without ruling out possible solutions.



#### 2.4.2. Partial disambiguation by constraining $\mathbf{g}$ and $\mathbf{x}_0$

Consequence of Remark 1 is that the state variables  $\mathbf{x}$  are just auxiliary variables to track the time evolution of the internal state of the model, not necessarily inferring a clear physical interpretation. There is thus large freedom in their choice. Consider the case when  $n \geq N_y$ : one could possibly decide a priori to force the first  $N_y$  state variables  $x_1, \dots, x_{N_y}$  to coincide with the outputs  $y_1, \dots, y_{N_y}$ . A natural question is then the following: given a model  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0}$ , is it always possible to rewrite it in an equivalent form such that the first  $N_y$  state variables coincide with the output itself? If the answer is affirmative, then we can restrict ourselves to models such that  $\mathbf{g}(\mathbf{x})$  is the function extracting the first  $N_y$  component of a vector, which we denote by  $\boldsymbol{\pi}^{N_y}(\mathbf{x}) = (x_1, x_2, \dots, x_{N_y})^T$ .

To answer this question, consider a model  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0} \in \Phi$  and suppose that there exists a smooth function  $\mathbf{q}: \mathbb{R}^n \rightarrow \mathbb{R}^{n-N_y}$  – we will address later the existence issue – such that  $\mathbf{h}_2(\mathbf{x}) = (\mathbf{g}^T(\mathbf{x}), \mathbf{q}^T(\mathbf{x}))^T$  is invertible (where  $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^{N_y}$ ,  $\mathbf{q}(\mathbf{x}) \in \mathbb{R}^{n-N_y}$  and thus  $\mathbf{h}_2(\mathbf{x}) \in \mathbb{R}^n$ ). In such a case, by applying the transformation (15) with  $\mathbf{h}_2$ , we get the equivalent model  $\varphi_{\tilde{\mathbf{f}}, \tilde{\mathbf{g}}, \tilde{\mathbf{x}}_0}$  where:

$$\begin{aligned}\tilde{\mathbf{f}}(\tilde{\mathbf{x}}, \mathbf{u}) &= (\nabla \mathbf{h}_2 \circ \mathbf{h}_2^{-1})(\tilde{\mathbf{x}}) \mathbf{f}(\mathbf{h}_2^{-1}(\tilde{\mathbf{x}}), \mathbf{u}) \\ \tilde{\mathbf{g}}(\tilde{\mathbf{x}}) &= \boldsymbol{\pi}^{N_y}(\tilde{\mathbf{x}}) \\ \tilde{\mathbf{x}}_0 &= (\mathbf{g}^T(\mathbf{x}_0), \mathbf{q}^T(\mathbf{x}_0))^T,\end{aligned}\tag{16}$$

for which the desired property, that the output is given by the first  $N_y$  state variables, holds. We notice that, in the expression of the initial condition  $\tilde{\mathbf{x}}_0$ , we can substitute  $\mathbf{g}(\mathbf{x}_0) = \mathbf{y}_0$ , which is available from the measurements. Moreover, as in the previous case, we may set, without loss of generality,  $\mathbf{q}(\mathbf{x}_0) = \mathbf{0}$  (this can be obtained by applying once again (15) with the transformation  $\mathbf{h}_3(\mathbf{x}) = \mathbf{h}_2(\mathbf{x}) - (\mathbf{0}^T, \mathbf{q}^T(\mathbf{x}_0))^T$ ). To summarize, we get a model in the form:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{x}(0) = (\mathbf{y}_0^T, \mathbf{0}^T)^T, \end{cases}\tag{17}$$

where the state is written in the form  $\mathbf{x} = (\mathbf{y}^T, \mathbf{z}^T)^T$ , where  $\mathbf{z}(t)$  are the other auxiliary variables. We have not ruled out yet all the ambiguity since, being the output transparent to such auxiliary variables, the latter can be still subject to invertible transformations which does not affect the input-output map of the model; nonetheless, the size of design space has been dramatically reduced since the unique design variable left is  $\mathbf{f}$ . We notice that this reduction is afforded by setting  $\hat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$  and  $\hat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$ . We have thus shown that all the models admitting the existence of a function  $\mathbf{q}: \mathbb{R}^n \rightarrow \mathbb{R}^{n-N_y}$  such that  $\mathbf{h}_2(\mathbf{x}) = (\mathbf{g}^T(\mathbf{x}), \mathbf{q}^T(\mathbf{x}))^T$  is invertible are in fact equivalent to a model in the form of (17).

Clearly, this hypothesis is not fulfilled by every model  $\varphi_{\mathbf{f}, \mathbf{g}, \mathbf{x}_0}$ . If, for instance, the outputs  $y_1, \dots, y_{N_y}$  are linearly dependent, then  $\mathbf{h}_2$  cannot be invertible. However, this case does not have any practical interest since in such case the dimension of the output can be reduced. Conversely, let us consider a point  $\mathbf{x}^*$ , and suppose that  $\nabla \mathbf{g}(\mathbf{x}^*)$  has full rank. Then, let  $\{\mathbf{v}_i\}_{i=1}^{n-N_y}$  be an orthonormal basis of  $\text{Col}(\nabla \mathbf{g}(\mathbf{x}^*))^\perp$ . By defining  $q_i(\mathbf{x}) := (\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{v}_i$  for  $i = 1, \dots, n - N_y$ , the Jacobian of  $\mathbf{h}_2(\mathbf{x}) = (\mathbf{g}^T(\mathbf{x}), \mathbf{q}^T(\mathbf{x}))^T$  has full-rank and  $\mathbf{h}_2(\mathbf{x})$  is thus locally invertible. Even if this property holds just locally, the message is that the constraint  $\mathbf{g} = \boldsymbol{\pi}^{N_y}$  keeps virtually intact the capacity of the class of models to approximate a given HF model. In Sec. 4.2 we will address numerically this issue.

#### 2.5. The best-approximation problem

We will address both the cases considered in Sec. 2.4, namely:

1.  $\hat{\mathcal{X}} = \{\mathbf{0}\}$ , which we refer to as *output-outside-the-state* (the output is a function of the state, but it is not part of the state);
2.  $\hat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$  and  $\hat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$ , which we refer to as *output-inside-the-state* (the first  $N_y$  state variables coincide with the output variables).

We notice that the second approach is available only for  $n \geq N_y$ . In both the cases,  $\mathbf{x}_0$  is given and thus it is not counted as a design variable. Therefore, the best-approximation problem, in the most general case, consists of:

- Collecting the input-output observations (6) of the HF model to be approximated (i.e. to be reduced or to be identified).
- Selecting a suitable state dimension  $n \geq 1$  for the reduced model, the subset  $\hat{\mathcal{F}}$  and, in the output-outside-the-state case, the subset  $\hat{\mathcal{G}}$ .
- Solving the abstract problem (7); this reads, in the output-outside-the-state case:

$$\begin{cases} \min_{\mathbf{f} \in \hat{\mathcal{F}}, \mathbf{g} \in \hat{\mathcal{G}}} & \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T |\hat{\mathbf{y}}_j(t) - \mathbf{g}(\mathbf{x}_j(t))|^2 dt \\ \text{s.t.} & \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \hat{\mathbf{u}}_j(t)), \quad t \in (0, T], \quad j = 1, \dots, N_s \\ & \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1, \dots, N_s, \end{cases}\tag{18}$$

where  $\mathbf{x}_0 = \mathbf{0}$ , while in the output-inside-the-state case:

$$\begin{cases} \min_{\mathbf{f} \in \widehat{\mathcal{F}}} & \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T |\widehat{\mathbf{y}}_j(t) - \boldsymbol{\pi}^{N_y}(\mathbf{x}_j(t))|^2 dt \\ \text{s.t.} & \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)), \quad t \in (0, T], \quad j = 1, \dots, N_s \\ & \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1, \dots, N_s, \end{cases} \quad (19)$$

where  $\mathbf{x}_0 = (\mathbf{y}_0^T, \mathbf{0}^T)^T$ . We notice that (19) can be seen as a particular case of (18) (by setting  $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$ ); therefore, in the following we will confine ourselves, without loss of generality, to (18).

## 2.6. Solution strategy

The unknowns of problem (19) and, more in general, (18) are the functions  $\mathbf{f}$  and, in the output-outside-the-state case, also  $\mathbf{g}$ . We are thus performing optimization in function spaces. One should look at the first-order optimality conditions for this optimization problem, when  $\widehat{\mathcal{F}} = \mathcal{F}_n$  and  $\widehat{\mathcal{G}} = \mathcal{G}_n$ . With this aim, we write the Lagrangian functional associated to problem (18). Thus, we introduce a family of Lagrange multipliers  $\mathbf{w}_j \in C^0([0, T]; \mathbb{R}^n)$ , for  $j = 1, \dots, N_s$ , associated to the constraints given by the state equations:

$$\begin{aligned} \mathcal{L}(\mathbf{f}, \mathbf{g}, \{\mathbf{x}_j\}_j, \{\mathbf{w}_j\}_j) &= \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T |\widehat{\mathbf{y}}_j(t) - \mathbf{g}(\mathbf{x}_j(t))|^2 dt \\ &\quad - \sum_{j=1}^{N_s} \int_0^T \mathbf{w}_j(t) \cdot (\dot{\mathbf{x}}_j(t) - \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t))) dt \\ &\quad - \sum_{j=1}^{N_s} \mathbf{w}_j(0) \cdot (\mathbf{x}_j(0) - \mathbf{x}_0). \end{aligned} \quad (20)$$

The adjoint equations are recovered by setting to zero the variation of the Lagrangian with respect to the state variables:

$$\begin{cases} -\dot{\mathbf{w}}_j(t) &= \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}_j(t))^T (\mathbf{g}(\mathbf{x}_j(t)) - \widehat{\mathbf{y}}_j(t)) + \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t))^T \mathbf{w}_j(t) \quad t \in [0, T] \\ \mathbf{w}_j(T) &= \mathbf{0}. \end{cases} \quad (21)$$

The first-order optimality conditions are obtained by setting equal to zero the Gâteaux derivative of the objective functional  $J$  with respect to the two unknowns  $\mathbf{f} \in \mathcal{F}_n$  and  $\mathbf{g} \in \mathcal{G}_n$ , for any possible variations  $\delta \mathbf{f} \in \mathcal{F}_n$  and  $\delta \mathbf{g} \in \mathcal{G}_n$ :

$$\begin{cases} \langle \frac{\partial J}{\partial \mathbf{f}}, \delta \mathbf{f} \rangle = \sum_{j=1}^k \int_0^T \delta \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)) \cdot \mathbf{w}_j(t) dt = 0 & \forall \delta \mathbf{f} \in \mathcal{F}_n \\ \langle \frac{\partial J}{\partial \mathbf{g}}, \delta \mathbf{g} \rangle = \sum_{j=1}^k \int_0^T \delta \mathbf{g}(\mathbf{x}_j(t)) \cdot (\mathbf{g}(\mathbf{x}_j(t)) - \widehat{\mathbf{y}}_j(t)) dt = 0 & \forall \delta \mathbf{g} \in \mathcal{G}_n. \end{cases} \quad (22)$$

In Eqs. (21)–(22), which state the conditions that the unknowns  $\mathbf{f}$  and  $\mathbf{g}$  must fulfill to yield a (local) minimum of the cost functional  $J$ ,  $\mathbf{f}$  and  $\mathbf{g}$  are evaluated just in  $\mathbf{x}_j(t)$  and  $\widehat{\mathbf{u}}_j(t)$  for  $j = 1, \dots, N_s$  and  $t \in [0, T]$ . However, the trajectories of  $\mathbf{x}_j(t)$  and  $\widehat{\mathbf{u}}_j(t)$  do not fill the whole spaces  $\mathbb{R}^n$  and  $U$ . Therefore, the problem of fulfilling Eqs. (21)–(22) is ill-posed, being underdetermined. This has to be compensated for by proper regularization of the unknown themselves. Hence, the differentials in (22) cannot be written in gradient form and thus gradient descent strategies cannot be applied for an iterative optimization procedure.

Our strategy is that of parametrizing both the functions  $\mathbf{f}$  and  $\mathbf{g}$  by a finite set of real parameters, which we denote respectively  $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$  and  $\boldsymbol{\nu} \in \mathbb{R}^{N_g}$ , and then tackling problem (18) by optimizing with respect to  $\boldsymbol{\mu}$ ,  $\boldsymbol{\nu}$  (to stress the parametrization, we write  $\mathbf{f}(\mathbf{x}, \mathbf{u}; \boldsymbol{\mu})$  and  $\mathbf{g}(\mathbf{x}; \boldsymbol{\nu})$ ). In this way the desired regularization is obtained by controlling the size of  $N_f$  and  $N_g$  since the complexity of candidate models is controlled by  $N_f$  and  $N_g$ . Indeed, by writing the variation of the functions as  $\delta \mathbf{f} = \nabla_{\boldsymbol{\mu}} \mathbf{f} \delta \boldsymbol{\mu}$  and  $\delta \mathbf{g} = \nabla_{\boldsymbol{\nu}} \mathbf{g} \delta \boldsymbol{\nu}$ , the sensitivity of the objective functional can now be written in gradient form:

$$\begin{cases} \nabla_{\boldsymbol{\mu}} J = \sum_{j=1}^k \int_0^T \nabla_{\boldsymbol{\mu}} \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t); \boldsymbol{\mu})^T \mathbf{w}_j(t) dt \\ \nabla_{\boldsymbol{\nu}} J = \sum_{j=1}^k \int_0^T \nabla_{\boldsymbol{\nu}} \mathbf{g}(\mathbf{x}_j(t); \boldsymbol{\nu})^T (\mathbf{g}(\mathbf{x}_j(t); \boldsymbol{\nu}) - \widehat{\mathbf{y}}_j(t)) dt. \end{cases} \quad (23)$$

We notice that the same result can be obtained by differentiating the Lagrangian functional (20) with respect to  $\boldsymbol{\mu}$  and  $\boldsymbol{\nu}$ . In Sec. 3 we will derive the discrete counterpart of (21) and (23), employed to numerically solve problem (18).

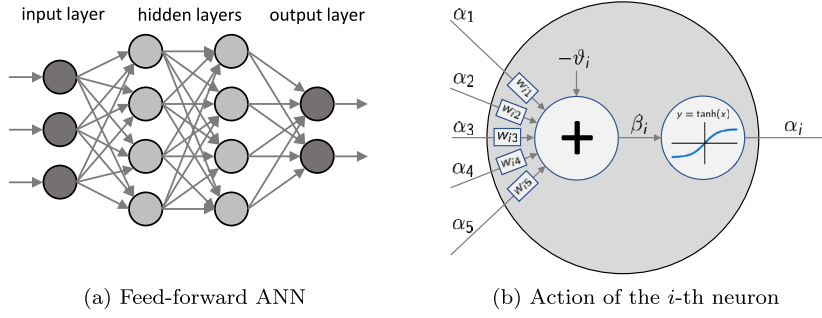


Fig. 2. Scheme of a feed-forward ANN with two hidden layers (a) and of a general neuron (b).

The parametrization of  $\mathbf{f}$  and  $\mathbf{g}$  can be obtained in different manners, such as by polynomials, piecewise polynomials, truncated Fourier series, splines, NURBS, etc. Here we choose to represent them by means of ANNs, which can be seen as nonlinear maps parametrized by a finite number of real parameters (see Sec. 3.1). Our choice is driven by the universal approximation properties of ANNs (see [63,61]) and their recognized properties of learning from data.

### 3. Optimization strategy

We address the solution of the model reduction problem, formulated as the constrained optimization problem (18). We briefly recall ANNs and their universal approximation properties and, then, we show how ANNs can be employed to approximate time-dependent models.

#### 3.1. Artificial Neural Networks

An ANN consists of a number of simple processing units (the *neurons*), each one incorporating a linear and a nonlinear application, interconnected to form a complex network (see e.g. [64]). We consider feed-forward ANNs (known as multilayer perceptrons), consisting of  $n_L$  layers of neurons (the input layer,  $n_L - 2$  hidden layers, and the output layer), where each neuron of a given layer has a connection (or synapse) towards each neuron of the next layer (see Fig. 2). A feed-forward ANN represents a class of functions written as

$$\mathbf{f}(\mathbf{p}; \boldsymbol{\mu}) = W_{n_L-1} f_{\text{act}}(\dots W_2 f_{\text{act}}(W_1 \mathbf{p} - \boldsymbol{\vartheta}_1) - \boldsymbol{\vartheta}_2 \dots) - \boldsymbol{\vartheta}_{n_L-1}, \quad (24)$$

where  $\mathbf{p} \in \mathbb{R}^{n_I}$  is the input vector and  $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$  is the parameters vector, collecting the entries of the weights matrices  $W_i$  and of the bias vectors  $\boldsymbol{\vartheta}_i$ . The action of the nonlinear activation function  $f_{\text{act}}$  must be interpreted component-wise. In this work, we use the sigmoidal activation function  $f_{\text{act}}(s) = \tanh(s)$ .

The calculation by means of the chain rule of the sensitivity of the ANN output with respect to the input  $\mathbf{p}$  or to the parameters  $\boldsymbol{\mu}$  (which we denote respectively by  $\nabla_{\mathbf{p}} \mathbf{f}$  and  $\nabla_{\boldsymbol{\mu}} \mathbf{f}$ ) gives rise to the so-called *back-propagation* [64].

In virtue of the *universal approximation theorem* ([61]), ANNs with a single hidden layer can approximate with arbitrarily small error any continuous function on a compact set, provided that a sufficient number of hidden neurons are employed.

##### 3.1.1. Normalization

Even if ANNs can work with data values spanning different order of magnitudes, their performance is optimal when both input and output data are normalized. Therefore, before training the network, we normalize both the input  $\mathbf{u}$  and the output  $\mathbf{y}$  so that their components take values in the interval  $[-1, 1]$ . Moreover, since the output of the ANN representing  $\mathbf{f}$  are time derivatives, we also normalize time with respect to the fastest time scale associated with the HF model (1), which can be inferred from the training set.

Normalizing inputs, outputs and time scales do not ensure that the state variables (or just the hidden variables in the output-inside-the-state case) lay in the range  $[-1, 1]$ , because of their hidden nature (see Sec. 2.4). Therefore, at each optimization epoch, if the mean square value of a component of the state exceeds a lower or an upper bound (that we set to 0.1 and 2 respectively), we renormalize it. To perform this task, we exploit the fact that the parameters of the ANN arising from another ANN after an invertible and affine change of variables can be derived as follows. Consider the following affine transformation of both the input and the output of the ANN  $\mathbf{q} = \mathbf{f}(\mathbf{p}; \boldsymbol{\mu})$ , i.e.  $\tilde{\mathbf{p}} = \mathbf{A}\mathbf{p} + \mathbf{b}$ ,  $\tilde{\mathbf{q}} = \mathbf{C}\mathbf{q} + \mathbf{d}$ . By simple calculations it turns out that the weights  $\tilde{\boldsymbol{\mu}}$  such that  $\tilde{\mathbf{q}} = \mathbf{f}(\tilde{\mathbf{p}}; \tilde{\boldsymbol{\mu}})$  are obtained by substituting the weights and the biases associated to the first and last layers of synapses as follows:

$$\begin{aligned} \tilde{W}_1 &= W_1 A^{-1}, & \tilde{\boldsymbol{\vartheta}}_1 &= \boldsymbol{\vartheta}_1 - W_1 A^{-1} \mathbf{b}, \\ \tilde{W}_{n_L-1} &= C W_{n_L-1}, & \tilde{\boldsymbol{\vartheta}}_{n_L-1} &= C \boldsymbol{\vartheta}_{n_L-1} - \mathbf{d}. \end{aligned}$$

### 3.2. Representation of the unknowns in terms of ANN

Motivated by the universal approximation property of ANNs, we choose the spaces of candidate functions  $\widehat{\mathcal{F}}$  and  $\widehat{\mathcal{G}}$  as subsets of the space of functions represented by ANNs. We denote by  $\mathcal{F}_n^{\text{ANN}}$  the space of ANNs with  $n + N_u$  input neurons and  $n$  output neurons and by  $\mathcal{G}_n^{\text{ANN}}$  the space of ANNs with  $n$  and  $N_y$  input and output neurons, respectively. We have the following result:

**Proposition 4.** *If  $U$  is compact, the space of ANN models  $\Phi^{\mathcal{F}_n^{\text{ANN}}, \mathcal{G}_n^{\text{ANN}}, \mathcal{X}_n}$  is dense in the model space  $\Phi^{\mathcal{F}_n, \mathcal{G}_n, \mathcal{X}_n}$ .*

**Proof.** ANNs are by construction Lipschitz continuous, being the composition of Lipschitz continuous functions; thus, we have  $\mathcal{F}_n^{\text{ANN}} \subset \mathcal{F}_n$  and  $\mathcal{G}_n^{\text{ANN}} \subset \mathcal{G}_n$ . Properties (10) and (11) hold by the universal approximation theorem ([61]). Hence, the thesis follows from Proposition 3.  $\square$

We conclude that models represented by ANNs are universal approximators of the class of models described by systems of ODEs.

### 3.3. Discretization of the state equation and of the objective functional

In order to numerically approximate problem (18), we discretize both the state equation (2) and the objective functional  $J$  (Eq. (8)). We split the time domain  $[0, T]$  into a collection of time instants  $0 = t_0 < t_1 < \dots < t_M = T$ . For simplicity, we consider the case of constant time step  $\Delta t$  (i.e.  $t_k = k\Delta t$  for  $k = 0, \dots, M$ ) as the generalization to the varying time step case is straightforward. On the other hand, it is convenient to consider the case when the experiments have different durations. Therefore, we suppose that the  $j$ -th experiment takes place in the interval  $[0, T_j]$ , where  $T_j = M_j\Delta t$  and we denote  $\mathbf{u}_j^k = \widehat{\mathbf{u}}_j(t_k) \in U$  and  $\mathbf{y}_j^k = \widehat{\mathbf{y}}_j(t_k) \in Y$  the input and the output at discrete times. The discretized version of the objective functional  $J$  thus reads

$$\mathcal{J} = \frac{1}{2} \Delta t \sum_{j=1}^{N_s} \sum_{k=0}^{M_j-1} |\mathbf{y}_j^k - \mathbf{g}(\mathbf{x}_j^k; \mathbf{v})|^2. \quad (25)$$

To simplify as much as possible the computational burden of the numerical solution of the state equation, which has to be performed many times in the optimization loop, we choose to discretize it by means of the forward Euler scheme. Thus, the discrete counterpart of problem (18) reads:

$$\begin{cases} \min_{(\boldsymbol{\mu}, \mathbf{v}) \in \mathbb{R}^{N_f + N_g}} & \frac{1}{2} \Delta t \sum_{j=1}^{N_s} \sum_{k=0}^{M_j-1} |\mathbf{y}_j^k - \mathbf{g}(\mathbf{x}_j^k; \mathbf{v})|^2 \\ \text{s.t.} & \mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \Delta t \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu}), \quad k = 0, \dots, M_j - 1, \quad j = 1, \dots, N_s \\ & \mathbf{x}_j^0 = \mathbf{x}_0, \quad j = 1, \dots, N_s. \end{cases} \quad (26)$$

We notice that, after time discretization, the ANN model of Eq. (26) has a structure resembling that of a *recurrent neural network* (RNN), that is an ANN whose internal state is fed back to the input after each evaluation of the network (see e.g. [65]). RNNs are widely used for machine learning tasks involving sequential inputs, such as handwriting recognition, speech recognition and natural language processing, and also for time series prediction ([66–68]). However, the approach presented in the present work is more general since it allows to recast the MOR problem within the setting of a best-approximation problem of a differential equation. Moreover, by addressing the problem at the continuous level and thanks to the setting of (14), the reduced model that we obtain by solving the best-approximation problem is independent of the RNN structure used in the training stage. In other words, once the optimization problem in the discrete setting (26) has been solved (i.e. the ANN has been trained), the reduced model is available in the form of Eq. (2). This allows more flexibility in the choice of the time discretization scheme and time step size than with RNN, wherein the forward Euler scheme is exclusively used with fixed  $\Delta t$ . A further advantage is the possibility of coupling the ROM problem with other mathematical models.

Finally, we notice that the RNN structure resulting by the time discretization is not restricted to this specific choice of time stepping scheme; therefore, it is preserved by employing higher-order methods, such as Runge-Kutta schemes.

### 3.4. Training the ANN: optimization algorithm

Problem (26) can be written in the form of the following nonlinear least-squares problem:

$$\min_{\boldsymbol{\xi}} \frac{1}{2} |\mathbf{r}(\boldsymbol{\xi})|^2, \quad (27)$$

where  $\xi = (\mu^T, \nu^T)^T$  is the vector collecting all the design variables and  $\mathbf{r}$  is the vector of residuals, containing all the terms in the following form, for  $j = 1, \dots, N_s$ ,  $k = 0, \dots, M_j - 1$ ,  $h = 1, \dots, N_y$ :

$$r^{j,k,h} = \sqrt{\Delta t} (\mathbf{g}(\mathbf{x}_j^k; \nu) - \mathbf{y}_j^k) \cdot \mathbf{e}_h,$$

where  $\mathbf{e}_h$  denotes the  $h$ -th element of the canonical basis of  $\mathbb{R}^{N_y}$ . To numerically find an approximate solution of this minimization problem we employ the Levenberg-Marquardt method, which is designed for least-squares problems in the form (27) (see e.g. [69]). At each iteration  $k$  of the optimization loop, one finds the descent direction  $\mathbf{d}^{(k)}$  by solving the following problem:

$$\left( \left( \nabla \mathbf{r}^{(k)} \right)^T \nabla \mathbf{r}^{(k)} + \lambda^{(k)} \mathbb{I} \right) \mathbf{d}^{(k)} = - \left( \nabla \mathbf{r}^{(k)} \right)^T \mathbf{r}^{(k)},$$

where  $\lambda^{(k)} \geq 0$  is a weight and  $\mathbf{r}^{(k)}$  denotes  $\mathbf{r}(\xi^{(k)})$ . The update of the solution follows the rule  $\xi^{(k+1)} = \xi^{(k)} + \gamma^{(k)} \mathbf{d}^{(k)}$ , where the step length  $\gamma^{(k)}$  is selected by means of line-search in such a way that the Wolfe conditions are fulfilled (see [69] for details). Specifically, in this work we employ the line-search Algorithm 3.5 in ([69]).

The Levenberg-Marquardt descent direction can be seen as a combination of the steepest-descent direction (which is recovered for  $\lambda^{(k)} \gg 1$ ) with the Gauss-Newton direction ( $\lambda^{(k)} = 0$ ), which is an approximation of the Newton direction obtained by neglecting the quadratic term in the computation of the Hessian. We choose  $\lambda^{(k)} = \min \{ |\mathbf{r}^{(0)}|^2, |(\nabla \mathbf{r}^{(k)})^T \mathbf{r}^{(k)}| \}$ , which ensures superlinear convergence for the method, provided that the objective functional is twice continuously differentiable (see [70]).

We adopt a random initialization of the design variables  $\xi^{(0)}$ , by taking independent samples within the standard normal distribution.

### 3.5. Computation of sensitivities

The Levenberg-Marquardt method requires the computation of the sensitivities of the residuals  $r^{j,k,h}$  with respect to the unknown parameters  $\mu$  and  $\nu$ . Since the map from  $\mu$  to the variables  $\mathbf{x}_j^k$  is implicitly given by the state equation in (26), we employ the Lagrange multipliers method to compute the sensitivities. If one aims at computing the sensitivity of a quantity  $Q = Q(\{\mathbf{x}_j^k\}_{j,k})$  with respect to  $\mu$ , then, by introducing a family of Lagrange multipliers  $\mathbf{w}_j^k$ , the Lagrangian associated to the problem reads:

$$\mathcal{L}(\mu, \nu, \{\mathbf{x}_j^k\}_{j,k}, \{\mathbf{w}_j^k\}_{j,k}) = Q(\{\mathbf{x}_j^k\}_{j,k}) - \sum_{j=1}^{N_s} \left[ \sum_{k=1}^{M_j} \mathbf{w}_j^k \cdot \left( \mathbf{x}_j^k - \mathbf{x}_j^{k-1} - \Delta t \mathbf{f}(\mathbf{x}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu) \right) + \mathbf{w}_j^0 \cdot \left( \mathbf{x}_j^0 - \mathbf{x}_0 \right) \right].$$

By setting the derivative of  $\mathcal{L}$  with respect to the variables  $\mathbf{x}_j^k$  equal to zero, it turns out that the dual variables  $\mathbf{w}_j^k$  solve the following backward difference equations, for  $j = 1, \dots, N_s$ :

$$\begin{cases} \mathbf{w}_j^{M_j} = \frac{\partial Q}{\partial \mathbf{x}_j^{M_j}} \\ \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \mu)^T \mathbf{w}_j^{k+1} + \frac{\partial Q}{\partial \mathbf{x}_j^k}, \quad \text{for } k = 0, \dots, M_j - 1. \end{cases} \quad (28)$$

Once the dual variables  $\mathbf{w}_j^k$  are available, the gradient of  $Q$  with respect to  $\mu$  reads:

$$\nabla_{\mu} Q = \frac{\partial Q}{\partial \mu} + \Delta t \sum_{j=1}^{N_s} \sum_{k=1}^{M_j} \nabla_{\mu} \mathbf{f}(\mathbf{x}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^T \mathbf{w}_j^k.$$

By setting  $Q = r^{j,m,h}$ , we have, for  $j = 1, \dots, N_s$ ,  $m = 0, \dots, M_j - 1$ :

$$\nabla_{\mu} r^{j,m,h} = \Delta t \sum_{k=1}^m \nabla_{\mu} \mathbf{f}(\mathbf{x}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^T \mathbf{w}_j^k, \quad \nabla_{\nu} r^{j,m,h} = \sqrt{\Delta t} \nabla_{\nu} \mathbf{g}(\mathbf{x}_j^m; \nu)^T \mathbf{e}_h,$$

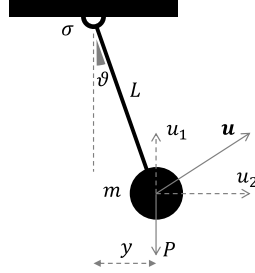
where

$$\begin{cases} \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \mu)^T \mathbf{w}_j^{k+1}, & \text{for } k = 0, \dots, m-1 \\ \mathbf{w}_j^m = \sqrt{\Delta t} \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}_j^m; \nu)^T \mathbf{e}_h. \end{cases} \quad (29)$$

The line-search algorithm for determining the step length  $\gamma^{(k)}$  also requires the computation of the gradient with respect to  $\mu$  and  $\nu$  of the discretized objective functional  $\mathcal{J} = \frac{1}{2} \mathbf{r}^T \mathbf{r}$ . If the residuals  $\mathbf{r}$  and their gradient  $\nabla \mathbf{r}$  has been already

**Table 1**  
Test cases list.

Test case	Name	Source of HF model	$N$	$N_u$	$N_y$
1	Nonlinear pendulum	System of ODEs	2	2	1
2	Nonlinear transmission line circuit	System of ODEs	1000	1	1
3	Heat equation	Parabolic PDE	3721	9	3
4	Wave equation	Hyperbolic PDE	202	2	1



**Fig. 3.** Nonlinear pendulum problem considered in Sec. 4.1.

computed, the gradient of  $\mathcal{J}$  can be recovered as  $\nabla \mathbf{r}^T \mathbf{r}$ . However, if it is not the case, it is more efficient to compute it by applying the above procedure with  $Q = \mathcal{J}$ , thus getting:

$$\begin{aligned} \nabla_{\mu} \mathcal{J} &= \Delta t \sum_{j=1}^{N_s} \sum_{k=1}^{M_j} \nabla_{\mu} \mathbf{f}(\mathbf{x}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^T \mathbf{w}_j^k, \\ \nabla_{\mathbf{v}} \mathcal{J} &= \Delta t \sum_{j=1}^{N_s} \sum_{k=0}^{M_j-1} \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}_j^k; \mathbf{v})^T (\mathbf{g}(\mathbf{x}_j^k; \mathbf{v}) - \mathbf{y}_j^k), \end{aligned}$$

where

$$\begin{cases} \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \mu)^T \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}_j^k; \mathbf{v})^T (\mathbf{g}(\mathbf{x}_j^k; \mathbf{v}) - \mathbf{y}_j^k), & \text{for } k = 0, \dots, M_j - 1 \\ \mathbf{w}_j^{M_j} = \mathbf{0}, \end{cases} \quad (30)$$

which are the discrete counterparts of (21) and (23).

#### 4. Numerical results

In this section we assess the capability of the proposed approach through four test cases, summarized in Table 1. First, in Sec. 4.1, we consider the nonlinear pendulum problem. In virtue of the modest complexity of this test case ( $N = 2$ ), we perform a sensitivity analysis on the network architecture. Moreover, we can directly compare the internal dynamics of the reduced models with that of the HF model.

Next, to test the proposed reduction approach on large-scale problems, we consider an electric circuit with nonlinear elements, described by a nonlinear system of 1000 ODEs (Sec. 4.2). Finally, we consider a parabolic PDE, whose HF model is given by its Finite Element (FE) approximation, featuring about 4000 state variables (Sec. 4.3), and an hyperbolic PDE (Sec. 4.4).

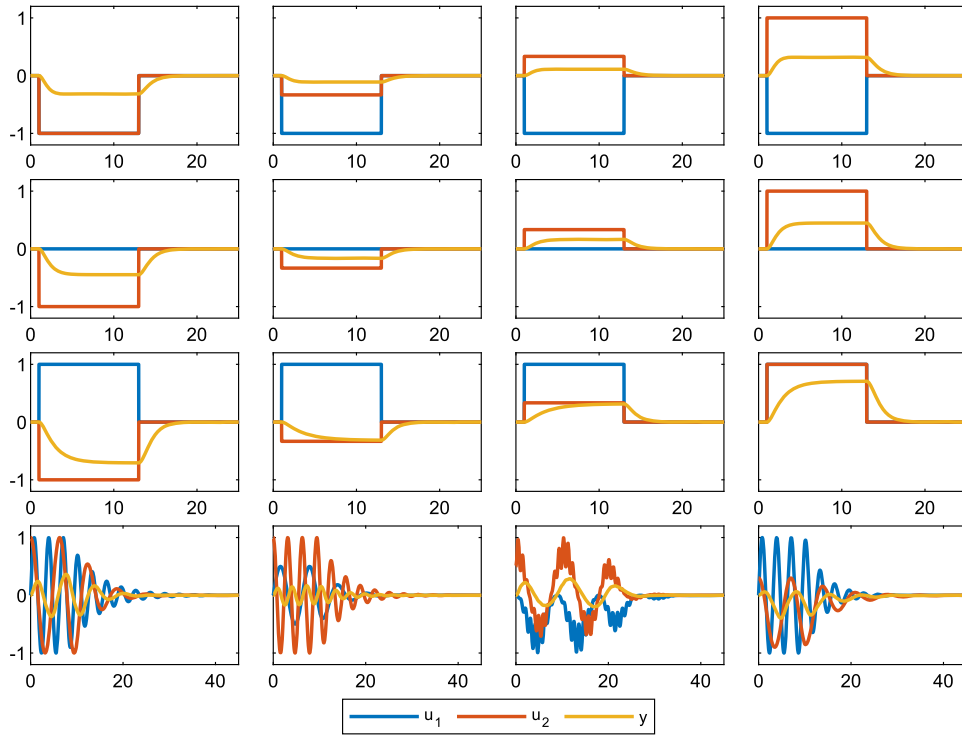
All the results shown in this paper are produced with the internally developed MATLAB library `model-learning`, publicly available in the online repository <https://github.com/FrancescoRegazzoni/model-learning>. In the same repository also the datasets accompanying this manuscript are available.

##### 4.1. Test case 1: Nonlinear pendulum

Consider a mass  $m$ , subject to its weight  $P$ , suspended by a weightless inextensible string of length  $L$  and connected to a fixed support through a hinge subject to viscous damping with constant  $\sigma$  (see Fig. 3). The pendulum, starting from its resting condition, is subject to an external force  $\mathbf{u}(t) = u_1(t)\mathbf{e}_1 + u_2(t)\mathbf{e}_2$ . By denoting the angle formed with the vertical direction by  $\vartheta(t)$ , we have that the motion of the pendulum is described by the following ODE:

$$\begin{cases} \ddot{\vartheta}(t) = \frac{1}{Lm} (u_2(t) \cos \vartheta(t) + (u_1(t) - P) \sin \vartheta(t)) - \sigma \dot{\vartheta}(t) & t \in (0, T], \\ \vartheta(0) = 0, \dot{\vartheta}(0) = 0. \end{cases} \quad (31)$$





**Fig. 4.** Test case 1: Training set. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We set the constants values to  $m = L = 1$ ,  $P = 2$ ,  $\sigma = 3$ . Moreover, we set the external force  $\mathbf{u} \in [-1, 1]^2$ . Suppose that we are interested in predicting the horizontal displacement of the mass  $y(t) = L \sin \vartheta(t)$  (in this case we drop the bold notation being the output a scalar), given the input  $\mathbf{u}(t)$ . The input-output map given by (31) fall within the concept of model introduced in Sec. 2.1, where  $N_u = 2$ ,  $N_y = 1$ ,  $U = [-1, 1]^2$ ,  $Y = [-1, 1]$ . Moreover, this model can be written in the form (1), by setting  $\mathbf{X} = (\vartheta, \dot{\vartheta})^T$  and thus we have  $N = 2$ .

For the discretization of the state equation (see Sec. 3.3), we set  $\Delta t = 5 \cdot 10^{-2}$ . Moreover, to mitigate the computational cost of the evaluation of the objective functional and its derivatives, we evaluate the error every 10 time steps. For this test, we consider the output-inside-the-state case.

#### 4.1.1. Training, validation and test sets

The optimization of the ANN described in Sec. 3.4 is led by the set of experiments (6) collected from the HF model. The choice of such *training set* is crucial since it is expected to be representative of all the possible working regimes of the system. Moreover, it should be large enough to avoid overfitting of the reduced model on the training set itself.

In this first example, we intentionally consider a poor training set, such that overfitting is likely to occur in order to better investigate the sensitivity of the proposed reduction approach with respect to  $n$  (i.e. the number of states) and to the complexity of the network. The training set is represented in Fig. 4. It comprises the step responses associated to different stationary values of the input, in the form  $\mathbf{u}(t) = \bar{\mathbf{u}} \mathbb{1}_{[t_1, t_2]}(t)$  (notice that  $\bar{\mathbf{u}}$  span all the set  $U = [-1, 1]^2$ ) and four samples with oscillating input, obtained by sinusoids with different frequency, amplitude and mean value. The frequency range of the signals is chosen in such a way that it covers several characteristic time scales of the considered HF model.

To monitor the ANN learning process, at each optimization epoch, we evaluate the performance of the ANN on a further set, that we call *validation set*. The comparison between the relative  $L^2$  error on the training set (which we denote by  $\mathcal{E}_{\text{train}}$ ) and the error on the validation set (which we denote by  $\mathcal{E}_{\text{val}}$ ) allows to perform *regularization by early stopping* (see e.g. [64]): as long as the error on the validation set start increasing, we stop the optimization loop since this is an indication that overfitting occurs. In the validation set we place 5 samples, comprising several working regimes of the system (see Fig. 5). Notice that in this set we also switch from a regime to another inside the same sample, to monitor the capability of the model to cope with it.

Finally, when the optimization loop is completed, we test the performance of the reduced model on a large-size *test set*, comprising step responses, oscillatory inputs and randomly generated ones. The test set amounts to 126 samples.

#### 4.1.2. Sensitivity analysis of the network complexity

The objective functional  $\mathcal{J}$  is non-convex and features several many local minima. As a consequence, the optimized ANN obtained by means of the strategy of Sec. 3.4 may depend on the initialization of the ANN itself. Since we adopt a random

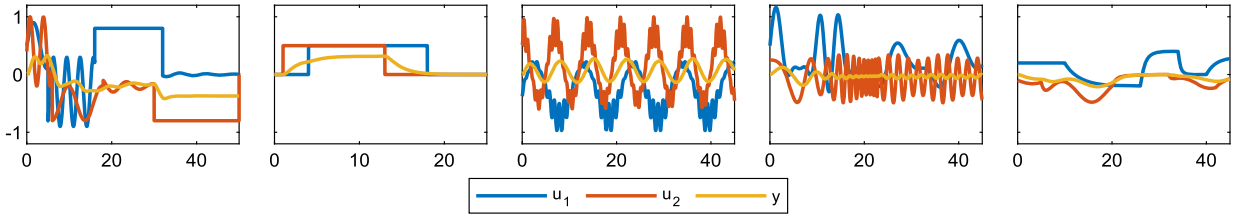
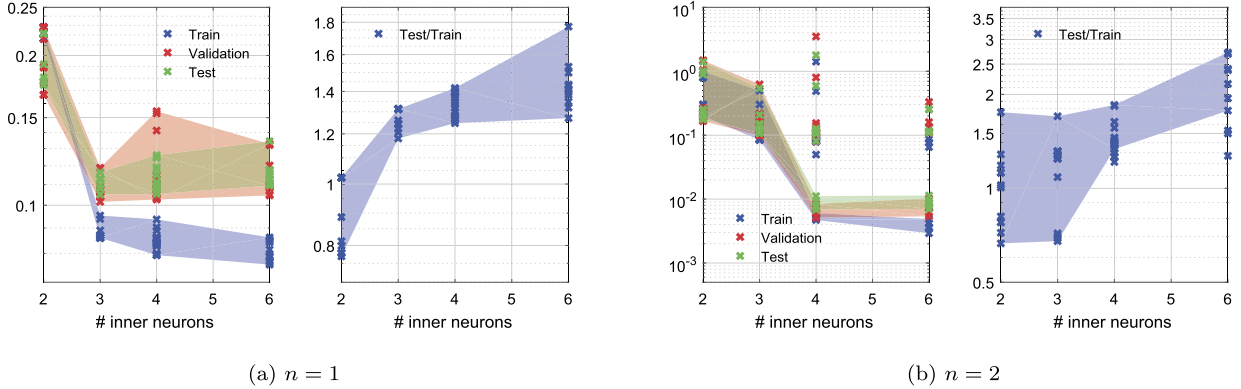


Fig. 5. Test case 1: Validation set.



**Fig. 6.** Test case 1: errors vs number of hidden neurons  $N_{\text{neurons}}$ , in the cases (a)  $n = 1$  and (b)  $n = 2$ . For both cases, the left plot shows the errors  $\mathcal{E}_{\text{train}}$ ,  $\mathcal{E}_{\text{val}}$  and  $\mathcal{E}_{\text{test}}$ , while the right plot shows the ratio  $\mathcal{E}_{\text{test}}/\mathcal{E}_{\text{train}}$ . Each cross represents the final result of a test (including when the tests gets stuck into a “bad” local minimum). Colored regions represent the areas spanned by the tests, excluding the tests which got stuck into a “bad” local minimum.

initialization for the ANN, by running the proposed strategy several times with the same ANN architecture, we actually end up with very different results. This is intimately linked to the non-uniqueness of representation of a given model (see Remark 1); very different ANNs may represent the same model. We will go back to this issue in Sec. 4.1.3.

In the following we perform a sensitivity analysis of the performance of the proposed reduction approach w.r.t. the complexity of the network, i.e. the number of hidden neurons, which we denote by  $N_{\text{neurons}}$ . For simplicity, we consider only the case of ANNs with a single hidden layer. Fig. 6 shows the dependency of the performance of the proposed reduction approach on the number of hidden neurons, in the cases  $n = 1$  and  $n = 2$ . Each cross is associated to a single test, while the colored regions highlight the areas spanned by the test which did not get stuck into a “bad” local minimum (we tag a local minimum as “bad” if the error  $\mathcal{E}_{\text{train}}$  is more than 10 times the best error obtained with the same ANN architecture).

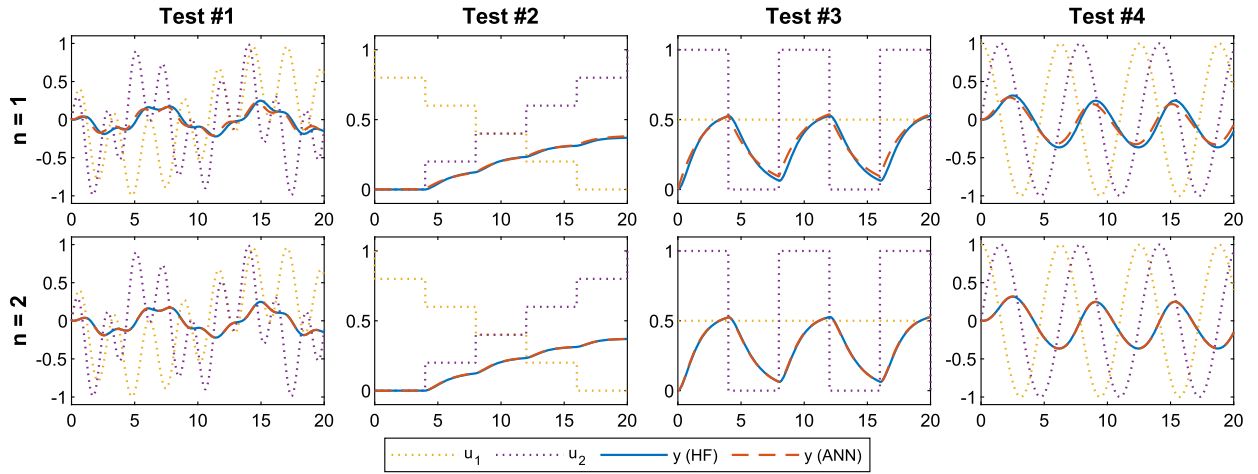
Let us consider first the case  $n = 1$ . By switching from  $N_{\text{neurons}} = 2$  to  $N_{\text{neurons}} = 3$ , thanks to the enhanced representative capacity of the ANN, the errors associated with the three sets significantly decrease (see Fig. 6a). However, by further increasing  $N_{\text{neurons}}$ , we are in presence of overfitting: even if  $\mathcal{E}_{\text{train}}$  keeps decreasing,  $\mathcal{E}_{\text{test}}$  and  $\mathcal{E}_{\text{val}}$  start increasing. This phenomenon is also evident from the right figure, showing the ratio  $\mathcal{E}_{\text{test}}/\mathcal{E}_{\text{train}}$  increasing with the network complexity. Even if the issue of designing the training set falls beyond the purposes of the present work, we notice that with a richer training set the optimal number of neurons would probably increase, thus allowing to reach a better performance of the reduced model.

We then consider the case  $n = 2$  (see Fig. 6b). The introduction of a further state variable in the system, w.r.t. the case  $n = 1$ , translates in a significant increment of the model ability to faithfully reproduce the input-output map given by the HF model, as expected according to Proposition 2. Indeed, even if for  $N_{\text{neurons}} \leq 3$  the errors are similar to the case  $n = 1$ , by increasing the network complexity the errors drop by one order of magnitude for  $N_{\text{neurons}} = 4$ , before slowly diverging (the training error  $\mathcal{E}_{\text{train}}$  decreases, while the validation  $\mathcal{E}_{\text{val}}$  and test error  $\mathcal{E}_{\text{test}}$  slightly increase). We also notice the occurrence of tests which got stuck in “bad” local minima (highlighted in Fig. 6b by the presence of crosses outside the colored areas) for  $N_{\text{neurons}} \geq 4$ , due to the raising of complexity of the landscape of  $\mathcal{E}_{\text{train}}$  when the dimensionality of the design space increases.

To sum up, we select as best networks the ones minimizing the error on the test set for  $N_{\text{neurons}} = 3$  in the case  $n = 1$  and for  $N_{\text{neurons}} = 4$  in the case  $n = 2$ . Fig. 7 shows the comparison of the results obtained in four test cases with the HF and the two selected reduced models (the first row refers to the model with  $n = 1$ , the second to the model obtained with  $n = 2$ ).

#### 4.1.3. What did the ANN actually learn?

The feasibility of the reduction approach proposed in this work is strictly related to the possibility of faithfully representing the state of the HF model (which we denote by  $\mathbf{X}(t) \in \mathbb{R}^N$ ) by means of a lower-dimensional state  $\mathbf{x}(t) \in \mathbb{R}^n$ . If this



**Fig. 7.** Test case 1: comparison of the exact solution (blue line) and the solution obtained with the ANN model (red line) in four different test cases. First row: one-variable model; Second column: two-variables model.

is the case, then the knowledge of  $\mathbf{x}(t)$  and  $\mathbf{u}(t)$  provides enough information to compute, with just a little approximation, both the output  $\mathbf{y}(t)$  and the evolution of the state of the system (that is to say the value of  $\frac{d}{dt}\mathbf{x}(t)$ ).

We may thus interpret the learning process as that of implicitly building a map from the reduced-order state  $\mathbf{x}$  to the full-order state  $\mathbf{X}$ , which is not directly observed, but it is observed only through its effects to the time evolution of  $\mathbf{y}(t)$ . This is somehow similar to projection-based MOR strategies (see Sec. 1.1), with the important differences that: (i) in this case the map is not explicitly built and thus is not available; (ii) while in projection-based methods the map is linear, here nonlinear mappings can be exploited; (iii) since the effects of the internal state  $\mathbf{X}$  are seen just through  $\mathbf{y}$ , only the features which are relevant for the input-output map are exploited, whereas POD for example may extract features from the snapshots which do not provide a significant contribution to the input-output map.

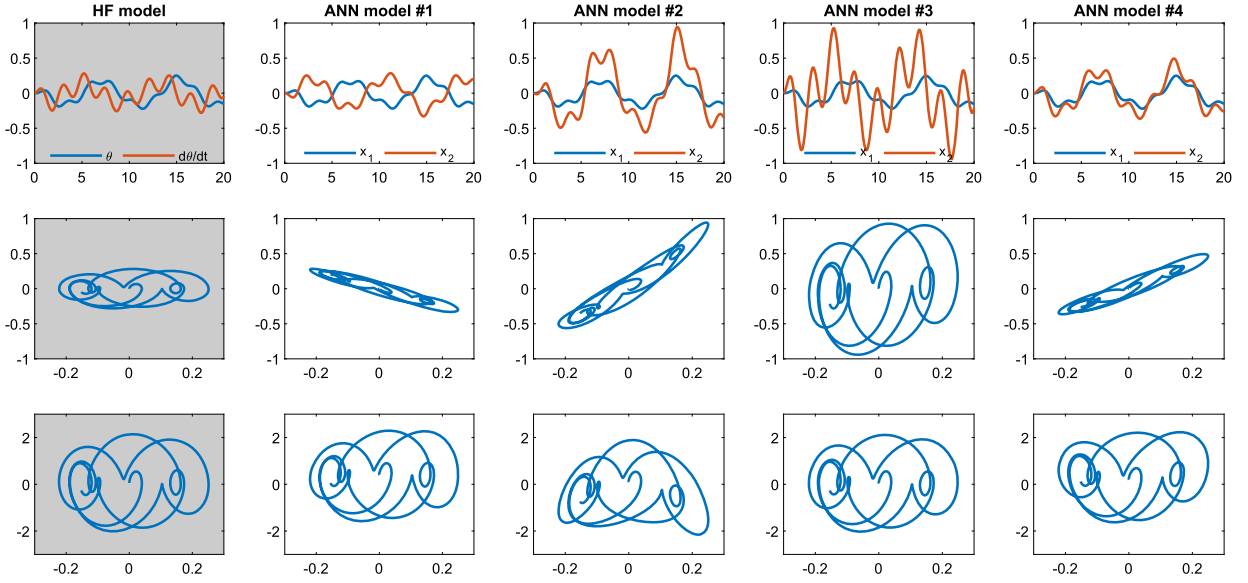
As mentioned before, the reduced model implicitly defines a map from the HF model state  $\mathbf{X}(t)$  to the reduced model state  $\mathbf{x}(t)$ . This map is not explicitly built. Moreover, as noticed in Remark 1, a model described by an ODE is invariant with respect to change of variables for its internal state  $\tilde{\mathbf{x}} = \mathbf{h}(\mathbf{x})$ . For these reasons, it is usually troublesome to give a physical interpretation to the state variables of the reduced model.

Let us consider the example addressed in this Section, for  $n = N = 2$ . This case may not be interesting in a MOR perspective, but it helps to shed light on the machinery behind the proposed reduction approach. Indeed, thanks to the choice  $\mathbf{g}(\mathbf{x}) = \pi^{N_y}$ , we have, in principle,  $x_1(t) = L \sin \vartheta(t)$ . Consequently, we may speculate that the knowledge of the first variable of the reduced model  $x_1$  provides all the information needed to reconstruct the first variable of the HF model  $\vartheta$ . Therefore, we expect that the second variable of the reduced model  $x_2$  provides the missing information to reconstruct  $\vartheta$ . This does not necessarily entail that  $\vartheta$  is a function of  $x_2$ , but in general we have that  $\vartheta$  is a function of *both*  $x_1$  and  $x_2$ . If this is true, then when a trajectory in the  $(\vartheta, \dot{\vartheta})$  plane crosses, then the same should happen for the trajectory in the  $(x_1, x_2)$  plane. In other words, all the trajectories of the reduced models, corresponding to the a given input  $\mathbf{u}$ , should be homotopic to the corresponding trajectory of the HF model (i.e. they can be obtained by a continuous deformation of the HF trajectory).

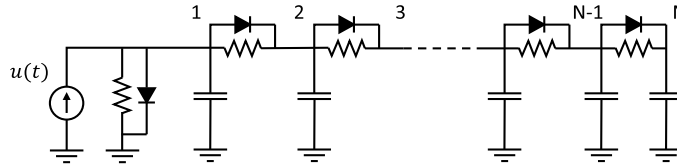
To verify these conjectures, we consider the oscillating input also employed in the first column of Fig. 7 (i.e.  $u_1 = \sin(t)\cos(1.3t)$ ,  $u_2 = \cos(1.8t)\sin(t)$ ) and we compare the trajectories of the HF model (grey background in Fig. 8) with those of four different reduced models (white background in Fig. 8), obtained with four different random initializations of the ANN. In the first line of Fig. 8 we show the time evolution of the two state variables of the systems. By looking at the figure it is apparent how different the obtained ANN models can be. However, by observing – in the second row of Fig. 8 – the trajectories in the phase space  $(x_1, x_2)$ , one can see that all the trajectories are approximatively homotopic to the trajectory of the HF model in its phase space  $(\vartheta, \dot{\vartheta})$ . Finally, in order to better visualize the role of the second variable, we decouple it from  $x_1$  in the following way. First, we collect the values of the two variables at the discrete times  $t_0, t_1, \dots$  in two vectors, which we denote by  $\mathbf{z}_1$  and  $\mathbf{z}_2$ , respectively. Next, we center the vectors ( $\mathbf{z}'_j = \mathbf{z}_j - \bar{\mathbf{z}}$ , for  $j = 1, 2$ , where  $\bar{\mathbf{z}}$  denotes the mean value of  $\mathbf{z}$ ). Next, we subtract to  $\mathbf{z}'_2$  the component parallel to  $\mathbf{z}'_1$  (that is to say  $\mathbf{z}''_2 = \mathbf{z}'_2 - (\mathbf{z}'_2 \cdot \mathbf{z}'_1) / \|\mathbf{z}'_1\|^2 \mathbf{z}'_1$ ). Finally, we normalize  $\mathbf{z}''_2$  (that is to say  $\mathbf{z}'''_2 = \mathbf{z}''_2 / \|\mathbf{z}''_2\|$ ). The third row of Fig. 8 shows the trajectories in the plane  $(x_1, x_2''')$ . For the original model, we show the trajectory in the plane  $(\vartheta, \dot{\vartheta}''')$ , where the coordinate  $\dot{\vartheta}'''$  is computed by the same procedure as  $x_2'''$ .

#### 4.2. Test case 2: Nonlinear transmission line circuit

In order to assess the capability of the proposed reduction approach to reduce the complexity of large-scale nonlinear systems, we consider a popular benchmark in MOR, namely the nonlinear transmission line circuit represented in Fig. 9 (see



**Fig. 8.** Test case 1: results of the test case considered in Sec. 4.1.3. Each column refers to a different model (first column: HF model; other columns: four different ANN models). First line: time evolution of the two state variables of the system (first column:  $\vartheta$  and  $\dot{\vartheta}$ ; other columns:  $x_1$  and  $x_2$ ). Second line: trajectories in the phase space (first column:  $(\vartheta, \dot{\vartheta})$ ; other columns:  $(x_1, x_2)$ ). Third line: trajectories in the  $(\vartheta, \dot{\vartheta}''')$  phase space (first column) and  $(x_1, x_2''')$  phase space (other columns).



**Fig. 9.** Test case 2: scheme of the nonlinear transmission line electric circuit considered in Sec. 4.2.

e.g. [71,72]). It is an electrical network comprising a current source,  $N = 1000$  unitary resistors,  $N$  unitary capacitors and  $N$  nonlinear diodes with law  $i = e^{40v} - 1$ . The input of the model is the current source  $u(t)$ , taking values in  $U = [0, 1]$ , and the output is the tension at the first node  $v_1(t)$ . By denoting by  $v_i$  the voltage at the  $i$ -th node, the HF model reads as follows:

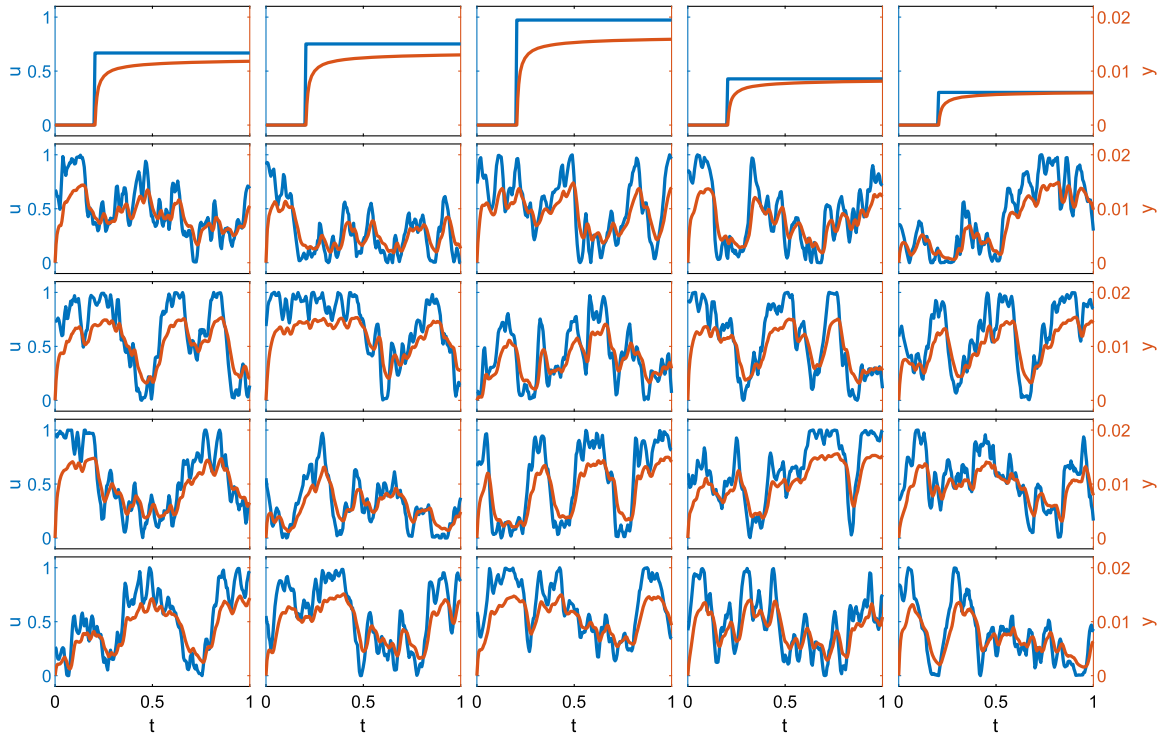
$$\begin{cases} \dot{v}_1(t) = -2v_1(t) + v_2(t) + 2 - e^{40v_1(t)} - e^{40(v_1(t)-v_2(t))} + u(t) \\ \dot{v}_i(t) = -2v_i(t) + v_{i-1}(t) + v_{i+1}(t) + e^{40(v_{i-1}(t)-v_i(t))} - e^{40(v_i(t)-v_{i+1}(t))}, & \text{for } i = 2, \dots, N-1 \\ \dot{v}_N(t) = -v_N(t) + v_{N-1}(t) - 1 + e^{40(v_{N-1}(t)-v_N(t))}, \end{cases} \quad (32)$$

supported by the initial condition  $v_1(0) = v_2(0) = \dots = v_N(0) = 0$ . Notice that the HF model is written in the form of Eq. (1), for  $\mathbf{X} = (v_1, \dots, v_N)^T$  and  $\mathbf{F}: \mathbb{R}^N \times [0, 1] \rightarrow \mathbb{R}^N$ .

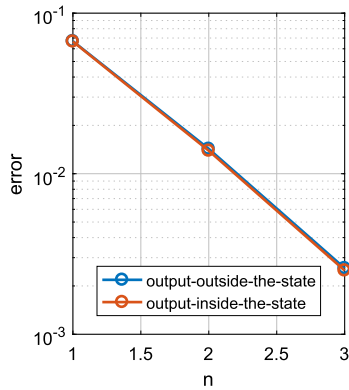
We consider the training set represented in Fig. 10, comprising 5 step inputs and 20 randomly generated input signals, each of 1 s duration. In this test case we consider both the output-inside-the-state and the output-outside-the-state cases, for  $n = 1, 2, 3$  and we compare the results. In all the cases we employ ANNs with a single hidden layer, with respectively 8 and 3 neurons in the ANN for  $\mathbf{f}$  and  $\mathbf{g}$ . For the time discretization, we employ a time step of  $\Delta t = 5 \cdot 10^{-3}$ .

As mentioned in Sec. 2.4.2, in the case  $n \geq N_y$ , models in the form (2) can be possibly rewritten in the form (17). Even if this is not always valid, we may expect that with the constraint  $\mathbf{g} = \boldsymbol{\pi}^{N_y}$  the capacity of the class of models to approximate the HF model is not substantially reduced. Therefore we expect the output-inside-the-state case to provide similar results to the ones obtained with the output-outside-the-state approach.

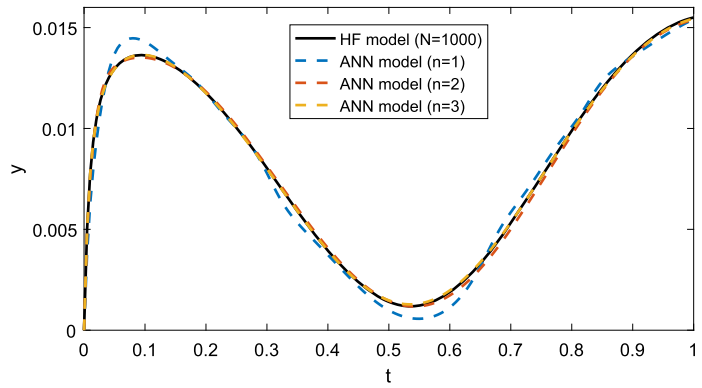
In Fig. 11a we show  $\mathcal{E}_{\text{test}}$ , the relative  $L^2$  error on the test set (which comprises 25 step responses and 80 randomly generated inputs), for the different cases considered. We notice that, as expected, the two approaches provide, for a given value of  $n$ , very similar results. The output-inside-the-state case is thus preferable since it is more efficient both in the offline phase (since the number of design variables is lower) and in the online phase (since  $\mathbf{g}$  does not need to be evaluated at each time step). We also notice that, coherently with Proposition 2, the error  $\mathcal{E}_{\text{test}}$  decreases as  $n$  increases, reaching, for  $n = 3$ , a remarkably good approximation level (nearly  $2.5 \cdot 10^{-3}$ ). In Fig. 11b we compare the response of the HF model with that of the three reduced models obtained with the output-inside-the-state case in the time-domain.



**Fig. 10.** Test case 2: training set. Blue lines (axis on the left) represent the time evolution of the input, red lines (axis on the right) represent the time evolution of the output.



(a)  $\mathcal{E}_{\text{test}}$  vs  $n$



(b) Time domain responses to the input  $\mathbf{u}(t) = \frac{1}{2}(1 + \cos(2\pi t))$ : comparison between the HF model and three ANN models.

**Fig. 11.** Test case 2: model errors for  $n = 1, 2, 3$  with both output-inside-the-state and output-outside-the-state cases (a) and time-domain model responses (b).

#### 4.3. Test case 3: Heat equation (parabolic PDE)

We consider now the application of the proposed reduction approach to the MOR of a parabolic PDE problem, which extends the benchmark problem considered in [73], where its Reduced Basis (RB) reduction has been considered. Consider the spatial domain  $\Omega = (0, 1.5)^2$ , whose boundary  $\partial\Omega$  is partitioned into the top border  $\Gamma_t$ , in contact with a heat reservoir with zero temperature, the bottom border  $\Gamma_b$ , with a constant inward heat flux  $\varphi = 1$ , and the wall borders  $\Gamma_w$ , characterized by no-flux boundary conditions (see Fig. 12). The time evolution of the spatially distributed temperature  $\psi(\mathbf{x}, t)$  in the domain  $\Omega$  is thus described by the heat equation:

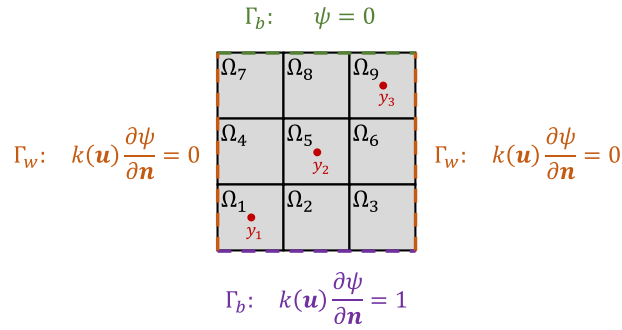


Fig. 12. Test case 3: domain and boundary conditions.

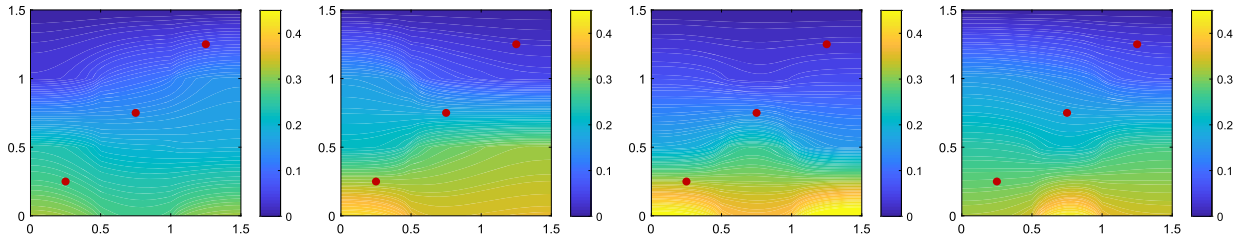


Fig. 13. Test case 3: 4 examples of snapshots obtained at different times with different inputs.

$$\begin{cases} \frac{\partial}{\partial t} \psi(\mathbf{x}, t) - \operatorname{div}(k(\mathbf{x}, \mathbf{u}(t)) \nabla \psi(\mathbf{x}, t)) = 0 & \text{in } \Omega, \text{ for } t > 0 \\ k(\mathbf{x}, \mathbf{u}(t)) \nabla \psi(\mathbf{x}, t) \cdot \mathbf{n} = 0 & \text{on } \Gamma_w, \text{ for } t > 0 \\ k(\mathbf{x}, \mathbf{u}(t)) \nabla \psi(\mathbf{x}, t) \cdot \mathbf{n} = \varphi & \text{on } \Gamma_b, \text{ for } t > 0 \\ \psi(\mathbf{x}, t) = 0 & \text{on } \Gamma_t, \text{ for } t > 0 \\ \psi(\mathbf{x}, 0) = 0 & \text{on } \Omega. \end{cases} \quad (33)$$

Let us partition the domain  $\Omega$  into 9 subdomains  $\Omega_i$  of equal size, for  $i = 1, \dots, 9$ , as in Fig. 12. Let us consider the piecewise constant thermal conductivity coefficient  $k$ , parametrized by the 9-dimensional input  $\mathbf{u}(t) \in [10, 100]^9$ , defined as follows:

$$k(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^9 u_i \mathbb{1}_{\Omega_i}(\mathbf{x}),$$

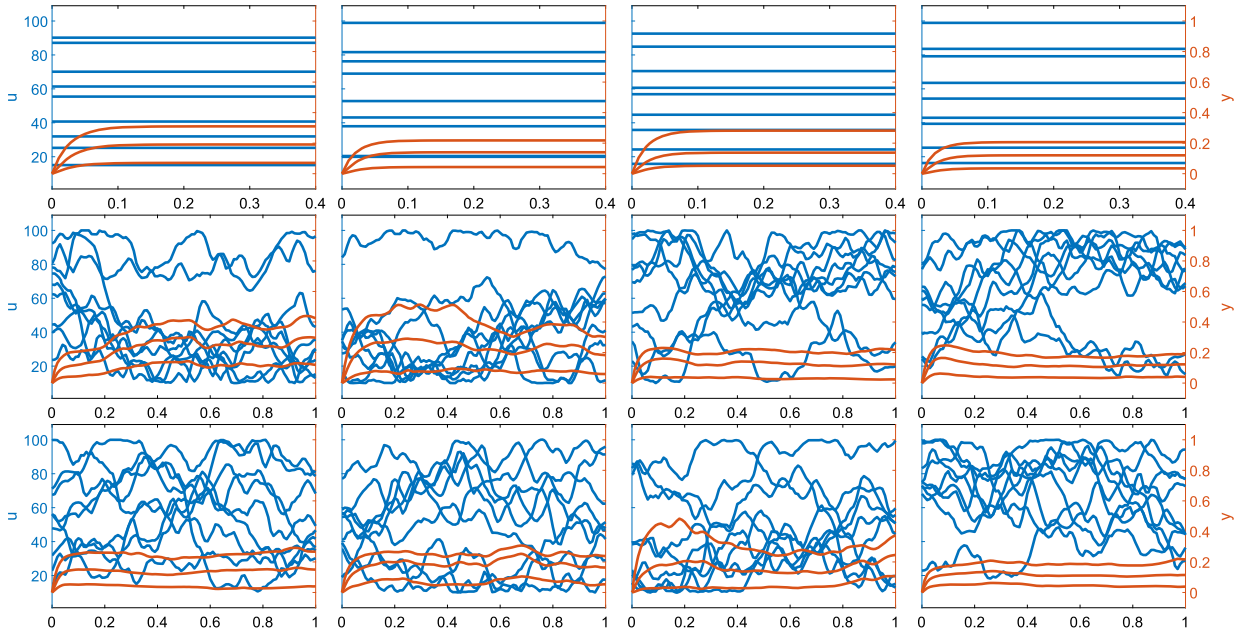
where  $\mathbb{1}_{\Omega_i}$  is the indicator function of  $\Omega_i$ . Consider then three probes, located at the center of the subdomains  $\Omega_1$ ,  $\Omega_5$  and  $\Omega_9$ , measuring the time evolution of the temperature in such points. The output  $\mathbf{y}(t) \in \mathbb{R}^3$  is the vector collecting the three temperature values.

For the HF solution of (33), we consider the P2 Finite Element approximation on a 30 by 30 uniform square elements grid and we employ the backward Euler scheme, with  $\Delta t = 10^{-2}$ , for the time discretization, implemented in the MATLAB finite element library `feamat` ([74]). The HF model has dimension  $N = 3721$ . In Fig. 13 we report some snapshots of the solution obtained at different times for different inputs.

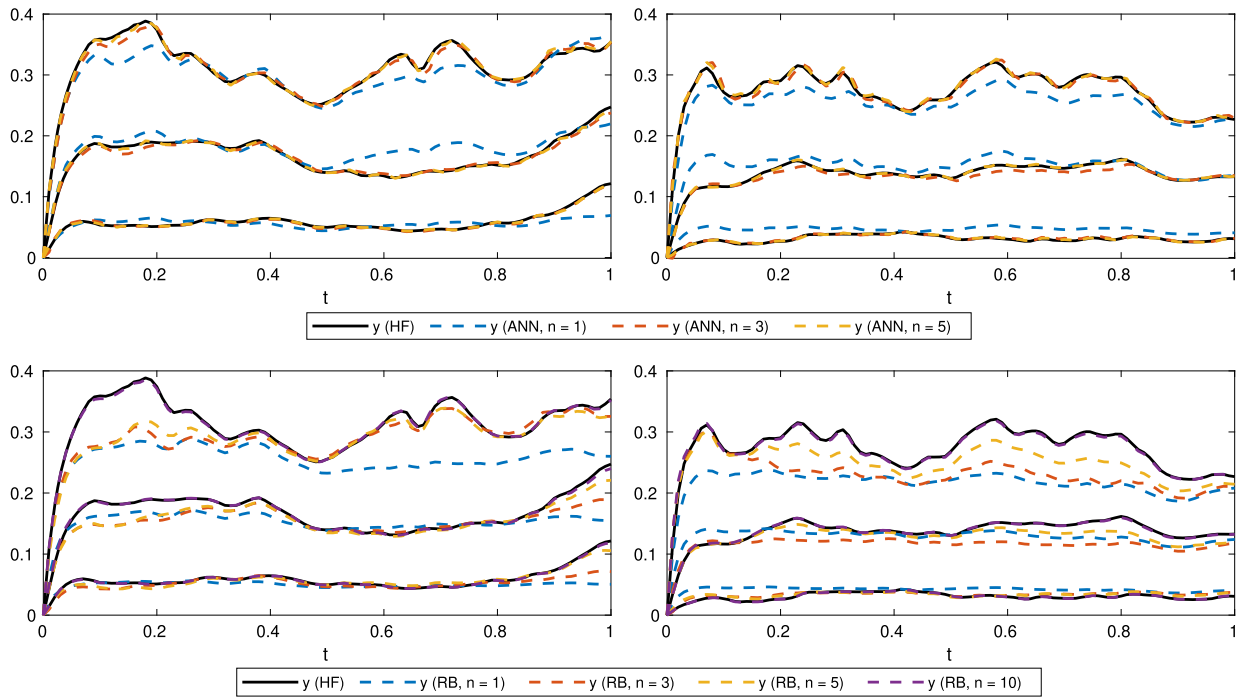
In this test case we compare the results obtained with the proposed reduction approach with those obtained with a popular MOR method in the field of PDEs, namely the RB method, which exploits the linearity of Eq. (33) and the affine dependence on  $\mathbf{u}$  ([4]). For the ANN-based reduction method we consider  $n$  ranging between 1 and 5, by employing both the output-outside-the-state approach and, for  $n \geq N_y = 3$ , also the output-inside-the-state approach. In each case we use single hidden layer ANNs, with 12 to 16 hidden neurons for  $\mathbf{f}$  and 3 or 4 hidden neurons for  $\mathbf{g}$ , by increasing the number of neurons as  $n$  increases. Fig. 14 reports a subset of the training set, comprising steady-state responses lasting for 0.4 s, obtained by sampling the input space  $U = [10, 100]^9$  by means of Latin Hypercube Sampling (see [75]), and twice as much random transients of duration 1 s. This choice aims at covering different working regimes of the HF model, including steady-state and transient regimes. We increase the number of training samples from 150 to 480 while increasing  $n$  to reflect the increased capacity of the ANN due to the increased number of neurons. As of the RB method, we build the basis by POD of the snapshot matrix obtained by sampling every 0.1 s the biggest set used to train the ANNs. In both cases we employ the same time step used for the HF model, i.e.  $\Delta t = 10^{-2}$ .

In Fig. 16 we compare the results obtained by the two methods, by evaluating the error on a test set composed by 20 steady-state inputs, 50 random input signals of duration 1 s (two examples are reported in Fig. 15) and 10 random input signals of duration 10 s. The purpose of the latter choice is to assess the capability of the proposed reduction approach



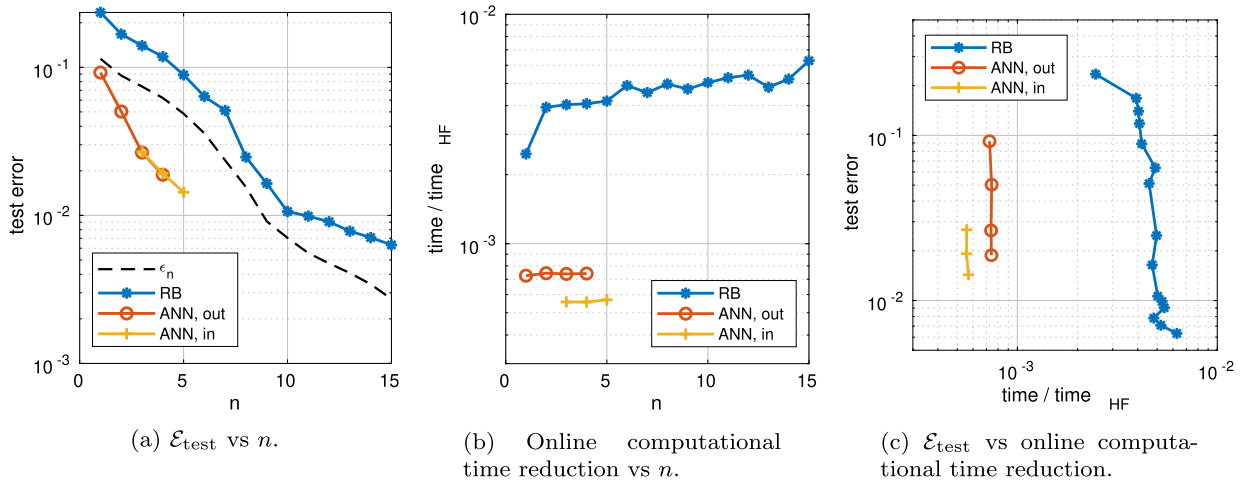


**Fig. 14.** Test case 3: a subset of the training set. Blue lines (axis on the left) show the time evolution of the 9 inputs (i.e. the thermal conductivities of the 9 subdomains), red lines (axis on the right) show the time evolution of the three outputs (i.e. the temperature at the three probes).



**Fig. 15.** Test case 3: time-domain comparison of the result of the HF model with those of the different reduced models for two random tests. Top: ANN models vs HF model; bottom: RB models vs HF model.

to approximate the evolution of the HF model also for longer time horizons than the ones used in the training phase. As shown in Fig. 16a, for a given model size  $n$ , the ANN based reduced model performs better than the RB one: for  $n = 3$ , e.g., the corresponding error is almost one order of magnitude smaller than the RB error and to reach the same approximation level with the RB method we need at least  $n = 8$ . Moreover, the reduction in terms of online computational time is larger with the proposed approach too.



**Fig. 16.** Test case 3: comparison between the proposed reduction approach and the RB method. (a) Relative  $L^2$  error versus number of variables of the reduced model  $n$ ; (b) Mean computational time required by the reduced model, normalized w.r.t. that required by the HF model to simulate the same amount of physical time, versus  $n$ ; (c) Relative  $L^2$  error versus normalized computational time. In the legend, “in” refers to output-inside-the-state, while “out” refers to output-outside-the-state.

In Fig. 16a we plot, together with the errors, the quantity  $\epsilon_n$ , defined as

$$\epsilon_n = \sqrt{\frac{\sum_{i=n+1}^S \sigma_i^2}{\sum_{i=1}^S \sigma_i^2}}$$

where  $\sigma_1, \dots, \sigma_S$  are the singular values of the snapshots matrix. The quantity  $\epsilon_n$  represents the fraction of energy of the snapshots that is not captured by the first  $n$  modes and can be used to estimate the “reducibility” of the state space in terms of an  $n$ -dimensional subspace (see [4]).

The major drawback of the proposed reduction approach lies in the offline phase, which requires the training of one or two ANNs (for  $\mathbf{f}$  and  $\mathbf{g}$ ), while for the RB method we just need to compute the SVD decomposition of the snapshot matrix and to project both the model matrices and right-hand side. Moreover, the computational complexity of the training rapidly grows with  $n$  and, for high  $n$ , large training sets are needed to avoid overfitting, thus preventing the applicability of the proposed approach to large  $n$ . For this reason, we limit ourselves to the cases  $n \leq 5$ . However, from the presented results it seems that with the proposed approach it is possible to get a good approximation of complex models (like the PDE model considered in this section) just with a few variables, so we do not actually need to increase  $n$  if the approximation level is satisfactory for the application. We also notice that in this work we employed very basic tools to train the ANNs, while the offline phase may be considerably speeded up with the application of more advanced techniques (see Sec. 5) which, besides decreasing the training time also minimize overfitting, thus reducing the number of samples needed to train the ANN, hence making it possible to train ANN models with larger values of  $n$ .

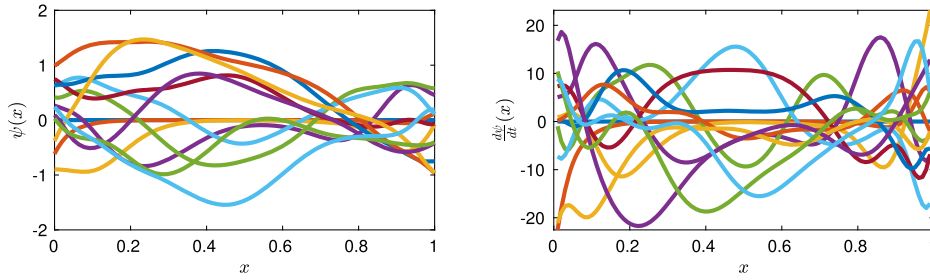
Finally, we notice that this comparison has been carried out in the most favorable case for RB, namely linear models with affine input dependence and with linear state-output dependence, while the proposed ANN-based reduction approach does not exploit any of those structural characteristic of the HF model. Therefore, in the nonlinear case or with non affine input dependence, while the RB method requires techniques such as EIM and DEIM (see Sec. 1.1), which reduce the performances of the method, the proposed reduction approach can be applied without modifications (see e.g. the Test case 2).

#### 4.4. Test case 4: Wave equation (hyperbolic PDE)

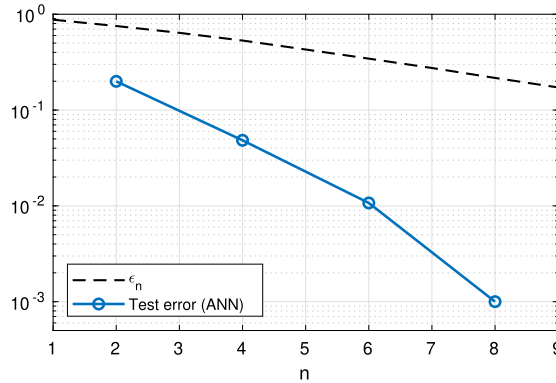
We now consider an hyperbolic PDE, namely the wave equation in the one-dimensional domain  $(0, L)$ , with  $L = 1$ :

$$\begin{cases} \frac{\partial^2}{\partial t^2} \psi(x, t) - c^2 \frac{\partial^2}{\partial x^2} \psi(x, t) = 0 & \text{for } x \in (0, L), t > 0 \\ \psi(0, t) = u_1(t) & \text{for } t > 0 \\ \psi(L, t) = u_2(t) & \text{for } t > 0 \\ \psi(x, 0) = 0 & \text{for } x \in (0, L) \\ \frac{\partial}{\partial t} \psi(x, 0) = 0 & \text{for } x \in (0, L), \end{cases} \quad (34)$$

where we set  $c = 2$ . The 2-dimensional input of the model consists in the Dirichlet boundary conditions at the two ends of the domain, while the output is defined as the value of the solution in the mid-point of the domain  $y(t) = \psi(L/2, t)$ .



**Fig. 17.** Test case 4: example of snapshots of the state  $\psi(t)$  and its time derivative  $\frac{\partial \psi}{\partial t}(x)$  at different time steps for a sample belonging to the training set.



**Fig. 18.** Test case 4:  $\mathcal{E}_{\text{test}}$  vs. number of variables of the reduced model  $n$  (solid line) and  $\varepsilon_n$  vs. number of modes (dashed line).

For the HF solution we use implicit second order BDF in time and centered second order finite differences in space, with a space discretization step of  $h = 10^{-2}$  and a time discretization step of  $\Delta t = 10^{-2}$ . In Fig. 17 we show a solution of the HF model for a random input at different time-steps.

In order to apply the proposed method, we rewrite Eq. (34) as a first order (in time) system of PDEs. To train the ANN we consider a training set consisting of 200 random inputs and we employ the same time step used to generate data. In Fig. 19 we compare the HF output with the output of the different learned models for 4 different tests. Fig. 18 shows the test error of the learned models for different values of  $n$ , by using the output-inside-the-state approach and by using a single hidden layer with  $N_{\text{neurons}} = n + 2$ . Also in this test case, the error decreases with  $n$  and, noticeably, it decreases faster than  $\varepsilon_n$ .

## 5. Conclusions and perspectives

We have proposed a data-driven nonlinear MOR technique, based on ANNs. We formulated the model reduction problem as a best-approximation or maximum-likelihood problem, where we look for the most suitable representation of the HF (high-fidelity) dynamical model into a class of simpler models. The latter consists in a class of ODE models described by means of an ANN (or by two of them, in the output-outside-the-state case), which is fed by input-output pairs originated from the HF system. Thanks to this formulation, it is possible to compute the sensitivity of the model error with respect to the parameters of the ANNs, and to exploit standard optimization techniques to make the ANN learn the underlying physics of the system. The proposed reduction approach can be applied to a wide class of dynamical models with time dependent inputs, subject to some minimal requirements (see Sec. 2.1).

We have shown that the class of ANN models used in this paper can approximate within any desired accuracy any model described by a system of ODEs. Moreover, the same result holds by replacing the class of ANNs with any class of functions that can approximate continuous functions on compact sets with arbitrarily small error.

The proposed technique can be flexibly applied for different purposes: (i) building a surrogate of a computationally expensive model, which allows for fast evaluations and which can be used for multi-query purposes; (ii) reducing the state dimension of a time-dependent model; (iii) learning a mathematical model starting from input-output pairs (see Fig. 1).

We have assessed the effectiveness of the proposed approach on a simple case study (namely the nonlinear pendulum) and by investigating the reduction of two large-scale problems (a nonlinear system of ODEs and a parabolic PDE), featuring thousands of degrees of freedom, and an hyperbolic PDE. In all the cases we derived reduced models capable of approximating the HF models with an error of order  $10^{-2} \div 10^{-3}$ , with just a few state variables. We have also compared the performance of the proposed reduction approach with that of the RB method, one of the most popular MOR methods in the field of PDEs. For a given reduced model size, despite a more expensive offline phase, the proposed reduction ap-

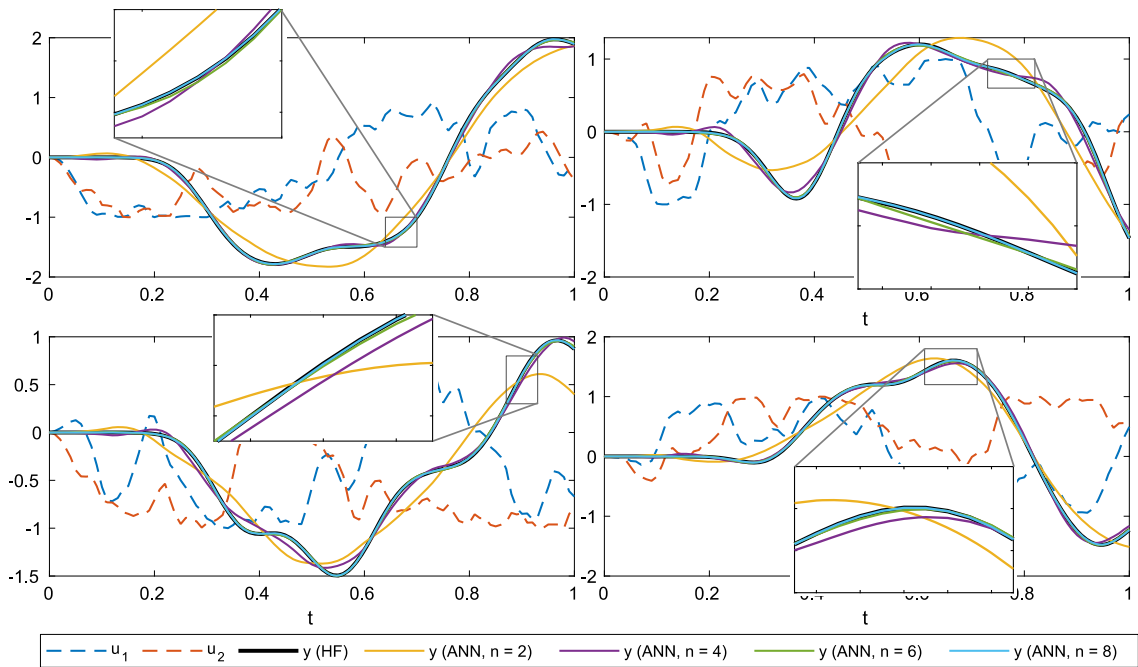


Fig. 19. Test case 4: comparison between the HF solution and the solution with the 4 trained ANN models in 4 different test cases.

proach yielded much more accurate reduced models than the RB method, also featuring a lower online cost in terms of computational time.

Some crucial aspects, where there is possibility for improvement of the proposed reduction approach, have not been fully explored in the present study. In particular, we have not dealt with the issue of the training set design, where it is possible to employ automatic procedures to select an optimized training set, in a similar manner to the selection of snapshots in the RB framework (see e.g. [4]). Moreover, globalization techniques can be taken into account to deal with the problem of local minima in the optimization process. Furthermore, different optimization algorithms can be considered in place of the Levenberg-Marquardt algorithm (whose computational cost becomes prohibitive when the number of parameters to be trained becomes large), such as Stochastic Gradient Descent [76], AdaGrad [77] or Adam [78], which are widely used in Machine Learning. The offline phase may be considerably speeded up also with the application of more advanced learning techniques than the one considered in this work, such as stochastic selection of the training set (or mini-batch training, see e.g. [76]), dropout ([79]), batch normalization ([80]) and progressive layers freezing ([81]). This would allow to speed up the training phase, thus allowing to further investigate the potential of the proposed method by considering larger values of  $n$ .

In this work we have performed hyperparameters tuning by a trial-and-error approach. In future developments we will consider ad-hoc algorithms, such as evolutionary optimization, bayesian optimization, grid search and k-means.

Finally, we notice that the formulation of the MOR problem in terms of minimization problem potentially allows to easily incorporate into the learning stage some a priori knowledge on the HF model, by suitably accounting for a penalization term.

## Acknowledgements

The authors gratefully thank A. Menafoglio and S. Pagani (MOX, Politecnico di Milano) for the interesting and useful discussions about MOR and statistical learning and L. Pegolotti (École Polytechnique Fédérale de Lausanne) for kindly sharing the library `feamat` ([74]).

## References

- [1] A.C. Antoulas, D.C. Sorensen, S. Gugercin, A Survey of Model Reduction Methods for Large-Scale Systems, Tech. Rep., 2000.
- [2] P. Benner, V. Mehrmann, D.C. Sorensen, Dimension Reduction of Large-Scale Systems, vol. 35, Springer, 2005.
- [3] A. Quarteroni, G. Rozza, Reduced Order Methods for Modeling and Computational Reduction, vol. 9, Springer, 2014.
- [4] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations: An Introduction, Vol. 92, Springer, 2015.
- [5] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, *SIAM Rev.* 57 (4) (2015) 483–531.
- [6] A.C. Antoulas, Approximation of Large-Scale Dynamical Systems, vol. 6, SIAM, 2005.
- [7] B. Peherstorfer, K. Willcox, Dynamic data-driven reduced-order models, *Comput. Methods Appl. Mech. Eng.* 291 (2015) 21–41.
- [8] B. Peherstorfer, K. Willcox, Online adaptive model reduction for nonlinear systems via low-rank updates, *SIAM J. Sci. Comput.* 37 (4) (2015) A2123–A2150.

- [9] B. Peherstorfer, S. Gugercin, K. Willcox, Data-driven reduced model construction with time-domain Loewner models, *SIAM J. Sci. Comput.* 39 (5) (2017) A2152–A2178.
- [10] Z. Bai, Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, *Appl. Numer. Math.* 43 (1–2) (2002) 9–44.
- [11] U. Baur, C. Beattie, P. Benner, S. Gugercin, Interpolatory projection methods for parameterized model reduction, *SIAM J. Sci. Comput.* 33 (5) (2011) 2489–2518.
- [12] R.W. Freund, Model reduction methods based on Krylov subspaces, *Acta Numer.* 12 (2003) 267–319.
- [13] B. Moore, Principal component analysis in linear systems: controllability, observability, and model reduction, *IEEE Trans. Autom. Control* 26 (1) (1981) 17–32.
- [14] K. Pearson, LIII. on lines and planes of closest fit to systems of points in space, *Philos. Mag.* 2 (11) (1901) 559–572.
- [15] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, *Q. Appl. Math.* 45 (3) (1987) 561–571.
- [16] H. Hotelling, Analysis of a complex of statistical variables into principal components, *J. Educ. Psychol.* 24 (6) (1933) 417.
- [17] M. Loeve, Probability Theory, vol. II, Graduate Texts in Mathematics, vol. 46, 1978, 387pp.
- [18] J.P. Fink, W. Rheinboldt, On the error behavior of the reduced basis technique for nonlinear finite element approximations, *Z. Angew. Math. Mech.* 63 (1) (1983) 21–28.
- [19] J.S. Peterson, The reduced basis method for incompressible viscous flow calculations, *SIAM J. Sci. Stat. Comput.* 10 (4) (1989) 777–786.
- [20] C. Prud'Homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, G. Turinici, Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods, *J. Fluids Eng.* 124 (1) (2002) 70–80.
- [21] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, Springer, 2016.
- [22] M. Barrault, Y. Maday, N.C. Nguyen, A.T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, *C. R. Math.* 339 (9) (2004) 667–672.
- [23] Y. Maday, N.C. Nguyen, A.T. Patera, G.S. Pau, A general multipurpose interpolation procedure: the magic points, *Commun. Pure Appl. Anal.* 8 (2009) 383.
- [24] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, *SIAM J. Sci. Comput.* 32 (5) (2010) 2737–2764.
- [25] M. Drogmann, B. Haasdonk, M. Ohlberger, Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation, *SIAM J. Sci. Comput.* 34 (2) (2012) A937–A969.
- [26] F. Negri, A. Manzoni, D. Amsallem, Efficient model reduction of parametrized systems by matrix discrete empirical interpolation, *J. Comput. Phys.* 303 (2015) 431–454.
- [27] R. Everson, L. Sirovich, Karhunen–Loeve procedure for gappy data, *J. Opt. Soc. Am. A* 12 (8) (1995) 1657–1664.
- [28] K. Lee, K. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, *arXiv preprint, arXiv:1812.08373*.
- [29] N.M. Alexandrov, R.M. Lewis, C.R. Gumbert, L.L. Green, P.A. Newman, Approximation and model management in aerodynamic optimization with variable-fidelity models, *J. Aircr.* 38 (6) (2001) 1093–1101.
- [30] F. Regazzoni, L. Dedè, A. Quarteroni, Active contraction of cardiac cells: a reduced model for sarcomere dynamics with cooperative interactions, *Biomech. Model. Mechanobiol.* (2018) 1–24.
- [31] A. Quarteroni, A. Veneziani, Analysis of a geometrical multiscale model based on the coupling of ode and pde for blood flow simulations, *Multiscale Model. Simul.* 1 (2) (2003) 173–195.
- [32] W. Hackbusch, On the fast solving of parabolic boundary control problems, *SIAM J. Control Optim.* 17 (2) (1979) 231–244.
- [33] K. Löwner, Über monotone matrixfunktionen, *Math. Z.* 38 (1) (1934) 177–216.
- [34] A. Mayo, A. Antoulas, A framework for the solution of the generalized realization problem, *Linear Algebra Appl.* 425 (2–3) (2007) 634–662.
- [35] S. Lefteriu, A.C. Antoulas, A new approach to modeling multiport systems from frequency-domain data, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 29 (1) (2010) 14–27.
- [36] A.C. Antoulas, I.V. Gosea, A.C. Ionita, Model reduction of bilinear systems in the Loewner framework, *SIAM J. Sci. Comput.* 38 (5) (2016) B889–B916.
- [37] I.V. Gosea, A.C. Antoulas, Model reduction of linear and nonlinear systems in the Loewner framework: a summary, in: *Control Conference (ECC), 2015 European, IEEE*, 2015, pp. 345–349.
- [38] C. Gu, QImor: a projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 30 (9) (2011) 1307–1320.
- [39] D. Deschrijver, T. Dhaene, Rational modeling of spectral data using orthonormal vector fitting, in: *9th IEEE Workshop on Signal Propagation on Interconnects*, 2005, pp. 111–114.
- [40] D. Deschrijver, B. Haegeman, T. Dhaene, Orthonormal vector fitting: a robust macromodeling tool for rational approximation of frequency domain responses, *IEEE Trans. Adv. Packaging* 30 (2) (2007) 216–225.
- [41] D.G. Krige, A statistical approach to some basic mine valuation problems on the Witwatersrand, *J. S. Afr. Inst. Min. Metall.* 52 (6) (1951) 119–139.
- [42] C.E. Rasmussen, Gaussian processes in machine learning, in: *Advanced Lectures on Machine Learning*, Springer, 2004, pp. 63–71.
- [43] A. Menafoglio, P. Secchi, M. Dalla Rosa, A universal kriging predictor for spatially dependent functional data of a Hilbert space, *Electron. J. Stat.* 7 (2013) 2209–2240.
- [44] A.F. Hernandez, M.G. Gallivan, An exploratory study of discrete time state-space models using kriging, in: *American Control Conference*, 2008, IEEE, 2008, pp. 3993–3998.
- [45] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proc. Natl. Acad. Sci.* (2016) 201517384.
- [46] M. Guo, J.S. Hesthaven, Data-driven reduced order modeling for time-dependent problems, *Comput. Methods Appl. Mech. Eng.* 345 (2019) 75–99.
- [47] L. Ljung, System identification, in: *Signal Analysis and Prediction*, Springer, 1998, pp. 163–173.
- [48] K.J. Keasman, System Identification: An Introduction, Springer Science & Business Media, 2011.
- [49] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer Science & Business Media, 2013.
- [50] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.* 1 (1) (1990) 4–27.
- [51] K.S. Narendra, K. Parthasarathy, Neural networks and dynamical systems, *Int. J. Approx. Reason.* 6 (2) (1992) 109–131.
- [52] M. Raissi, P. Perdikaris, G.E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, *J. Comput. Phys.* 348 (2017) 683–693.
- [53] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, *J. Comput. Phys.* 357 (2018) 125–141.
- [54] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, *arXiv preprint, arXiv:1711.10561*.
- [55] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations, *arXiv preprint, arXiv:1711.10566*.
- [56] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707.
- [57] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, *arXiv preprint, arXiv:1801.01236*.

- [58] O. San, R. Maulik, Neural network closures for nonlinear model order reduction, *Adv. Comput. Math.* (2018) 1–34.
- [59] S. Trehan, K.T. Carlberg, L.J. Durlofsky, Error modeling for surrogates of dynamical systems using machine learning, *Int. J. Numer. Methods Eng.* 112 (12) (2017) 1801–1827.
- [60] B.A. Freno, K.T. Carlberg, Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations, *arXiv preprint*, arXiv:1808.02097.
- [61] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.* 2 (4) (1989) 303–314.
- [62] G. Casella, R.L. Berger, *Statistical Inference*, vol. 2, Duxbury, Pacific Grove, CA, 2002.
- [63] G. Cybenko, Continuous Valued Neural Networks with Two Hidden Layers are Sufficient, Technical Report, Department of Computer Science, Tufts University, 1988.
- [64] B. Yegnanarayana, *Artificial Neural Networks*, PHI Learning Pvt. Ltd., 2009.
- [65] S.S. Haykin, *Neural Networks and Learning Machines*, vol. 3, Pearson Upper Saddle River, 2009.
- [66] A. Graves, Generating sequences with recurrent neural networks, *arXiv preprint*, arXiv:1308.0850.
- [67] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, *IEEE Signal Process. Mag.* 29 (6) (2012) 82–97.
- [68] J.T. Connor, R.D. Martin, L.E. Atlas, Recurrent neural networks and robust time series prediction, *IEEE Trans. Neural Netw.* 5 (2) (1994) 240–254.
- [69] J. Nocedal, S. Wright, *Numerical Optimization*, 2nd edition, Springer Science & Business Media, New York, NY, USA, 2006.
- [70] L. Grippo, M. Sciandrone, *Metodi di Ottimizzazione Non Vincolata*, Springer Science & Business Media, 2011.
- [71] Y. Chen, J. White, A quadratic method for nonlinear model order reduction, <https://briefs.techconnect.org/papers/a-quadratic-method-for-nonlinear-model-order-reduction/>.
- [72] M. Rewieński, J. White, A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices, in: *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*, IEEE Press, 2001, pp. 252–257.
- [73] A. Manzoni, S. Pagani, T. Lassila, Accurate solution of Bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models, *Int. J. Uncertain. Quantificat.* 4 (1) (2016) 380–412.
- [74] L. Pegolotti, *Feamat*, <https://github.com/lucapegolotti/feamat>.
- [75] M.D. McKay, R.J. Beckman, W.J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* 42 (1) (2000) 55–61.
- [76] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.
- [77] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.* 12 (Jul) (2011) 2121–2159.
- [78] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, *arXiv preprint*, arXiv:1412.6980.
- [79] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* 15 (1) (2014) 1929–1958.
- [80] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, *arXiv preprint*, arXiv:1502.03167.
- [81] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Freezeout: accelerate training by progressively freezing layers, *arXiv:1706.04983*.