# Breaking complexity in dynamical systems through Artificial Neural Networks

F. Regazzoni[1], L. Dedè[1], and A. Quarteroni[1,2]

[1]MOX - Dipartimento di Matematica, Politecnico di Milano, P.zza Leonardo da Vinci 32, 20133 Milano, Italy
[2]Mathematics Institute, Ecole Polytechnique Fédérale de Lausanne, Av. Piccard, CH-1015 Lausanne, Switzerland (*Honorary Professor*)

December 4, 2018

## Contents

**Abstract**

TODO

## 1 Introduction

The numerical simulation of time-dependent mathematical models plays a fundamental role in the study of complex systems with applications ranging from Natural Sciences to industry, medicine, finance and social behaviour. The increasing demand of more complex and reliable mathematical models may however lead to an unbearable demand for computational resources, either in terms of computational power or memory storage. In many practical applications, moreover, there is the need to perform simulations of a given model 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 space, where a dynamic model needs to be solved virtual in any point of a computational domain. In several contexts, such as computational medicine, the results of complex mathematical models, although precise and reliable, are useless for practical purposes if these cannot be provided 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 model*.

In this paper, we focus on time-invariant systems, whose behaviour is determined by an 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 form for the high-fidelity (HF) model:

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

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 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 the time variable, with equation $\dot{X}_{N+1}(t) = 1$ and initial condition $X_{N+1}(0) = 0$. We also notice that this is not the most general form; indeed the evolution equation can be given in implicit form, however let us stick to this form just to illustrate the concept.

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

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

where $\mathbf{x}(t)$, the reduced-order state, belongs to a lower dimensional space $\mathbb{R}^n$ (with $n \ll N$) and the right-hand side $\mathbf{f}\colon \mathbb{R}^n \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_y}$ can be evaluated with a smaller computational effort than $\mathbf{F}\colon \mathbb{R}^N \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_y}$ in Eq. (1). Notice that, in many practical applications, the knowledge of the evolution of the full-order state $\mathbf{X}(t)$ does not have any practical interest since the final user is interested in just one or a few output quantities, represented by $\mathbf{y}(t)$. Therefore, if the reduced model (2) is able of reproducing in a reliable manner the input-output map $\mathbf{u} \to \mathbf{y}$, then it can be employed in place of the high-fidelity model (1), hopefully with a considerable gain in terms of computational resources and simulation time.

The approaches to MOR can be split in two categories: *model-based* and *data-driven* approaches (Benner, Gugercin, and Willcox 2015). With the first strategy, by exploiting the knowledge of the HF model, Eq. (1) itself is the starting point to derive its reduced version (2). With data-driven methods 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 from the HF model structural properties (e.g. stability) and that the underlying HF system structure provides the base for deriving theoretical error estimates. However, the dynamics of the HF model in terms of an equation in the form of (1) is not always available, but it may be accessible only trough 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. trough samples of the transfer function, in the case of a linear system), or when the HF system is accessible trough the simulations of a black-box code. 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 endowed with a non-intrusive nature 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 (Antoulas 2005; Antoulas, Sorensen, and Gugercin 2000; Benner, Gugercin, and Willcox 2015; Benner, Mehrmann, and Sorensen 2005). In this framework, the full-order state space $\mathbb{R}^N$ is approximated by a lower-dimensional subspace 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). This is equivalent to imposing orthogonality of the HF residual to span($\mathbf{V}$) (or 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 the first few moments, up to a desired order, of the full-order transfer function for a given frequency (Bai 2002; Baur et al. 2011; Freund 2003). The Balanced-Truncation approach applies in the linear case too: it consists in neglecting the states corresponding to the smallest Hankel singular values, which measure the relevance of each state in terms of both reachability and observability (Antoulas 2005; Moore 1981). Proper Orthogonal Decomposition (POD), whose original concept goes back to Pearson 1901, was developed by Sirovich in 1987 (see Sirovich 1987) and it is tightly related to Principal Component Analysis (PCA, see Hotelling 1933) and to Karhunen-Loève expansion, also known as Hotelling transform in stochastic process theory (Loeve 1978). 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, which can be computed in a straightforward way through Singular Value Decomposition (SVD) of the snapshot matrix itself. Such basis is optimal in the sense that it minimizes the least-square error of snapshot reconstruction (Antoulas 2005). The use of POD has gained a huge popularity in the PDE framework, where the POD Galerkin projection-based approach is known with the name of Reduced Basis (RB) method (see e.g. Fink and Rheinboldt 1983; Hesthaven, Rozza, and Stamm 2016; Peterson 1989; Quarteroni, Manzoni, and Negri 2015).

The effectiveness of projection-based MOR relies in the possibility of performing an offline/online decoupling of the projection. 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) describing the reduced model can be precomputed *offline* (i.e. during the construction of the reduced model itself) by projection of the full-order algebraic structures (Benner, Mehrmann, and Sorensen 2005; Quarteroni, Manzoni, and Negri 2015); these do not need to be accessible in the *online* phase (i.e. during the actual 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 undermining the attempt to reduce the complexity of the model. To overcome this issue, the nonlinear dependence on either the state, the input or both is typically replaced by affine approximations by employing techniques such as the empirical interpolation method, EIM (Barrault et al. 2004; Maday et al. 2007), the discrete empirical interpolation method, DEIM (Chaturantabut and Sorensen 2010; Drohmann, Haasdonk, and Ohlberger 2012), its matrix version MDEIM (Negri, Manzoni, and Amsallem 2015) and the gappy POD reconstruction (Everson and Sirovich 1995).

Besides projection-based methods, another class of model-based MOR techniques is that of hierarchical surrogates, that is to say models derived, starting from the HF one, under simplified physical assumptions, simplified geometries or coarser computational grids. While this approach is ubiquitous in several applications, we just limit to mention a few in different fields, e.g. Alexandrov et al. 2001; Hackbusch 1979; Quarteroni and Veneziani 2003; Regazzoni, Dedè, and Quarteroni 2018. Most of the methods of this category are dependent on the class of differential problem that describes the phenomenon or are tailored to a specific model.

## 1.2 Data-driven MOR

In the Loewner framework, whose original ideas date back to Löwner 1934, a linear reduced model is derived starting 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 algebraic structures are computed in such a way that the reduced model transfer function interpolates the full-order one out the interpolation points and in the tangential directions (see Lefteriu and Antoulas 2010; Mayo and Antoulas 2007). Even if the Loewner framework only applies to linear models, it has recently been extended to bilinear (i.e. independently linear in the state and in the input) models (Antoulas, Gosea, and Ionita 2016), quadratic-bilinear models (Gosea and Antoulas 2015) and to analytic nonlinear models with affine input dependence. This has been allowed by rewriting models with analytic nonlinearities in the state as quadratic-bilinear models (increasing however the size of the original model, see Gu 2011).

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, which have been shown to enhance numerical stability in the identification of the approximation coefficients (Deschrijver and Dhaene 2005; Deschrijver, Haegeman, and Dhaene 2007).

Kriging, originally developed in the geosciences field (Krige 1951) and also known as Gaussian Process regression, is widely used to perform MOR for nonlinear models in the steady-state case (Menafoglio, Secchi, and Dalla Rosa 2013; Rasmussen 2004). Kriging is a regression method which employs, as prior for the outcome of a function, a Gaussian Process whose covariance function depend on so–called hyperparameters, tuned according to a maximum likelihood principle. This technique has been extended to the dynamical case in Hernandez and Gallivan 2008 under the name of dynamic mapping kriging (DMK), where the authors, by considering the discrete time version of the evolution equation (1) (i.e. $\mathbf{x}^{k+1} = \mathbf{f}(\mathbf{x}^k, \mathbf{u}^k)$), perform kriging on the function $\mathbf{f}$ starting from sample data for several $(\mathbf{x}, \mathbf{u})$ pairs.

In Brunton, Proctor, and Kutz 2016 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 under the hypothesis that $\mathbf{f}$ depends just on a few combinations of the inputs (such as linear combinations, products, and trigonometric functions), by seeking a sparse solution for the coefficients of a predetermined collection of linear and nonlinear terms.

Both DMK and SINDy techniques, despite they can be applied to nonlinear systems, require the full-order state of the model to be accessible and, most of all, do not perform any reduction in the state dimension.

## 1.3 Learning models from data

As mentioned before, 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 family of mathematical models: in practice, when one is unable to explicitly build a model or may not be interested in building it. This is the case when a physical system is accessible through measurements and one tries to build a mathematical model starting from data, that is to say to *identify* the underlying law linking the input-output pairs. This task is commonly known in the field of control theory as *System Identification*, SI (see e.g. Keesman 2011; Ljung 1998). 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 dynamic models (see

NARX/NARMAX models, Nelles 2013), i.e. discrete time models in the form

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

where the prediction for the next output $\widehat{\mathbf{y}}^{k+1}$ depend on the value of the previous $p+1$ output measurements $\{\widehat{\mathbf{y}}^j\}_{j=k-p}^{k-1}$ and the previous $q+1$ inputs $\{\widehat{\mathbf{u}}^j\}_{j=k-q}^{k}$. However, models laying in this family are designed for online identification and, most of all, for online predictions: the model should be fed with the measured output at previous iterations, so that the *true* output should be available not only at the identification (or training/offline) stage, but also at the prediction (online) stage. However, we are looking here for a model that can be used in a stand-alone way in the online stage, once an offline training stage has been carried out. We notice that the use of Artificial Neural Networks in the context of nonlinear SI is quite popular (Narendra and Parthasarathy 1990, 1992; Nelles 2013), 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 Gaussian Processes (Raissi and Karniadakis 2018; Raissi, Perdikaris, and Karniadakis 2017a) or Artificial Neural Networks (Raissi, Perdikaris, and Karniadakis 2017b,c), 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 coefficients, reaction coefficients, etc.) are unknown.

## 1.4 Original contributions and outline

In this work we address the problem of data-driven MOR for nonlinear dynamical systems (which can interpreted, as previously noticed, 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 (1) reconstructing the internal state of the system trough its reduced description $\mathbf{x}(t)$ without the possibility of observing the true internal state of the system $\mathbf{X}(t)$ and (2) 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 $\mathbf{u} \to \mathbf{y}$.

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 in order to derive an optimization algorithm to solve the best-approximation problem. Because of their capacity to approximate any continuous function with a desired level of accuracy (see Cybenko 1989) and their ability to learn from data, we represent the model right-hand side $\mathbf{f}$ in (2) through an Artificial Neural Network (ANN), which we train in such a way that it learns from input-output pairs the underlying physics.

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: (1) how the unknowns of the problem are discretized; (2) how the time discretization is performed; (3) the optimization algorithm employed. Finally, in Sec. 4, we show the obtained results and we analyse and critically discuss them.

## 2 Model reduction strategy

We start by giving a definition of *model* (at least what we intend here in this paper). We use the term model even if we do not need to access to a HF mathematical model: all we need is just a black-box which maps time-dependent inputs in time-dependent outputs. On these bases, we give an abstract definition of model, which must fulfil some minimal assumptions.

### 2.1 The dynamical model

A *model* for us denotes a general framework including either on object (a group of interacting items, a natural phenomenon, a industrial plant, a group of agents, etc.) or a mathematical or numerical model, which associates a time-dependent output to a time-dependent input. We denote by $U \subset \mathbb{R}^{N_u}$ and by $Y \subset \mathbb{R}^{N_y}$ the sets where the input $\mathbf{u}(t)$ and the output $\mathbf{y}(t)$ take values, respectively. We denote by the term *experiment* the action of giving an input $\mathbf{u}(t)$ to the model and recording the corresponding output $\mathbf{y}(t)$; a and by the term *sample* the couple $(\mathbf{u}, \mathbf{y})$. We make the following minimal assumptions on the model, aiming at abstracting from its specific form the structure of the input-output map represented by Eq. (1):

(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 an other 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 can depend just on past values of the input and not on future values. In other words, 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 of 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 neglected, by allowing $\mathbf{g}$ in (2) to depend also on $\mathbf{u}(t)$. However, for the sake of simplicity, we will not consider this case in this work.

We consider a limited time interval $(0, T)$ and, 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 $\boldsymbol{\varphi}\colon \mathcal{U} \to \mathcal{Y}$ from the space of input signals $\mathcal{U} = \mathcal{C}([0, T]; U)$ to the space of output signals $\mathcal{Y} = \mathcal{C}([0, T]; Y)$. Thanks to assumptions (A1) and (A2) the map $\boldsymbol{\varphi}$ is well defined. Moreover, assumption (A3) can be written as

$$\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U} \quad \forall t^* \in [0, T] \qquad \mathbf{u}_1|_{[0,t^*]} = \mathbf{u}_2|_{[0,t^*]} \implies (\boldsymbol{\varphi}\mathbf{u}_1)|_{[0,t^*]} = (\boldsymbol{\varphi}\mathbf{u}_2)|_{[0,t^*]}, \quad (3)$$

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

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

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

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

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

## 2.2 Abstract formulation of the model reduction problem

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

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

Then we suppose to 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 $\widehat{\Phi}$ :

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

where the objective functional is given by

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

We notice that, when the measures of the output $\widehat{\mathbf{y}}_j$ are affected by gaussian noise, the least-square best-approximation corresponds to the maximum-likelihood estimation (see e.g. Casella and Berger 2002).

The next step is that of selecting in a suitable way $\widehat{\Phi} \subseteq \Phi$, the subset of candidate models. A possible approach is that of approximating directly the input-output map $\mathbf{u} \to \mathbf{y}$ 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 ruled 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 of models represents a subset of $\Phi$, as we will show. First, we notice that, given two functions $\mathbf{f} \in \mathcal{F}_n := \mathrm{Lip}(\mathbb{R}^n \times U; \mathbb{R}^n)$ and $\mathbf{g} \in \mathcal{G}_n := \mathrm{Lip}(\mathbb{R}^n; Y)$ 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) uniquely identifies a map from $\mathcal{U}$ to $\mathcal{Y}$. We denote by $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$ 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 $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0} \in \Phi$, 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-Lindelf theorem, Eq. (2) has a unique solution. Thus, the map $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$ is well defined. Property (4) is easy to be checked, while (5) holds by setting $\mathbf{y}_0 = \mathbf{g}(\mathbf{x}_0)$. □

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:

$$\Phi^{\widehat{\mathcal{F}},\widehat{\mathcal{G}},\widehat{\mathcal{X}}} = \left\{ \boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0} \in \Phi \text{ s.t. } \mathbf{f} \in \widehat{\mathcal{F}}, \mathbf{g} \in \widehat{\mathcal{G}}, \mathbf{x}_0 \in \widehat{\mathcal{X}} \right\} \subset \Phi. \tag{9}$$

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

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

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

*Proof.* Given a model in the form Eq. (2) of with $n$ internal variables, by adding $m-n$ further variables which do not affect neither the dynamics of the other variables, nor the output, we get a model with $m$ internal variables, still representing the same input-output map of the previous one. □

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

$$\begin{cases} \displaystyle\min_{\mathbf{f}\in\widehat{\mathcal{F}},\mathbf{g}\in\widehat{\mathcal{G}},\mathbf{x}_0\in\widehat{\mathcal{X}}} & \frac{1}{2}\sum_{j=1}^{N_s} \int_0^T |\widehat{\mathbf{y}}_j(t) - \mathbf{g}(\mathbf{x}_j(t))|^2 dt \\ \text{s.t.} & \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)), \quad t \in (0,T], \quad j = 1,\dots,N_s \\ & \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1,\dots,N_s, \end{cases} \tag{10}$$

We are thus addressing a least-squares minimization problems where the design variables are the two functions $\mathbf{f}$ and $\mathbf{g}$ and the vector $\mathbf{x}_0$.

## 2.4 Non-uniqueness of the representation

We make the following important remark:

*Remark* 1. Given a model $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0} \in \Phi^{\widehat{\mathcal{F}},\widehat{\mathcal{G}},\widehat{\mathcal{X}}}$, its representation in terms of $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$ may be not unique. Indeed, by taking any invertible and sufficiently regular map $\mathbf{h}\colon \mathbb{R}^n \to \mathbb{R}^n$ and by defining

$$\begin{aligned} \widehat{\mathbf{f}}(\widehat{\mathbf{x}}, \mathbf{u}) &= (\nabla\mathbf{h} \circ \mathbf{h}^{-1})(\widehat{\mathbf{x}})\ \mathbf{f}(\mathbf{h}^{-1}(\widehat{\mathbf{x}}), \mathbf{u}) \\ \widehat{\mathbf{g}}(\widehat{\mathbf{x}}) &= \mathbf{g}(\mathbf{h}^{-1}(\widehat{\mathbf{x}})) \\ \widehat{\mathbf{x}}_0 &= \mathbf{h}(\mathbf{x}_0), \end{aligned} \tag{11}$$

we have $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0} = \boldsymbol{\varphi}_{\widehat{\mathbf{f}},\widehat{\mathbf{g}},\widehat{\mathbf{x}}_0}$ (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 $\widehat{\mathbf{f}}(\widehat{\mathbf{x}}, \mathbf{u}) = \alpha\mathbf{f}(\widehat{\mathbf{x}}/\alpha, \mathbf{u})$, $\widehat{\mathbf{g}}(\widehat{\mathbf{x}}) = \mathbf{g}(\widehat{\mathbf{x}}/\alpha)$, $\widehat{\mathbf{x}}_0 = \alpha\mathbf{x}_0$, the triplets $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$ and $(\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{x}}_0)$ identify the same model.

We notice that, 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 (11) – their equivalent counterparts $(\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{x}}_0)$, the solution of problem (7) may loose uniqueness, in terms of its $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$ representation. This is certainly an issue since non-uniqueness can deteriorate the optimization algorithm performances. Nevertheless, it may be seen also as an opportunity: we can indeed decrease the size of the spaces $\widehat{\mathcal{F}}$, $\widehat{\mathcal{G}}$ and $\widehat{\mathcal{X}}$, by imposing specific constraints on the solution, in order 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 $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$. Consider then the (invertible) state transformation $\widehat{\mathbf{x}} = \mathbf{h}_1(\mathbf{x}) = \mathbf{x} - \mathbf{x}_0$. By applying the transformation (11), we get an equivalent model $\boldsymbol{\varphi}_{\widehat{\mathbf{f}},\widehat{\mathbf{g}},\widehat{\mathbf{x}}_0}$, where $\widehat{\mathbf{f}}(\widehat{\mathbf{x}}, \mathbf{u}) = \mathbf{f}(\widehat{\mathbf{x}}+\mathbf{x}_0, \mathbf{u})$, $\widehat{\mathbf{g}}(\widehat{\mathbf{x}}) = \mathbf{g}(\widehat{\mathbf{x}}+\mathbf{x}_0)$ and $\widehat{\mathbf{x}}_0 = \mathbf{0}$. Therefore, when we look for the solution of the best-approximation problem (7) we can 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 under the constraint $\mathbf{x}_0 = \mathbf{0}$. The statement of the best-approximation problem has been improved since the number of design variables has decreased (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 $\mathbf{x}_0$

A consequence of Remark 1 is that the variables $\mathbf{x}$ are just tools to track the time evolution of the internal state of the model, without 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 $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$, 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 yes, then we can restrict ourselves to models such that $\mathbf{g}$ 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 to this question, consider a model $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$ and suppose that there exists a smooth function $\mathbf{q}\colon \mathbb{R}^n \to \mathbb{R}^{n-N_y}$ – we will go back later to the issue of its existence – 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 (11) with $\mathbf{h}_2$, we get the equivalent model $\boldsymbol{\varphi}_{\widehat{\mathbf{f}},\widehat{\mathbf{g}},\widehat{\mathbf{x}}_0}$ where:

$$\begin{aligned} \widehat{\mathbf{f}}(\widehat{\mathbf{x}}, \mathbf{u}) &= (\nabla \mathbf{h}_2 \circ \mathbf{h}_2^{-1})(\widehat{\mathbf{x}})\ \mathbf{f}(\mathbf{h}_2^{-1}(\widehat{\mathbf{x}}), \mathbf{u}) \\ \widehat{\mathbf{g}}(\widehat{\mathbf{x}}) &= \boldsymbol{\pi}^{N_y}(\widehat{\mathbf{x}}) \\ \widehat{\mathbf{x}}_0 &= (\mathbf{g}^T(\mathbf{x}_0), \mathbf{q}^T(\mathbf{x}_0))^T, \end{aligned} \tag{12}$$

which has the property, as we wanted, that the output is given by the first $N_y$ state variables. We notice that, in the expression of the initial condition $\widehat{\mathbf{x}}_0$, we can replace $\mathbf{g}(\mathbf{x}_0) = \mathbf{y}_0$, which is available from the measurements. Moreover, as in the previous case, we can suppose without loss of generality $\mathbf{q}(\mathbf{x}_0) = \mathbf{0}$ (this can be obtained by applying once again (11) 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 following form:

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \qquad t \in (0, T] \\ \mathbf{x}(0) &= (\mathbf{y}_0^T, \mathbf{0}^T)^T \end{cases} \tag{13}$$

where the state is written in the form $\mathbf{x} = (\mathbf{y}^T, \mathbf{z}^T)^T$, where we call $\mathbf{z}(t)$ the *hidden variables.* We have not still ruled out all the ambiguity since, being the output transparent to the hidden 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 $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$ and $\widehat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$.

To summarize, we have thus shown that all the models admitting the existence of a function $\mathbf{q}\colon \mathbb{R}^n \to \mathbb{R}^{n-N_y}$ such that $\mathbf{h}_2(\mathbf{x}) = (\mathbf{g}^T(\mathbf{x}), \mathbf{q}^T(\mathbf{x}))^T$ is invertible are equivalent to a model in the form of (13). Clearly, this hypothesis is not fulfilled by any model $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$: if, for instance, the outputs $y_1, \dots, y_{N_y}$ are linearly dependent, then $\mathbf{h}_2$ cannot be invertible. However, this case is of no practical interest since in such case the dimension of the output can be reduced. It follows that, assuming that the image of $\mathbf{g}$ spans an $N_y$-dimensional variety in $\mathbb{R}^n$, it is reasonable that a function $\mathbf{q}(\mathbf{x})$ such that $\mathbf{h}_2(\mathbf{x})$ is (at least locally) invertible exists. Even if this cannot be rigorously proven without the introduction of technical assumptions, 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

In the following we will consider both the cases considered in Sec. 2.4:

1. $\widehat{\mathcal{X}} = \{\mathbf{0}\}$, which we call *output-outside-the-state approach* (since the output is a function of the state, but it is not incorporated into the state);
2. $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$ and $\widehat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$, which we call *output-inside-the-state approach* (since the state includes the output in itself).

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

- Collect the input-output observations (6) of the model to approximate (i.e. to reduce or to identify).
- Select a suitable state dimension $n \geq 1$, the subset $\widehat{\mathcal{F}}$ and, in the output-outside-the-state approach, the subset $\widehat{\mathcal{G}}$.
- Solve the abstract problem (7), which reads, in the output-outside-the-state approach:

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

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

$$\begin{cases} \min\limits_{\mathbf{f}\in\widehat{\mathcal{F}}} & \frac{1}{2}\sum_{j=1}^{N_s}\int_0^T |\widehat{\mathbf{y}}_j(t) - \boldsymbol{\pi}^{N_y}(\mathbf{x}_j(t))|^2 dt \\ \text{s.t.} & \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)), \quad t \in (0,T], \quad j = 1,\dots,N_s \\ & \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1,\dots,N_s, \end{cases} \tag{15}$$

where $\mathbf{x}_0 = (\mathbf{y}_0^T, \mathbf{0}^T)^T$. We notice that (15) can be seen as a particular case of (14) (by putting $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$); therefore, in the following we will confine ourselves, without loss of generality, to (14).

## 2.6 On the choice of the sets $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$

The richness of the spaces $\widehat{\mathcal{F}}$ (and, in the output-outside-the-state approach, $\widehat{\mathcal{G}}$) should be chosen according to the Occam's razor principle of parsimony (William of Ockham, 1287–1347, English Franciscan friar, scholastic philosopher and theologian), by which *frustra fit per plura quod fieri potest per pauciora* (It is useless to do with more what can be done with less). One should avoid the two opposite situations: when too poor spaces are considered, the expressive power of the model class $\Phi^{\widehat{\mathcal{F}},\widehat{\mathcal{G}},\widehat{\mathcal{X}}}$ is too small to capture the complexity of the HF model; on the other hand, if $\widehat{\mathcal{F}}$ (and $\widehat{\mathcal{G}}$) are too rich (in the extreme, $\widehat{\mathcal{F}} = \mathcal{F}_n$ and $\widehat{\mathcal{G}} = \mathcal{G}_n$), we expect a very good match on the train set (6), but this typically results into overfitting. The compromise stays in the middle, where the so-called Occam's hill is located (see Rasmussen and Ghahramani 2001), i.e. where the richness of the spaces $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$ is enough to satisfactorily reproduce the observations (6), but not beyond.

## 2.7 The solution strategy

The unknowns of problem (14) are the functions $\mathbf{f}$ and, in the output-outside-the-state approach, $\mathbf{g}$. We are thus performing optimization in functional spaces. It is interesting to look at the first-order optimality condition 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 (14). Thus, we introduce a family of Lagrange multipliers $\mathbf{w}_j \in \mathcal{C}([0,T];\mathbb{R}^n)$, for $j = 1,\dots,N_s$, associated to the constrains given by the state equations:

$$\begin{aligned} \mathcal{L}(\mathbf{f},\mathbf{g},\{\mathbf{x}_j\}_j,\{\mathbf{w}_j\}_j) = & \frac{1}{2}\sum_{j=1}^{N_s}\int_0^T |\widehat{\mathbf{y}}_j(t) - \mathbf{g}(\mathbf{x}_j(t))|^2 dt \\ & - \sum_{j=1}^{N_s}\int_0^T \mathbf{w}_j(t)\cdot(\dot{\mathbf{x}}_j(t) - \mathbf{f}(\mathbf{x}_j(t),\widehat{\mathbf{u}}_j(t)))\,dt \\ & - \sum_{j=1}^{N_s}\mathbf{w}_j(0)\cdot(\mathbf{x}_j(0) - \mathbf{x}_0) \end{aligned} \tag{16}$$

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

$$\begin{cases} -\dot{\mathbf{w}}_j(t) & = \nabla_{\mathbf{x}}^T\mathbf{g}(\mathbf{x}_j(t))\ (\mathbf{g}(\mathbf{x}_j(t)) - \widehat{\mathbf{y}}_j(t)) + \nabla_{\mathbf{x}}^T\mathbf{f}(\mathbf{x}_j(t),\widehat{\mathbf{u}}_j(t))\,\mathbf{w}_j(t) \\ \mathbf{w}_j(T) & = \mathbf{0} \end{cases} \tag{17}$$

The first-order optimality conditions are given by setting equal to zero the Gateaux 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 $\delta\mathbf{f}$ and $\delta\mathbf{g}$:

$$\begin{cases} \langle \frac{\partial J}{\partial \mathbf{f}}, \delta\mathbf{f} \rangle = \sum_{j=1}^{k} \int_0^T \delta\mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t)) \cdot \mathbf{w}_j(t) dt = 0 & \forall\, \delta\mathbf{f} \in \mathcal{F}_n \\ \langle \frac{\partial J}{\partial \mathbf{g}}, \delta\mathbf{g} \rangle = \sum_{j=1}^{k} \int_0^T \delta\mathbf{g}(\mathbf{x}_j(t)) \cdot (\mathbf{g}(\mathbf{x}_j(t)) - \widehat{\mathbf{y}}_j(t))\, dt = 0 & \forall\, \delta\mathbf{g} \in \mathcal{G}_n \end{cases} \tag{18}$$

By (18) it is evident that the two differentials cannot be written in gradient form, unless $\mathbf{x}_j(t)$ and $\widehat{\mathbf{u}}_j(t)$ span all $\mathbb{R}^n$ and $U$, which is never the case for a finite number of samples $N_s$ and finite time $T$. This accounts to say, as one might expect, that the unknowns are underdetermined in the points of their domain we do not have direct observations. This has to be compensated for by proper regularization of the unknown themselves, by Occam's razor principle (see Sec. 2.6).

Our strategy is that of parametrizing both functions by a finite set of real parameters, which we call respectively $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$ and $\boldsymbol{\nu} \in \mathbb{R}^{N_g}$, and then tackling problem (14) 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 required regularization is obtained in a natural way by controlling the size of $N_f$ and $N_g$ since the complexity of candidate models is bounded 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:

$$\begin{cases} \nabla_{\boldsymbol{\mu}} J = \displaystyle\sum_{j=1}^{k} \int_0^T \nabla_{\boldsymbol{\mu}}^T \mathbf{f}(\mathbf{x}_j(t), \widehat{\mathbf{u}}_j(t); \boldsymbol{\mu})\, \mathbf{w}_j(t) dt \\ \nabla_{\boldsymbol{\nu}} J = \displaystyle\sum_{j=1}^{k} \int_0^T \nabla_{\boldsymbol{\nu}}^T \mathbf{g}(\mathbf{x}_j(t); \boldsymbol{\nu})\ (\mathbf{g}(\mathbf{x}_j(t); \boldsymbol{\nu}) - \widehat{\mathbf{y}}_j(t))\, dt. \end{cases} \tag{19}$$

We notice that the same result can be obtained by differentiating the Lagrangian (16) with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. In the next section we will derive the discrete counterpart of (17) and (19), which we will employ to tackle the solution of problem (14) from a numerical viewpoint.

The parametrization of $\mathbf{f}$ and $\mathbf{g}$ can be obtained in different manners, such as by polynomial approximation, truncated Fourier series, Finite Elements, Spectral elements, splines, etc. Here we choose to represent them by ANNs, which can be seen as nonlinear maps parametrized by a finite number of scalars (see Sec. 3.1). Our choice is driven by the universal approximation properties of ANNs (see Cybenko 1988, 1989) and their well assessed skills in learning from data. Moreover, the global influence of the parameters of the ANN makes them more suitable than localized basis, such as Finite Elements, due to the fact that the sensitivity of the objective functional to the unknown functions is available only in a small part of the domain (see Eq. 18).

## 2.8 Training strategy

The training strategy is the following. We generate randomly an initial guess for $\mathbf{f}^{(0)}$ and $\mathbf{g}^{(0)}$, and we solve model $\boldsymbol{\varphi}^{(0)} := \boldsymbol{\varphi}_{\mathbf{f}^{(0)}, \mathbf{g}^{(0)}, \mathbf{x}_0}$ on the train set $\{\widehat{\mathbf{u}}_j\}_j$, thus getting $\mathbf{y}_j^{(0)} = \boldsymbol{\varphi}^{(0)}\, \widehat{\mathbf{u}}_j$. Then we evaluate the errors $\|\widehat{\mathbf{y}}_j - \mathbf{y}_j^{(0)}\|_{L^2}$ and, by a suitable descent strategy, we update the unknowns getting $\mathbf{f}^{(1)}$ and $\mathbf{g}^{(1)}$ and the associated model $\boldsymbol{\varphi}^{(1)}$. By iterating this process, we get a sequence of models $\boldsymbol{\varphi}^{(k)}$ and of associated functions $\mathbf{f}^{(k)}$ and $\mathbf{g}^{(k)}$. At each iteration $k$ (to which we will refer as *epoch*), the optimization strategy exploits the internal state $\mathbf{x}_j^{(k)}(t)$ to drive the dynamics, including that of the outputs $\mathbf{y}_j^{(k)}(t)$. The learning process is thus pushed in a spontaneous way to make the internal state be representative of the actual state of the HF model which generated the training data. 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 $\dot{\mathbf{y}}$. This is somehow similar to projection-based MOR strategies (see Sec. 1.1), with the important differences that: (1) in this case the map is not explicitly built, and thus is not available; (2) while in projection-based methods the map is always linear, here also nonlinear mappings can be exploited; (3) since the effects of the internal state $\mathbf{X}$ are seen just though $\mathbf{y}$, only the features which are relevant for the input-output map are exploited, whereas POD may extract features from the snapshot which are not relevant in this sense.

## 3 Optimization strategy

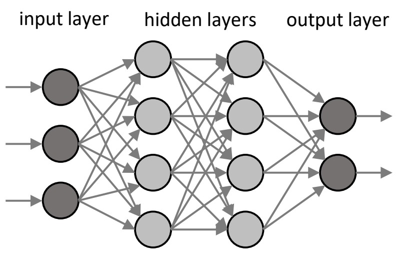
In Sec. 2 we have stated the model reduction problem in terms of a constrained optimization problem. In this section we address the problem of numerically finding an approximate solution for such problem.

### 3.1 Representation of the unknown in terms of ANN

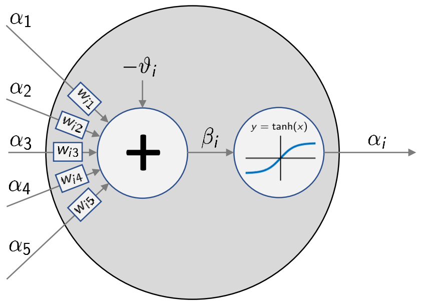
As previously mentioned, in this work we represent the unknowns $\mathbf{f}$ and $\mathbf{g}$ by means of ANNs, trough the parameters $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$ and $\boldsymbol{\nu} \in \mathbb{R}^{N_g}$. An ANN consists in a number of simple processing units (the *neurons*), each one incorporating a nonlinear mapping, interconnected to form a complex network (see e.g. Yegnanarayana 2009). In this work we consider the case of feed-forward ANNs (also known as multilayer perceptrons), consisting in $n_L$ 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. 1a).

Each connection in the network is endowed with a weight (we denote by $w_{ij} \in \mathbb{R}$ the weight of the connection form the $j$-th the $i$-th neuron) and each neuron, but for the ones in the input layer, is characterized by an activation threshold $\vartheta_i \in \mathbb{R}$. The $i$-th neuron of the network takes as input the weighted sum of the output values of the neurons of the previous layer, shifted by the threshold $\vartheta_i$ (i.e. $\beta_i = \sum_{j \in I_i} w_{ij}\alpha_j - \vartheta_i$, where $I_i$ is the set of the neurons of the previous layer) and applies to the result a nonlinear function (i.e. $\alpha_i = f_{\text{act}}(\beta_i)$). The activation function $f_{\text{act}}$ is a sigmoidal nonlinear function, mimicking the Heaviside-like activation function of biological neurons. In this work we take $f_{\text{act}}(s) = \tanh(s)$. In Fig. 1b a schematic picture of a generic neuron is shown.

Thus, by denoting with $n_I$ and $n_O$ the number of neurons in the input and output layers respectively, an ANN can be seen as a map from $\mathbb{R}^{n_I}$ to $\mathbb{R}^{n_O}$, where the input data flow through the layers, from the first to the last one. The values $\alpha_i$ associated to the input layer coincide with the input of the ANN, while the output of the ANN consists in the values $\beta_i$ of the output layer (notice that the activation function is not applied to the last later). In other words, let us denote by $\mathbf{q} = \mathbf{f}(\mathbf{p}; \boldsymbol{\mu})$ the map represented by the ANN, where $\mathbf{p} \in \mathbb{R}^{n_I}$, $\mathbf{q} \in \mathbb{R}^{n_O}$ and $\boldsymbol{\mu}$ is the vector collecting both the weights $w_{ij}$ and the thresholds $\vartheta_i$. Let $i_1, \dots, i_{n_I}$ be the indexes of the neurons of the input layer and $o_1, \dots, o_{n_O}$ be the indexes of the neurons of the output layer. Then we have $\alpha_{i_h} = q_h$ for $h = 1, \dots, n_I$ and $p_h = \beta_{o_h}$ for $h = 1, \dots, n_O$.



(a) Feed-forward ANN



(b) Scheme of the $i$-th neuron

Figure 1: Scheme of a Feed-forward ANN with two hidden layers (a) and of a general neuron (b).

In a typical machine learning algorithm, the number of layers and of neurons are fixed a priori and the values of the parameters $\boldsymbol{\mu}$ are optimized according to a proper learning strategy. In particular, in the *supervised learning* framework, a measure of the performance of the network is available, so that gradient-based optimization algorithms can be applied to update the parameters $\boldsymbol{\mu}$ in an iterative manner, until a convergence criterion is satisfied.

In the following, we show some useful results on ANNs which will be exploited in the following sections.

### 3.1.1 Computing the sensitivities of the ANN output

The calculation by means of the chain rule of the sensitivity of the output $\mathbf{p}$ with respect to the input $\mathbf{q}$ or to the parameters $\boldsymbol{\mu}$ (which we denote respectively by $\nabla_{\mathbf{x}}\mathbf{f}$ and $\nabla_{\boldsymbol{\mu}}\mathbf{f}$) give raise to the so-called back-propagation formulas. To derive such formulas, we need first to compute the sensitivity of the network output $\mathbf{p}$ with respect to the neurons outputs $\alpha_j$. Thus, we have, for $h = 1, \dots, n_O$, and for any $j$ belonging to the inner layer or to the hidden layers:

$$\frac{\partial p_h}{\partial \alpha_j} = \begin{cases} w_{o_h j} & \text{if } j \in I_{o_h} \\ \displaystyle\sum_{k \text{ s.t. } j \in I_k} \frac{\partial p_h}{\partial \alpha_k} f'_{\text{act}}(\beta_k) w_{jk} & \text{otherwise.} \end{cases}$$

Notice that to derive the sensitivities associated to a given layer, the sensitivities of the following layer are required, due to the chain rule. For this reason, the computation should be performed first for the last hidden layer and then back to the input layer (whence the name of *back-propagation*). Then, we have, for $h = 1, \dots, n_O$, for $k =$

$1, \dots, n_I$ and $j \in I_i$:

$$\begin{aligned}
\frac{\partial p_h}{\partial q_k} &= \frac{\partial p_h}{\partial \alpha_{i_k}}, \\
\frac{\partial p_h}{\partial \vartheta_i} &= \begin{cases} -\frac{\partial p_h}{\partial \alpha_i} f'_{\text{act}}(\beta_i) & \text{if } i \text{ does not belong to the output layer} \\ -1 & \text{if } i = o_h \\ 0 & \text{otherwise,} \end{cases} \\
\frac{\partial p_h}{\partial w_{ij}} &= -\alpha_j \frac{\partial p_h}{\partial \vartheta_i}.
\end{aligned}$$

### 3.1.2 Compact representation of the ANN

By denoting with $N_I$ the set of indices of neurons belonging to the layer $I = 1, \dots, n_L$, we collect the weights of the synapses between the layer $I$ and the layer $I+1$ in a matrix, which we denote by $\mathrm{W}_I = [w_{ij}]_{i \in N_I, j \in N_{I+1}}$, for $I = 1, \dots, n_L - 1$. In a similar manner, we collect the thresholds of the same synapses in a vector, which we denote by $\boldsymbol{\vartheta}_I = [\vartheta_i]_{i \in N_I}$. With this notation, the map represented by the ANN can be written as follows:

$$\mathbf{f}(\mathbf{p}; \boldsymbol{\mu}) = \mathrm{W}_{n_L-1} f_{\text{act}} \left( \dots \mathrm{W}_2 f_{\text{act}} \left( \mathrm{W}_1 \mathbf{p} - \boldsymbol{\vartheta}_1 \right) - \boldsymbol{\vartheta}_2 \dots \right) - \boldsymbol{\vartheta}_{n_L-1}, \tag{20}$$

where the application of the activation function $f_{\text{act}}$ must be interpreted componentwise. We notice that the parameters vector $\boldsymbol{\mu}$ collects the entries of the matrices $\mathrm{W}_I$ and of the vectors $\boldsymbol{\vartheta}_I$.

### 3.1.3 Transforming the ANN through affine changes of variables

Thanks to the compact representation (20), it is easy to derive the parameters of the ANN arising from another ANN after an invertible and affine change of variables. Consider the following affine transformation of both the input and the output of the ANN $\mathbf{q} = f(\mathbf{p}; \boldsymbol{\mu})$: $\widehat{\mathbf{p}} = \mathrm{A}\mathbf{p} + \mathbf{b}$, $\widehat{\mathbf{q}} = \mathrm{C}\mathbf{q} + \mathbf{d}$. By simple calculations it turns out that the weights $\widehat{\boldsymbol{\mu}}$ such that $\widehat{\mathbf{q}} = f(\widehat{\mathbf{p}}; \widehat{\boldsymbol{\mu}})$ are obtained by substituting the weight and the thresholds associated to the first and last layers of synapses as follows:

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

### 3.1.4 The importance of normalization

Even if ANNs can work with data spanning different order of magnitudes, their performances are optimized 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 roughly 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, which can be estimated by the train set. A precise estimation is not necessary, but a rough estimation of the order of magnitude is enough. When the network has been trained, we can easily recover the model associated with the non-normalized variables by exploiting the affine transformation of Sec. 3.1.3.

Normalizing the inputs, the outputs and time is not enough to make also the internal variables (or just the hidden variables in the output-inside-the-state approach) to lay in $[-1,1]$, because of their hidden nature (see again Sec. 2.4). Therefore, at each optimization epoch, if the mean squared 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 by the affine transformation formulae of Sec. 3.1.3.

## 3.2 Discretization of the state equation and of the objective functional

In order to find a numerical approximation of problem (14), we discretize both the state equation (2) and the objective functional $J$. Therefore, we subdivide the time domain $[0,T]$ into a collection of time instants $0 = t_0 < t_1 < \cdots < t_M = T$. For simplicity, we consider the case of constant time step $\Delta t$ (i.e. $t_k = k\Delta t$ for $k = 0,\dots,M$) since the generalization to the varying time step case is straightforward. On the other hand, it will be useful 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.

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 discretize it by means of the forward Euler scheme. Thus, the discrete counterpart of Problem (14) reads:

$$\begin{cases} \displaystyle\min_{\boldsymbol{\mu}\in\mathbb{R}^{N_f},\boldsymbol{\nu}\in\mathbb{R}^{N_g}} & \frac{1}{2}\sum_{j=1}^{N_s}\sum_{k=0}^{M_j-1} |\mathbf{y}_j^k - \mathbf{g}(\mathbf{x}_j^k;\boldsymbol{\nu})|^2\Delta t \\ \text{s.t.} & \mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \Delta t\,\mathbf{f}(\mathbf{x}_j^k,\mathbf{u}_j^k;\boldsymbol{\mu}), \qquad \text{for } k = 0,\dots,M_j-1, \quad j = 1,\dots,N_s \\ & \mathbf{x}_j^0 = \mathbf{x}_0, \quad j = 1,\dots,N_s. \end{cases} \tag{21}$$

In the following we will denote by $\mathcal{J}$ the discretized version of the objective functional $J$.

## 3.3 Optimization algorithm

Problem (21) can be written in the form of the following nonlinear least-squares problem:

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

where $\boldsymbol{\xi} = (\boldsymbol{\mu}^T,\boldsymbol{\nu}^T)^T$ 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 $j = 1,\dots,N_s$, $k = 0,\dots,M_j-1$, $h = 1,\dots,N_y$:

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

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 problem we employ the Levenberg-Marquardt method, which is designed for least-squares problems in the form (3.3) (see e.g. Nocedal and Wright 2006). At each iteration $k$ of the optimization loop, one finds the descent direction $\mathbf{d}^{(k)}$ by solving the following problem:

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

where $\lambda^{(k)} \geq 0$ is a weight. The update of the solution follows the rule $\boldsymbol{\xi}^{(k+1)} = \boldsymbol{\xi}^{(k)} + \gamma^{(k)}\mathbf{d}^{(k)}$, where the step length $\gamma^{(k)}$ is selected by means of line-search in such a way that the Wolfe conditions are fulfilled (see Nocedal and Wright 2006 for details). Specifically, in this work we employ the line-search Algorithm 3.5 of Nocedal and Wright 2006.

The Levenberg-Marquardt 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 ($D^2(\frac{1}{2}\mathbf{r}^T\mathbf{r}) = \nabla^T\mathbf{r}\nabla\mathbf{r} + \sum_j r_j D^2 r_j$, where we denote by $D^2$ the second derivative operator). To retain the advantages of both techniques, the scalar $\lambda^{(k)}$ should be iteratively adapted in order to follow the steepest-descent direction when the solution is far from a minimum and to switch progressively to the Gauss-Newton direction when the solution approaches a minimum. We follow the choice $\lambda^{(k)} = \min\left\{|\mathbf{r}^{(0)}|^2, |\nabla^T\mathbf{r}^{(k)}\mathbf{r}^{(k)}|\right\}$, which ensures, provided that the objective functional is twice continuously differentiable, superlinear convergence for the method (see Grippo and Sciandrone 2011).

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

## 3.4 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 $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Since the map from $\boldsymbol{\mu}$ to the variables $\mathbf{x}_j^k$ is implicitly given by the state equation contained in (21), we employ the Lagrange multipliers method to compute the sensitivities. Generally speaking, suppose that one needs to compute the sensitivity of a quantity $Q$, function of the variables $\mathbf{x}_j^k$, with respect to $\boldsymbol{\mu}$. By introducing a family of Lagrange multipliers $\mathbf{w}_j^k$, the Lagrangian associated to the problem reads:

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

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 $j = 1, \ldots, N_s$:

$$\begin{cases} \mathbf{w}_j^{M_j} &= \frac{\partial Q}{\partial \mathbf{x}_j^{M_j}} \\ \mathbf{w}_j^k &= \mathbf{w}_j^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}_j^{k+1} + \frac{\partial Q}{\partial \mathbf{x}_j^k}, \qquad \text{for } k = 0, \ldots, M_j - 1. \end{cases} \tag{22}$$

Once the dual variables $\mathbf{w}_j^k$ are available, the gradient of $Q$ with respect to the $\boldsymbol{\mu}$ can be obtained as follows:

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

By following this procedure for $Q = r^{j,m,h}$, it turns out that for $j = 1, \dots, N_s$, $m = 0, \dots, M_j - 1$ we have:

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

where

$$\begin{cases}
\mathbf{w}^m = \sqrt{\Delta t}\, \nabla_{\mathbf{x}}^T \mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu})\, \mathbf{e}_h \\
\mathbf{w}^k = \mathbf{w}^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}^{k+1}, \qquad \text{for } k = 0, \dots, m-1.
\end{cases} \tag{23}$$

The line-search algorithm for the determination of the step length $\gamma^{(k)}$ also requires the computation of the gradient with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ 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 computed, the gradient of $\mathcal{J}$ can be recovered as $\left(\nabla^T \mathbf{r}\right) \mathbf{r}$. 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:

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

where

$$\begin{cases}
\mathbf{w}_j^{M_j} = \mathbf{0} \\
\mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}_j^{k+1} \\
\qquad + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu})\, (\mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu}) - \mathbf{y}_j^k), \qquad \text{for } k = 0, \dots, M_j - 1,
\end{cases} \tag{24}$$

which are the discrete counterparts of (17) and (19).

## 4 Numerical Results

In this section we assess the capabilities of the proposed method through three test cases, whose specifications are summarized in Table 1. First, in Sec. 4.1, we consider a toy problem, namely the nonlinear pendulum. Thanks to the modest complexity of this test case, we can perform a sensitivity analysis on the network architecture, highlighting the presence of the Occam's hill (Sec. 2.6). Moreover, thanks to the fact that we are here in a very simple case ($N = 2$), we can directly compare the internal dynamics of the reduced models with that of the HF model, shedding light on the machinery behind the proposed strategy.

Then, to test the proposed method on large-scale problems, we consider an electrical circuit with nonlinear elements, ruled by a nonlinear system of 1000 ODEs (Sec. 4.2) and a parabolic PDE, whose HF model is given by its Finite Element (FE) approximation, featuring thousands of internal variables (Sec.4.3).

| Test case | Name | Model type | $N$ | $N_u$ | $N_y$ |
|---|---|---|---|---|---|
| 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 |

Table 1: Test cases list.

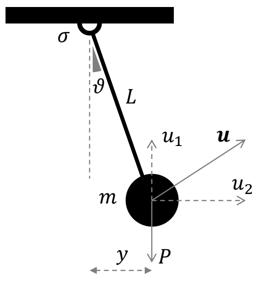


Figure 2: Nonlinear pendulum problem considered in Sec. 4.1

## 4.1 *Test case 1*: Nonlinear pendulum

Consider an object of mass $m$, subject to the weight force $P$, suspended by a weightless inextensible string of length $L$, connected to a fixed support through a hinge subject to viscous damping with constant $\sigma$. Suppose then that the pendulum, starting from its rest 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 ruled by the following ODE:

$$\begin{cases} \ddot{\vartheta}(t) = \frac{1}{Lm}\left(u_2(t)\cos\vartheta(t) + (u_1(t) - P)\sin\vartheta(t)\right) & t \in (0, T] \\ \vartheta(0) = 0,\ \dot{\vartheta}(0) = 0 \end{cases} . \tag{25}$$

where we set the constants values to $m = L = 1$, $P = 2$, $c = 3$. Moreover, we set the lower and upper bounds for the external force $\mathbf{u} \in [-1, 1]^2$. Suppose that we are interested in predicting the horizontal displacement of the mass $\mathbf{y}(t) = L \sin\vartheta(t)$, given the input $\mathbf{u}(t)$. The input-output map given by (25) 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. 3.2), 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 (or, equivalently, we employ a 10 times bigger time step in the evaluation of the objective functional $\mathcal{J}$. In this first test we consider the output-inside-the-state approach.

### 4.1.1 Train, validation and test sets

The optimization of the ANN described in Sec. 3.3 is led by the set of experiments (6) collected from the HF model. The choice of such *train 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 train set itself.

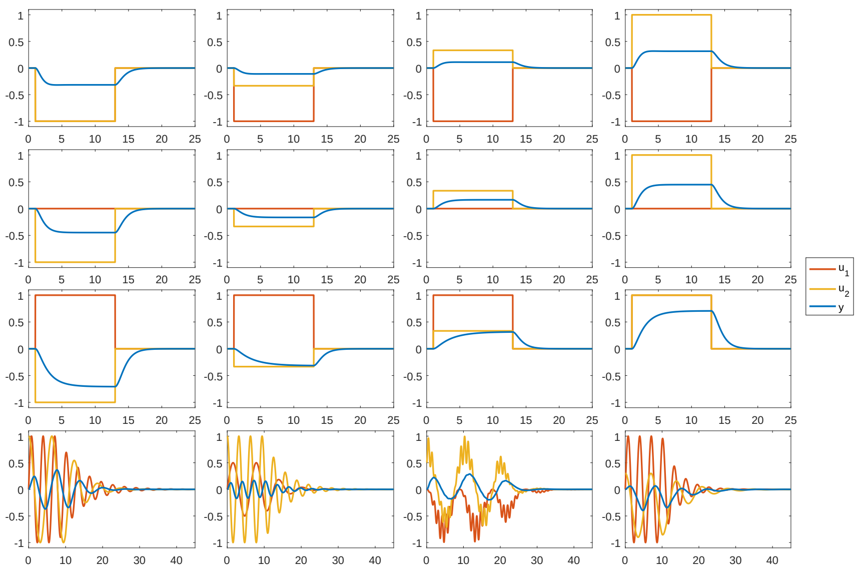


Figure 3: *Test case 1*: Train set.

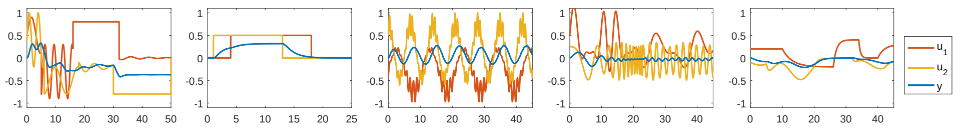


Figure 4: *Test case 1*: Validation set.

In this first example, we intentionally consider a poor train set, such that overfitting is likely to occur, in order to better investigate the sensitivity of the proposed method with respect to $n$ and to the complexity of the network. The train set is represented in Fig. 3. 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 the considered values $\bar{\mathbf{u}}$ span all the set $U = [-1, 1] \times [-1, 1]$) and four samples with oscillating input, obtained by sinusoids with different frequency, amplitude and mean value. The frequency range is chosen in such a way that it covers all the characteristic time scales of the considered system.

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 train 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: as long as the error on the validation set start increasing, we stop the optimization loop since this is a signal of overfitting. In the validation set we put 5 samples, comprising several working regimes of the system (see Fig. 4). 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.

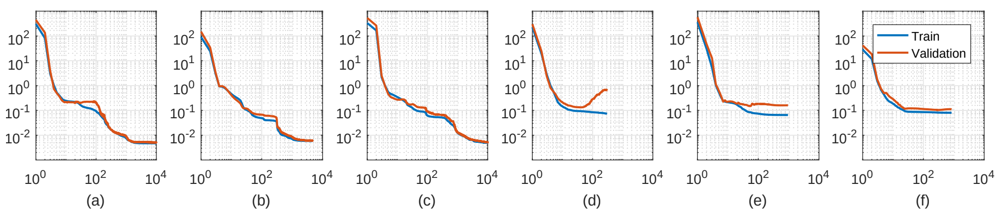


Figure 5: Examples of evolution of the error on the test ($\mathcal{E}_{\text{train}}$) and validation sets ($\mathcal{E}_{\text{val}}$).

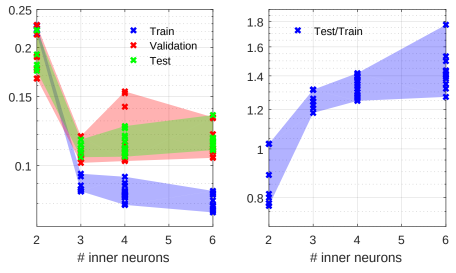
### 4.1.2 Sensitivity analysis of the network complexity

The objective functional $\mathcal{J}$ is highly non-convex and features a landscape characterized by many local minima. As a consequence, the optimized ANN obtained by means of the strategy of Sec. 3.3 may depend on the initialization of the ANN itself. Since we adopt a random 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): two ANNs which are very different may represent the same model. We will go back to this issue in Sec. 4.1.3.

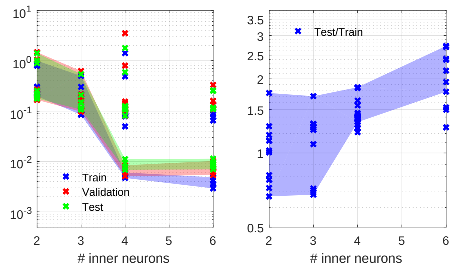
Fig. 5 shows the evolution, through the optimization process, of the error on the test ($\mathcal{E}_{\text{train}}$) and validation sets ($\mathcal{E}_{\text{val}}$) for different ANN initializations. Cases (a)-(c) show "good" outcomes of the optimization process: both errors decrease until a minimum of the functional is reached; a very good correlation between the two errors is observed during the optimization process, sign that the learning process is going well; the final levels of error for the test and validation set are comparable, thus we are not in the presence of overfitting. The remaining cases, in turn, show "bad" outcomes:

- In (d) the optimization proceeds well, until $\mathcal{E}_{\text{val}}$ starts increasing, which is a typical sign a overfitting. Notice that the online evaluation of $\mathcal{E}_{\text{val}}$ allows to detect this phenomenon and to perform early stopping of the optimization process.
- In (e), instead, a more subtle case of overfitting occurs since it originates in the early stages of the optimization, so we do not observe an increase of $\mathcal{E}_{\text{val}}$.
- In (f) finally, even if we are not in presence of significant overfitting, the ANN is not performing well since the final levels of $\mathcal{E}_{\text{train}}$ and $\mathcal{E}_{\text{val}}$ are much higher than the usual values obtained with the same ANN architecture. In this case the optimization problem got stuck into a "bad" local minimum. Globalization strategies, such as Simulated Annealing (see e.g. Press et al. 1986), can be addressed to handle with this issue, but this is out of the scope of the present work.

In the following we perform a sensitivity analysis of the performance of the proposed method w.r.t. the complexity of the network, i.e. the number of neurons, which we denote by $N_{\text{neurons}}$. For simplicity, we consider only the case of ANNs with a single hidden layer. Figure 6 shows the dependency of the performance of the proposed strategy on the number of hidden neurons, in the cases $n = 1$ and $n = 2$. Each cross is associated to a single test, while the coloured 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).



(a) $n = 1$



(b) $n = 2$

Figure 6: *Test case 1*: error versus number of 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 the tests got stuck into a "bad" local minimum). Coloured regions represent the areas spanned by the tests, excluding the tests got stuck into a "bad" local minimum.

Consider first the case $n = 1$. By switching form $N_{\text{neurons}} = 2$ 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 cross the Occam's hill (see Sec. 2.6) and, even if $\mathcal{E}_{\text{train}}$ keep decreasing, $\mathcal{E}_{\text{test}}$ and $\mathcal{E}_{\text{val}}$ start increasing. This phenomenon is also evident from the right graph, showing the ratio $\mathcal{E}_{\text{test}}/\mathcal{E}_{\text{train}}$ increasing with the network complexity. Even if the issue of designing the train set falls beyond the purposes of the present work, we notice that with a richer train set the top of the Occam's hill would probably move towards higher values of $N_{\text{neurons}}$, 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 internal variable in the system, w.r.t. the case $n = 1$, translates in a much higher capacity of the model to faithfully reproduce the input-output model given by the HF model, as expected according to Remark 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 train error decreasing, while the validation an test errors slightly increasing). We notice also the occurrence of tests got stuck in "bad" local minima (signalled in Fig. 6b by the presence of crosses outside the coloured 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 $n = 2$, namely where the top of the Occam's hill is located in the two cases. Figure 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 learned?

The feasibility of the reduction method proposed in this work is strictly related to the possibility of faithfully representing the state of the HF model (which we denote by

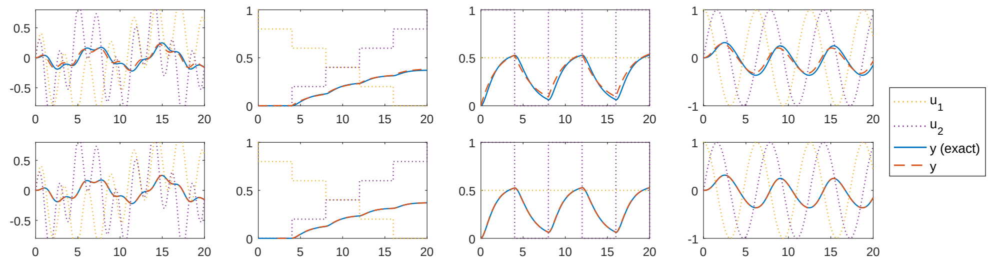


Figure 7: *Test case 1*: comparison of the exact solution (blue line) and the solution obtained with the ANN model (red line) in four different test cases. First row: one-variable model; Second column: two-variables model.

$\mathbf{X}(t) \in \mathbb{R}^N$) by means of a lower-dimensional state $\mathbf{x}(t) \in \mathbb{R}^n$. If this 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 $\frac{d}{dt}\mathbf{x}(t)$).

As mentioned before, the reduced model implicitly defines a map from the HF model state $\mathbf{X}(t)$ to the reduced model state $\mathbf{x}(t)$. However, this map is not explicitly computed. Moreover, as noticed in Remark 1, a model ruled by an ODE is invariant with respect to invertible transformations of its internal state $\widehat{\mathbf{x}} = \mathbf{h}(\mathbf{x})$. For these reasons, it is usually hard to give a physical interpretation to the internal variables of the reduced model.

Consider the considered in this section, for $n = N = 2$. This case may not be interesting in MOR applications, but it helps to shed light on the machinery behind the proposed method. Indeed, thanks to the choice $g(\mathbf{x}) = \boldsymbol{\pi}^{N_y}$, we have, in principle, $x_1(t) = L \sin \vartheta(t)$. So 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 $\vartheta$ (notice that, in the range of motion of the system, the relationship is invertible). Therefore, we expect that the second variable of the reduced model $x_2$ provides the missing information to reconstruct $\dot{\vartheta}$. This does not necessarily entail that $\dot{\vartheta}$ is a function of $x_2$, but in general we have that $\dot{\vartheta}$ is a function of *both* $x_1$ and $x_2$. If this is true, then we have that when a trajectory in the $(\vartheta, \dot{\vartheta})$ plane self-intersects, the same should happen for the trajectory in the $(x_1, x_2)$ plane. In other words, all the trajectories of the obtained models, corresponding to the a given input $\mathbf{u}$, should be homotopic to the corresponding trajectory of the HF model.

To validate the above considerations, we consider the oscillating input also employed in the first column of Fig. 7 (i.e. $u_1 = \sin(t)\cos(1.3\,t)$, $u_2 = \cos(1.8\,t)\sin(t)$), so that the trajectory performs some loops in the phase space, and we compare the trajectories of the HF model (grey background in Fig. 8) with those of four different ANN models (white background in Fig. 8), obtained by running the proposed method starting from four different random initializations. In the first line of Fig. 8 we show the time evolution of the two internal variables of the systems. By the figure it is evident 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, 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 $t_0, t_1, \ldots$ in two vectors, which we denote by

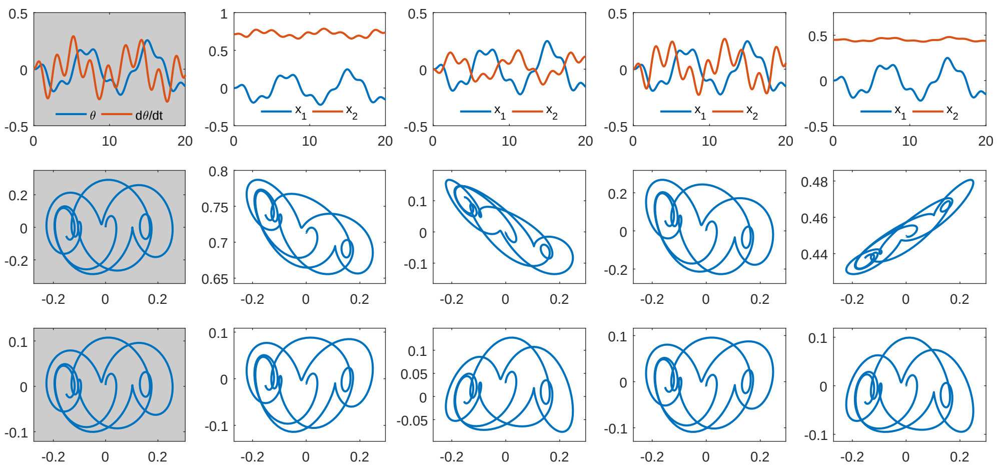


Figure 8: Results of the test case considered in Sec. 4.1.3. Each column refers to a different model (first column: HF model; other columns: four different ANN models). First line: time evolution of the two internal variables of the system (first column: $\vartheta$ and $\dot{\vartheta}$; other columns: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

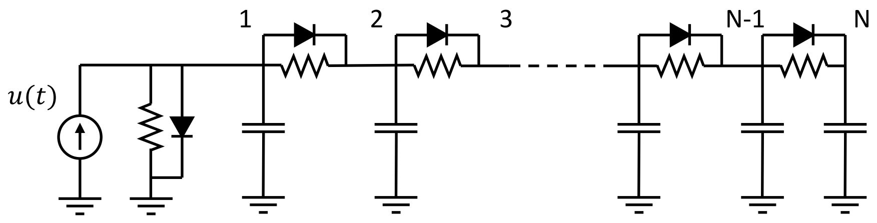


Figure 9: Scheme of the nonlinear transmission line circuit considered in Sec. 4.2

$\mathbf{z}_1$ and $\mathbf{z}_2$, respectively. Then, we centre the vectors ($\mathbf{z}'_j = \mathbf{z}_j - \bar{\mathbf{z}}_j$, for $j = 1, 2$, where $\bar{\mathbf{z}}$ denotes the mean value of $\mathbf{z}$). Then, 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

To assess the capability of the proposed method 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 e.g. Chen and White 2000; Rewieński and White 2001). The transmission line circuit is an electrical network consisting in a current source, $N = 1000$ unitary resistors, $N$ unitary capacitors and $N$ nonlinear diods with low $i = e^{40\,v} - 1$. The input of the model is the current source $u(t)$, taking values in $U = [0, 1]$, and the output is the voltage at the first node $v_1(t)$.

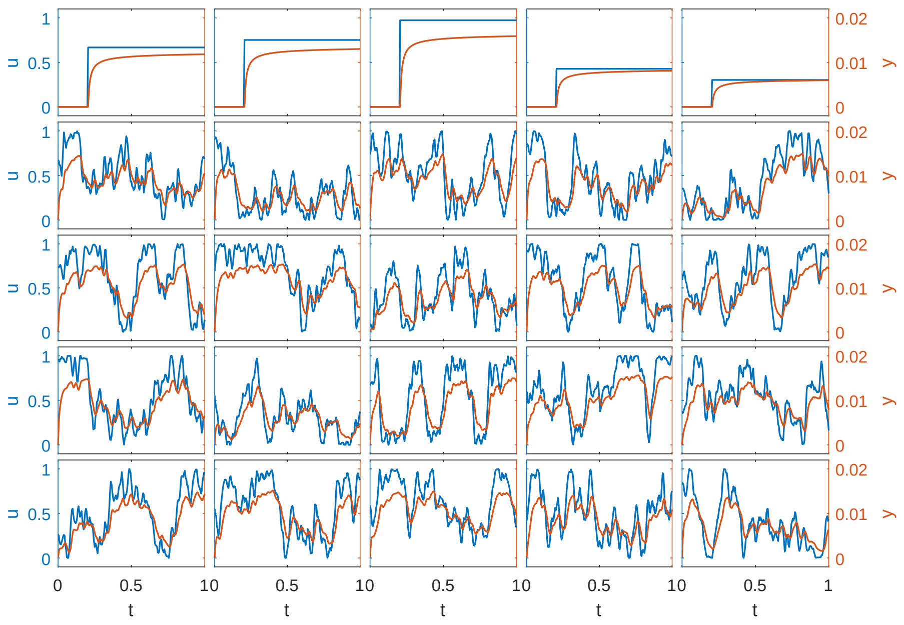


Figure 10: *Test case 2*: train set.

Denoting by $v_i$ the voltage at the $i$-th node, the HF model reads as follows:

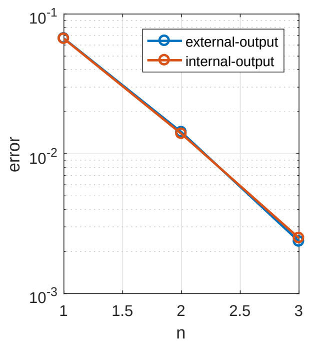
$$\begin{cases} \dot{v}_1(t) = -2\,v_1(t) + v_2(t) + 2 - e^{40\,v_1(t)} - e^{40(v_1(t)-v_2(t))} + u(t) \\ \dot{v}_i(t) = -2\,v_i(t) + v_{i-1}(t) + v_{i+1}(t) + e^{40(v_{i-1}(t)-v_i(t))} - e^{40(v_i(t)-v_{i+1}(t))}, \\ \hfill 2 \le i \le N-1 \\ \dot{v}_N(t) = -v_N(t) + v_{N-1}(t) - 1 + e^{40(v_{N-1}(t)-v_N(t))} \end{cases} \tag{26}$$

supported by the initial condition $v_1(0) = v_2(0) = \cdots = v_N(0) = 0$.

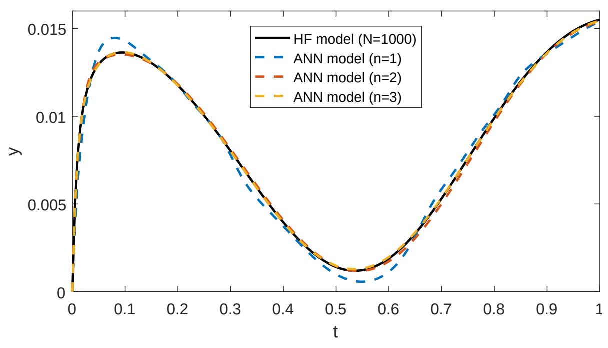
We consider the train set represented in Fig. 10, consisting in 5 step responses and 20 randomly generated input signals, each of 1 s duration. In this test case we consider both the output-inside-the-state and the output-outside-the-state approach, for $n = 1, 2, 3$, and we compare the results. In all the considered 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 $n \ge N_y$, models in the form (2) might be rewritten in the form (13). Even if this is not always valid (a technical hypothesis must be fulfilled), it is reasonable 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 approach to provide similar results that 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 approach 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 Remark 2, the error $\mathcal{E}_{\text{test}}$ decreases as $n$ increases,

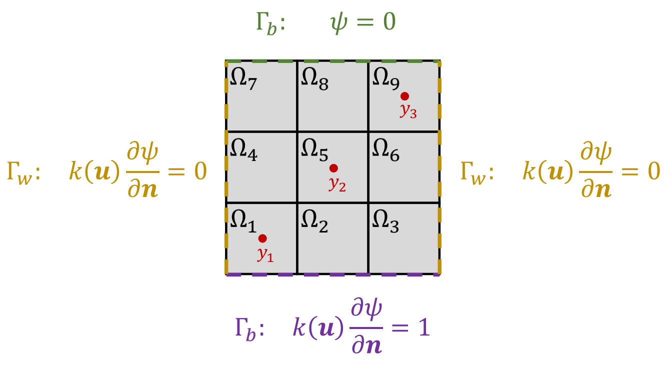


(a) $\mathcal{E}_{\text{test}}$ vs $n$



(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.

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



$\Gamma_b$: $\psi = 0$
$\Omega_7$
$\Omega_8$
$\Omega_9$
$y_3$
$\Omega_4$
$\Omega_5$
$\Omega_6$
$y_2$
$\Omega_1$
$\Omega_2$
$\Omega_3$
$y_1$
$\Gamma_w$: $k(\boldsymbol{u})\frac{\partial\psi}{\partial\boldsymbol{n}} = 0$
$\Gamma_w$: $k(\boldsymbol{u})\frac{\partial\psi}{\partial\boldsymbol{n}} = 0$
$\Gamma_b$: $k(\boldsymbol{u})\frac{\partial\psi}{\partial\boldsymbol{n}} = 1$

Figure 12: *Test case 3*: domain and boundary conditions.

reaching, for $n = 3$, a remarkably good approximation level (nearly $2.5 \cdot 10^{-3}$). In Fig. 11b we compare in the time-domain the response of the HF model with that of the three reduced models obtained with the output-inside-the-state approach.

## 4.3 *Test case 3*: Heat equation

In this section we consider the application of the proposed method to the reduction of a parabolic PDE problem. We consider the unsteady version of the elliptic problem considered in Manzoni, Pagani, and Lassila 2016, where its Reduced Basis (RB) reduction has been considered. Consider the spatial domain $\Omega = (0, 1.5)^2$, partitioned in 9 subdomains $\Omega_i$, for $i = 1, \ldots, 9$, as in Fig. 12. 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 $\Omega_i$. The boundary of the domain $\Omega$ is partitioned into the top border $\Gamma_t$, in contact with a heat reservoir with zero temperature, the

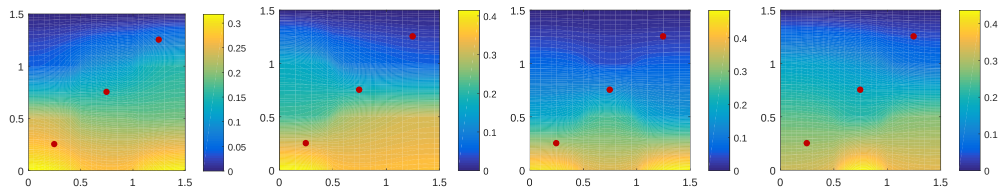


Figure 13: *Test case 3*: 4 examples of snapshots obtained at different times with different inputs.

bottom border $\Gamma_b$, with a constant inward unitary heat flux, and in the wall borders $\Gamma_w$, characterized by no-flux boundary conditions. The time evolution of the spatially distributed temperature $\psi(\mathbf{x},t)$ in the domain $\Omega$ is thus ruled by the heat equation:

$$\begin{cases} \frac{\partial}{\partial t}\psi(\mathbf{x},t) - \operatorname{div}\left(k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\right) & \text{in } \Omega, \text{ for } t>0 \\ k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\cdot\mathbf{n} = 0 & \text{on } \Gamma_w, \text{ for } t>0 \\ k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\cdot\mathbf{n} = 1 & \text{on } \Gamma_b, \text{ for } t>0 \\ \psi(\mathbf{x},t) = 0 & \text{on } \Gamma_t, \text{ for } t>0 \\ \psi(\mathbf{x},0) = 0 & \text{on } \Omega. \end{cases} \tag{27}$$

Consider then three probes, located in the centre 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 high-fidelity solution of (27), we consider the P2 Finite Element approximation on a 30 by 30 uniform square elements grid, employing the Forward Euler scheme, with $\Delta t = 10^{-2}$, for the time discretization, implemented in the MATLAB finite element library `feamat` (Pegolotti 2018). The HF model has dimension $N = 3721$.

In this test case we compare the results obtained with the proposed method with those obtained with a popular MOR method in the field of PDEs, namely the RB method. For the ANN-based reduction method we consider the cases $n = 1$, 2 and 3. In the first two cases we employ the output-outside-the-state approach, being $N_y > n$, while in the third case we employ the output-inside-the-state one. In each case we use single hidden layer ANNs, with 12 hidden neurons for $\mathbf{f}$ and (if necessary) 3 hidden neurons for $\mathbf{g}$. Figure 14 reports a subset of the train set, comprising 10 steady-state responses of duration 0.4 s, obtained by sampling the input space $U = [10, 100]^9$ by means of Latin Hypercube Sampling (see McKay, Beckman, and Conover 2000), and 50 random transients of duration 1 s. As to the RB method, we build the basis by POD of the snapshot matrix obtained by sampling every 0.1 s the same train 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. 15 we compare the results, obtained with 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 and 10 random input signals of duration 10 s. The purpose of the latter is to asses the capability of the proposed method to approximate the evolution of the HF model also for longer time horizons than the ones used in the training stage. As it is shown by Fig. 15a, for a given model size $n$, the ANN based reduced model performs better than the RB one: for $n = 3$ its 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 greater with the proposed method too. Even if this result is implementation

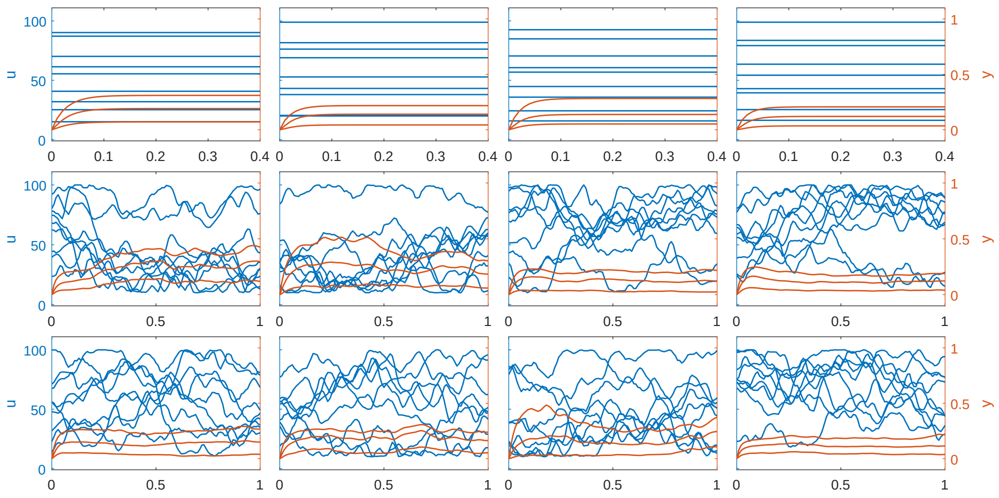


Figure 14: *Test case 3*: a subset of the trainset.

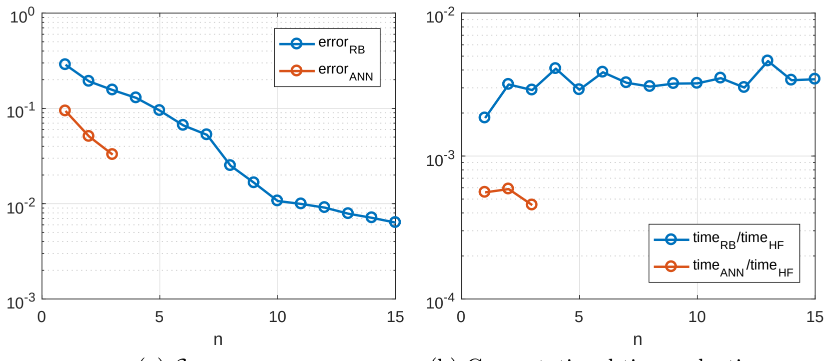
dependent, it can be ascribed to the fact that the online phase of the RB method requires, at each time step, before the actual time advance, the assembling of the system matrices and right-hand side as affine combination, weighted by the current value of $\mathbf{u}(t)$, of precomputed matrices and vectors.

The major drawback of the proposed method lies in the offline phase, which requires the training of one or two ANNs, while for the RB method we just need to compute the SVD decomposition of the snapshot matrix and to project the system matrices and right-hand side. Moreover, the computational complexity of the training rapidly grows with $n$ and for high $n$ big train sets are needed to avoid overfitting, thus preventing the applicability of the proposed method to large $n$. However, from the presented results it seems that with the proposed method 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.

Finally, we notice that the this comparison has been carried out in the most favourable case for RB, namely linear models with affine input dependence and with linear state-output dependence, while the proposed method 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 method can be applied without modifications (see e.g. the *Test case 2*).

## 5 Conclusions and perspectives

We have proposed a data-driven nonlinear MOR technique, based on ANNs. We formulate the model reduction problem as a best-approximation or maximum-likelihood problem, where we look for the most suitable representation of the HF model into



(a) $\mathcal{E}_{\text{test}}$ vs $n$. (b) Computational time reduction vs $n$.

Figure 15: *Test case 3*: comparison between the proposed method 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 $n$.

a class of simpler models. The candidate models are represented by means an ANN (or by two of them, in the output-outside-the-state approach), which is fed by input-output pairs originated from the HF system. Thanks to this formulation, is it possible to compute the sensitivity of the model error with respect to the parameters of the ANNs, and to exploit standard optimization algorithms to make the ANN learn the underlying physics of the system. The proposed method can be applied to a wide class of models, satisfying some minimal requirements (see Sec. 2.1).

We have assessed the effectiveness of the proposed approach through a simple case study (namely the nonlinear pendulum) and through the reduction of two large-scale problems (a nonlinear system of ODEs and a parabolic PDE), featuring thousands of variables. In both cases we derived reduced models capable of approximating the HF models with an error of order $10^{-3}$ for the ODEs system and $10^{-2}$ for the PDE, with just 3 internal variables. We have also compared the performances of the proposed method with those 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 method yielded much more accurate reduced models than the RB method, also featuring a cheaper online cost in terms of computational time.

Still, in this work we have not investigated some aspects, where there is room for improvement of the proposed technique. In particular, we have not dealt with the issue of the train set design, where one can think to employ automatic procedures to select an optimized train set, in a similar manner to the selection of snapshots in the RB framework (see e.g. Quarteroni, Manzoni, and Negri 2015). Moreover, globalization techniques can be taken into account to deal with the problem of local minima in the optimization process. Furthermore, the offline phase may be considerably optimized with the application of more advanced learning techniques than the one considered in this work, such as stochastic selection of the train set (SGD, see e.g. Bottou 2010), dropout (Srivastava et al. 2014), batch normalization (**ioffe2015batcho**) and progressive layers freezing (Brock et al. 2017).

Finally, we notice that the formulation of the MOR problem in terms of minimization problem allows to easily incorporate into the learning stage some a priori knowledge on the HF model, by incorporating into the cost functional suitable penalization terms.

## References

Alexandrov, Natalia M, Robert Michael Lewis, Clyde R Gumbert, Lawrence L Green, and Perry A Newman (2001). "Approximation and model management in aerodynamic optimization with variable-fidelity models". In: *Journal of Aircraft* 38.6, pp. 1093–1101.

Antoulas, Athanasios C (2005). *Approximation of large-scale dynamical systems*. Vol. 6. Siam.

Antoulas, Athanasios C, Ion Victor Gosea, and Antonio Cosmin Ionita (2016). "Model reduction of bilinear systems in the Loewner framework". In: *SIAM Journal on Scientific Computing* 38.5, B889–B916.

Antoulas, Athanasios C, Danny C Sorensen, and Serkan Gugercin (2000). *A survey of model reduction methods for large-scale systems*. Tech. rep.

Bai, Zhaojun (2002). "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems". In: *Applied numerical mathematics* 43.1-2, pp. 9–44.

Barrault, Maxime, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera (2004). "An empirical interpolationmethod: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus Mathematique* 339.9, pp. 667–672.

Baur, Ulrike, Christopher Beattie, Peter Benner, and Serkan Gugercin (2011). "Interpolatory projection methods for parameterized model reduction". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2489–2518.

Benner, Peter, Serkan Gugercin, and Karen Willcox (2015). "A survey of projection-based model reduction methods for parametric dynamical systems". In: *SIAM review* 57.4, pp. 483–531.

Benner, Peter, Volker Mehrmann, and Danny C Sorensen (2005). *Dimension reduction of large-scale systems*. Vol. 35. Springer.

Bottou, Léon (2010). "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, pp. 177–186.

Brock, Andrew, Theodore Lim, James M Ritchie, and Nick Weston (2017). "Freeze-Out: Accelerate Training by Progressively Freezing Layers". In: *arXiv preprint arXiv:1706.04983*.

Brunton, Steven L, Joshua L Proctor, and J Nathan Kutz (2016). "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences*, p. 201517384.

Casella, George and Roger L Berger (2002). *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA.

Chaturantabut, Saifon and Danny C Sorensen (2010). "Nonlinear model reduction via discrete empirical interpolation". In: *SIAM Journal on Scientific Computing* 32.5, pp. 2737–2764.

Chen, Y, J White, et al. (2000). "A quadratic method for nonlinear model order reduction". In:

Cybenko, George (1988). *Continuous valued neural networks with two hidden layers are sufficient*.

— (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Deschrijver, D and T Dhaene (2005). “Rational modeling of spectral data using orthonormal vector fitting”. In: *9th IEEE Workshop on Signal Propagation on Interconnects*, pp. 111–114.

Deschrijver, Dirk, Bart Haegeman, and Tom Dhaene (2007). “Orthonormal vector fitting: A robust macromodeling tool for rational approximation of frequency domain responses”. In: *IEEE Transactions on advanced packaging* 30.2, pp. 216–225.

Drohmann, Martin, Bernard Haasdonk, and Mario Ohlberger (2012). “Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation”. In: *SIAM Journal on Scientific Computing* 34.2, A937–A969.

Everson, Richard and Lawrence Sirovich (1995). “Karhunen–Loeve procedure for gappy data”. In: *JOSA A* 12.8, pp. 1657–1664.

Fink, JP and WC Rheinboldt (1983). “On the error behavior of the reduced basis technique for nonlinear finite element approximations”. In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 63.1, pp. 21–28.

Freund, Roland W (2003). “Model reduction methods based on Krylov subspaces”. In: *Acta Numerica* 12, pp. 267–319.

Gosea, Ion Victor and Athanasios C Antoulas (2015). “Model reduction of linear and nonlinear systems in the Loewner framework: A summary”. In: *Control Conference (ECC), 2015 European.* IEEE, pp. 345–349.

Grippo, Luigi and Marco Sciandrone (2011). *Metodi di ottimizzazione non vincolata.* Springer Science & Business Media.

Gu, Chenjie (2011). “QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.9, pp. 1307–1320.

Hackbusch, Wolfgang (1979). “On the fast solving of parabolic boundary control problems”. In: *SIAM journal on control and optimization* 17.2, pp. 231–244.

Hernandez, Andres F and Martha Grover Gallivan (2008). “An exploratory study of discrete time state-space models using kriging”. In: *American Control Conference, 2008.* IEEE, pp. 3993–3998.

Hesthaven, Jan S, Gianluigi Rozza, Benjamin Stamm, et al. (2016). *Certified reduced basis methods for parametrized partial differential equations.* Springer.

Hotelling, Harold (1933). “Analysis of a complex of statistical variables into principal components.” In: *Journal of educational psychology* 24.6, p. 417.

Keesman, Karel J (2011). *System identification: an introduction.* Springer Science & Business Media.

Krige, Daniel G (1951). “A statistical approach to some basic mine valuation problems on the Witwatersrand”. In: *Journal of the Southern African Institute of Mining and Metallurgy* 52.6, pp. 119–139.

Lefteriu, Sanda and Athanasios C Antoulas (2010). “A new approach to modeling multiport systems from frequency-domain data”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29.1, pp. 14–27.

Ljung, Lennart (1998). “System identification”. In: *Signal analysis and prediction.* Springer, pp. 163–173.

Loeve, Michel (1978). “Probability theory, vol. ii”. In: *Graduate texts in mathematics* 46, pp. 0–387.

Löwner, Karl (1934). “Über monotone matrixfunktionen”. In: *Mathematische Zeitschrift* 38.1, pp. 177–216.

Maday, Yvon, Ngoc Cuong Nguyen, Anthony T Patera, and George SH Pau (2007). “A general, multipurpose interpolation procedure: the magic points”. In:

Manzoni, Andrea, Stefano Pagani, and Toni Lassila (2016). “Accurate solution of Bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models”. In: *SIAM/ASA Journal on Uncertainty Quantification* 4.1, pp. 380–412.

Mayo, AJ and AC Antoulas (2007). “A framework for the solution of the generalized realization problem”. In: *Linear algebra and its applications* 425.2-3, pp. 634–662.

McKay, Michael D, Richard J Beckman, and William J Conover (2000). “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code”. In: *Technometrics* 42.1, pp. 55–61.

Menafoglio, Alessandra, Piercesare Secchi, Matilde Dalla Rosa, et al. (2013). “A Universal Kriging predictor for spatially dependent functional data of a Hilbert Space”. In: *Electronic Journal of Statistics* 7, pp. 2209–2240.

Moore, Bruce (1981). “Principal component analysis in linear systems: Controllability, observability, and model reduction”. In: *IEEE transactions on automatic control* 26.1, pp. 17–32.

Narendra, Kumpati S and Kannan Parthasarathy (1990). “Identification and control of dynamical systems using neural networks”. In: *IEEE Transactions on neural networks* 1.1, pp. 4–27.

— (1992). “Neural networks and dynamical systems”. In: *International Journal of Approximate Reasoning* 6.2, pp. 109–131.

Negri, Federico, Andrea Manzoni, and David Amsallem (2015). “Efficient model reduction of parametrized systems by matrix discrete empirical interpolation”. In: *Journal of Computational Physics* 303, pp. 431–454.

Nelles, Oliver (2013). *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media.

Nocedal, J. and S. Wright (2006). *Numerical optimization*. Springer Science & Business Media.

Pearson, Karl (1901). “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Pegolotti, Luca (2018). *feamat*. URL: `https://github.com/lucapegolotti/feamat`.

Peterson, Janet S (1989). “The reduced basis method for incompressible viscous flow calculations”. In: *SIAM Journal on Scientific and Statistical Computing* 10.4, pp. 777–786.

Press, William H, Brian P Flannery, Saul A Teukolsky, and William T Vetterling (1986). *Numerical recipes: the art of scientific computing*. Cambridge Univ. Press, New York.

Quarteroni, Alfio, Andrea Manzoni, and Federico Negri (2015). *Reduced basis methods for partial differential equations: an introduction*. Vol. 92. Springer.

Quarteroni, Alfio and Alessandro Veneziani (2003). “Analysis of a geometrical multiscale model based on the coupling of ODE and PDE for blood flow simulations”. In: *Multiscale Modeling & Simulation* 1.2, pp. 173–195.

Raissi, Maziar and George Em Karniadakis (2018). “Hidden physics models: Machine learning of nonlinear partial differential equations”. In: *Journal of Computational Physics* 357, pp. 125–141.

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2017a). “Machine learning of linear differential equations using Gaussian processes”. In: *Journal of Computational Physics* 348, pp. 683–693.

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2017b). “Physics Informed Deep Learning (Part I): Data-driven solutions of nonlinear partial differential equations”. In: *arXiv preprint arXiv:1711.10561*.

— (2017c). “Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations”. In: *arXiv preprint arXiv:1711.10566*.

Rasmussen, Carl Edward (2004). “Gaussian processes in machine learning”. In: *Advanced lectures on machine learning*. Springer, pp. 63–71.

Rasmussen, Carl Edward and Zoubin Ghahramani (2001). “Occam’s razor”. In: *Advances in neural information processing systems*, pp. 294–300.

Regazzoni, Francesco, Luca Dedè, and Alfio Quarteroni (2018). “Active contraction of cardiac cells: a reduced model for sarcomere dynamics with cooperative interactions”. In: *Biomechanics and modeling in mechanobiology*, pp. 1–24.

Rewieński, Michał and Jacob White (2001). “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, pp. 252–257.

Sirovich, Lawrence (1987). “Turbulence and the dynamics of coherent structures. I. Coherent structures”. In: *Quarterly of applied mathematics* 45.3, pp. 561–571.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Yegnanarayana, B (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.