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

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, École Polytechnique Fédérale de Lausanne,
Av. Piccard, CH-1015 Lausanne, Switzerland (*Professor Emeritus*)

Abstract

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

Keywords Machine Learning, Differential equations, Model Order Reduction, System Identification, Artificial Neural Networks, Data-driven modeling

1 Introduction

The numerical simulation of time-dependent mathematical models is often needed in applied sciences. The increasing demand of more complex and reliable mathematical models may sometimes lead to an unbearable demand for computational resources, either in terms of computational power or memory storage. Moreover, in many applications, 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 dynamical model needs to be solved virtually in any point of a computational domain. In several contexts, such as computational medicine, complex mathematical models, although precise and reliable, may be useless for practical purposes if these cannot be solved nearly in real-time.

This strongly motivates the development of *reduced models*, that is computationally tractable, lower dimensional mathematical models which can be solved with a smaller effort (both in terms of time and computational resources), yet reproducing with a good approximation the results of the *high-fidelity* (HF) model (Antoulas, Sorensen, and Gugercin 2000; Benner, Mehrmann, and Sorensen 2005; Quarteroni, Manzoni, and Negri 2015; Quarteroni and Rozza 2014).

In this paper, we focus on time-invariant systems, whose behaviour is determined by a time-dependent input $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ and endowed with an output $\mathbf{y}(t) \in \mathbb{R}^{N_y}$. Let us consider the following

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

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

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

MOR can be *model-based* or *data-driven* (Antoulas 2005; Benner, Gugercin, and Willcox 2015; Peherstorfer, Gugercin, and Willcox 2017; Peherstorfer and Willcox 2015a,b). With the first strategy, Eq. (1) itself is the starting point to derive its reduced version (2). With data-driven approaches instead, the reduced model is built upon a collection of input-output pairs, through which the dynamics of the system is inferred. The advantages of model-based approaches are that, often, the reduced model inherits structural properties (e.g. stability) from the HF model; moreover, the underlying HF system structure provides the base for deriving error estimates, hence error certification. Unfortunately, an equation in the form of (1) is not always available to express the dynamics of the HF model, which may be accessible only through input-output data. This is the case, for instance, when the system dynamics is available only through experimental measurements, either in the time domain or in the frequency domain (i.e. through samples of the transfer function, in the case of a linear system), or when the HF system is accessible through the simulations of a black-box 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

[1] 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 $X_{N+1}(0) = 0$. Moreover, Eq. (1) also comprises parametric differential equations, that is the case when $\mathbf{u}(t)$ is constant in time and can thus be regarded as a parameter.

$\mathbf{X}(t) \simeq \mathbf{V}\mathbf{x}(t)$ and the HF model equation is projected in the Galerkin (or Petrov-Galerkin) sense, by left multiplying it by $\mathbf{V}^T$ (or by another matrix $\mathbf{W}^T$, respectively):

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

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, up to a desired order, the first few moments 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 dates 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 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 represents one of the two classical forms of the Reduced Basis (RB) method, the other one being based on the greedy algorithm to generate the snapshots (see e.g. Fink and Rheinboldt 1983; Hesthaven, Rozza, and Stamm 2016; Peterson 1989; Prud'Homme et al. 2002; 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).

In Lee and Carlberg 2018 the authors proposed a projection-based MOR technique, where the system is projected into a nonlinear manifold, which is computed by means of convolutional autoencoders from deep learning.

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 methods in 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 matrices and arrays 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). In Peherstorfer, Gugercin, and Willcox 2017 the Loewner framework has been extended to derive reduced models from time-domain data, instead of frequency-domain data. Although the Loewner framework only applies to linear models, it has recently been extended to bilinear (i.e. 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 made possible 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 enhance numerical stability in the identification of the approximation coefficients (Deschrijver and Dhaene 2005; Deschrijver, Haegeman, and Dhaene 2007).

Kriging, originally developed in the field of geosciences (Krige 1951), also known as Gaussian Process (GP) regression, is widely used to perform MOR for nonlinear models in the steady-state case (Menafoglio, Secchi, and Dalla Rosa 2013; Rasmussen 2004). It 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 technique, under the assumption that $\mathbf{f}$ depends just on a few combinations of the inputs (such as linear combinations, products, and trigonometric functions), seeks a sparse solution for the coefficients of a predetermined collection of linear and nonlinear terms.

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

The method proposed in Guo and Hesthaven 2019 represents an intermediate approach between model-based and data-driven ones, by combining the RB method with GP regression. In particular the authors build a reduced basis by POD of a collection of snapshots collected by the HF solution of time-dependent parametric differential equations; then, by GP regression, they approximate the map from the parameters and the time instant to the reduced basis coefficients of the solution. We notice that this method, however, is restricted to parametric differential equations (i.e. when $\mathbf{u}(t)$ is constant in time) and cannot be easily extended to the case of models with time-dependent inputs.

1.3 Learning models from data

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 families 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 *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- dynamics 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 of 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 phase, once an offline training phase 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, 2019), 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.

In Raissi, Perdikaris, and Karniadakis 2018 the authors make use of ANNs to perform data-driven discovery of nonlinear dynamical systems. In particular, they train the ANN to minimize the residuals of a given multi-step time-stepping scheme (such as Adams-Bashforth or BDF schemes) on a collection of available snapshots of the system state $\{\mathbf{X}(t^k)\}_{k=1}^{M}$. However, this approach, like DMK and SINDy techniques, requires the full-order state $\mathbf{X}$ to be accessible and, moreover, does not perform any dimensionality reduction of the state space.

In San and Maulik 2018 the authors have used ANNs, in combination with projection-based MOR for time-dependent models, to approximate the effect of the discarded modes on the retained ones. In Freno and Carlberg 2018; Trehan, Carlberg, and Durlofsky 2017 the authors have used machine-learning techniques to model the error of projection-based reduced model of parametrized nonlinear dyanmical systems.

1.4 Original contributions and outline

In this work we address the problem of data-driven MOR for nonlinear dynamical systems (which can be interpreted, as 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 through 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.

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

- Building surrogates of time-dependent differential models, which allow for fast evaluations and are suitable for multi-query problems.

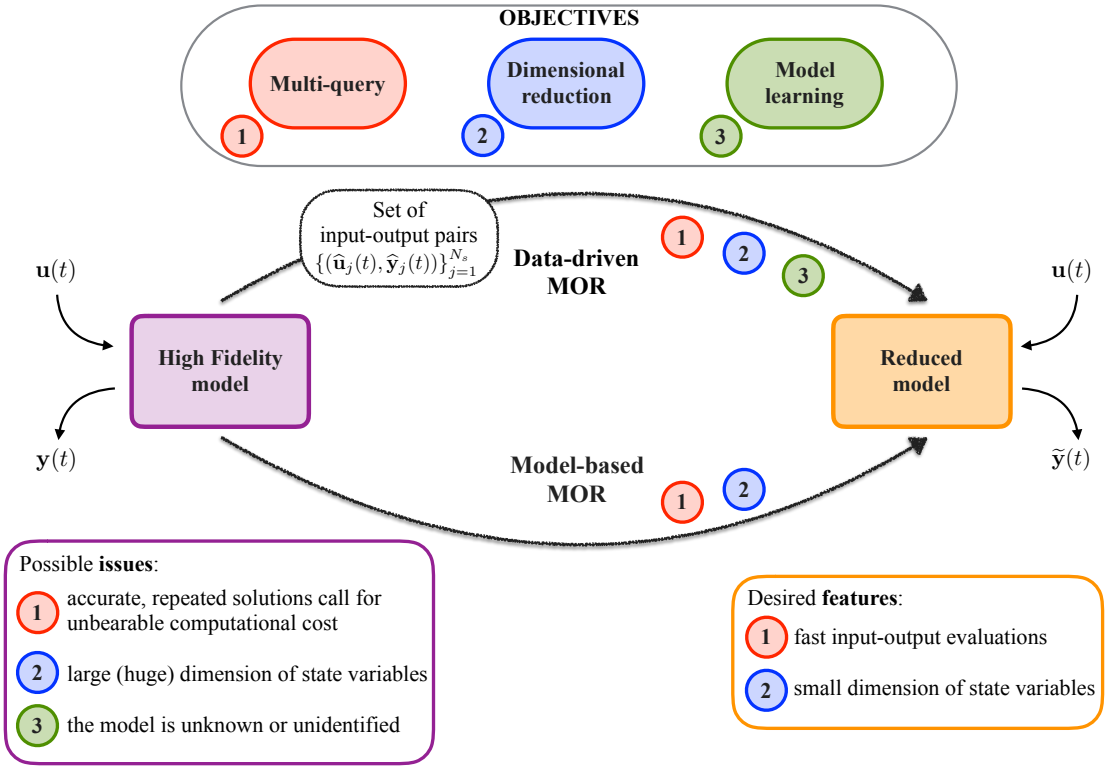


Figure 1: Model-based and data-driven MOR. The HF model can have three undesired features: ① it may be computationally demanding, ② it can have a high state dimension, ③ it may not be accessible. A reduced model can be derived either directly from the HF model (model-based MOR), or from input-output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

- Reducing the dimensionality of the state variables of time-dependent differential models (e.g. when a mathematical model needs to be solved virtually in each point of a computational domain and the overall memory storage must be contained).
- Learning mathematical models from available input-output pairs.

This paper is structured as follows. In Sec. 2 we present our strategy, by rephrasing the model reduction problem in terms of an optimization problem, for which we define the objective functional. Then, in Sec. 3, we present the strategy employed to numerically find a solution of the optimization problem. In particular, we show: (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* (in the way this is meant in this paper): an object which maps time-dependent inputs into time-dependent outputs.

2.1 The dynamical model

By *model* we intend a general framework including either a physical model (like a natural phenomenon or an engineering process) or a mathematical or numerical model, which associates a

time-dependent output to a time-dependent input. We consider a limited time interval $[0, T]$ and we denote by $U \subset \mathbb{R}^{N_u}$ and by $Y \subset \mathbb{R}^{N_y}$ the sets where the input $\mathbf{u} : [0, T] \to U$ and the output $\mathbf{y} : [0, T] \to Y$ take values, respectively. We denote by the term *experiment* the action of inputing $\mathbf{u}(t)$ to the model and recording the corresponding output $\mathbf{y}(t)$ and by the term *sample* we refer to the couple $(\mathbf{u}, \mathbf{y})$. For the sake of abstraction, we make the following minimal assumptions on the model:

(A1) **Time invariance**: by denoting with $\mathbf{y}(t)$ the output obtained by starting at time t_0 an experiment with input $\mathbf{u}(t)$, the output of another experiment started at time $t_1 \geq t_0$ with input $\mathbf{u}(t - (t_1 - t_0))$ is $\mathbf{y}(t - (t_1 - t_0))$. Hence in the following we will consider, without loss of generality, each experiment starting from the initial time $t = t_0 = 0$.

(A2) **Existence of an initial state**: at time $t = 0$, for each experiment, the model is in the same initial state, that is it always responds in the same way to a prescribed input. Otherwise, the map $\mathbf{u} \mapsto \mathbf{y}$ would not be well defined.

(A3) **Causality principle**: The input-output relationship must be consistent with the arrow of time, that is the output of the model 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 on the state of the system at the same time, but not (directly) on the input at time t. As a consequence, thanks to (A2), the output at time $t = 0$ will be the same for each experiment. Notice that this assumption could be removed, by allowing $\mathbf{g}$ in (2) to depend also on $\mathbf{u}(t)$. However, for the sake of simplicity, we will not consider this case in this work.

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

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

where $\boldsymbol{\varphi}\mathbf{u}_1$ 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 $\boldsymbol{\varphi}\mathbf{u}_1$ are functions of time) and $(\boldsymbol{\varphi}\mathbf{u}_1)|_{[0,s]}$ denotes the restriction of the output $\mathbf{y}_1$ to the time interval $[0, s]$. On the other hand, assumption (A4) entails

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

where $(\boldsymbol{\varphi}\mathbf{u})(0)$ 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

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

endowed with the norm of the supremum:

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

2.2 Building a reduced model

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

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

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

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

where the objective functional is given by

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

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 the choice of $\widehat{\Phi} \subseteq \Phi$, the subset of candidate models. A possible approach is that of directly approximating 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 $\{\mathbf{u}(t_0), \mathbf{u}(t_1), \ldots, \mathbf{u}(t_M)\}$ of values associated to a collection of time instants and returns the corresponding output values $\{\mathbf{y}(t_0), \mathbf{y}(t_1), \ldots, \mathbf{y}(t_M)\}$. 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 Φ, which is based on assumptions (A1)–(A4), we restrict the investigation to input-output maps described by systems of ODEs. This dramatically reduces the size of the problem, as we show in the following.

2.3 Models described by systems of ODEs

We refer now to the specific class of models, in the framework of Sec 2.1, which are governed by a system of ODEs in the form of (2). Such class represents a subset of Φ, as we will show. First, we notice that, given two functions $\mathbf{f} \in \mathcal{F}_n := \{\mathbf{f} \in \mathcal{C}^0(\mathbb{R}^n \times U; \mathbb{R}^n)$, which are Lipschitz continuous in $\mathbf{x}$ uniformly in $\mathbf{u}\}$ and $\mathbf{g} \in \mathcal{G}_n := \mathcal{C}^0(\mathbb{R}^n; Y)$ and a vector $\mathbf{x}_0 \in \mathcal{X}_n \equiv \mathbb{R}^n$ (where the subscript n stands for the number of internal reduced states), the system (2) identifies a unique map from $\mathcal{U}$ to $\mathcal{Y}$. We denote by $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$ 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-Lindelöf theorem, Eq. (2) has a unique solution. Thus, the map $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$ is well defined. Moreover, it is continuous from $\mathcal{U}$ to $\mathcal{Y}$, thus the norm in Φ is finite. 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 state variables grows as n increases:

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

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

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

Moreover, we have the following result.

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

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

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

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

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

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

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

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

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}\left|\mathbf{x}(t)\right|^2\right)&\le L_{\mathbf{f}}\left|\mathbf{x}(t)\right|^2+\left(L_{\mathbf{f}}\left|\mathbf{x}_0\right|+M\right)\left|\mathbf{x}(t)\right|\\ &\le\left(L_{\mathbf{f}}+\frac{1}{2}\right)\left|\mathbf{x}(t)\right|^2+\frac{1}{2}\left(L_{\mathbf{f}}\left|\mathbf{x}_0\right|+M\right)^2.\end{aligned}$$

Using the Gronwall lemma we get:

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

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

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

We define the positive number:

$$\varepsilon''=\min\left\{\varepsilon',\frac{1}{2}R\right\}\left(Te^{(2L_{\mathbf{f}}+1)T}\right)^{-1/2}.$$

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

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

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

By Gronwall inequality we get

$$\left|\mathbf{x}(t)-\widehat{\mathbf{x}}(t)\right|\le\varepsilon''\sqrt{Te^{(2L_{\mathbf{f}}+1)T}}\le\frac{1}{2}R,$$

where the second inequality follows from the definition of ε''. The, we have $|\widehat{\mathbf{x}}(t)|\le|\mathbf{x}(t)-\widehat{\mathbf{x}}(t)|+|\mathbf{x}(t)|\le\frac{3}{2}R$ for $t\in[0,s)$. Since $|\widehat{\mathbf{x}}(s)|=2R$, we reach a contradiction and we have proved that

the trajectory of $\widehat{\mathbf{x}}(t)$ is contained into B_{2R}. Therefore, inequality (13) holds on the whole time interval $[0,T]$ and we conclude:

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

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

$$\begin{aligned} |(\boldsymbol{\varphi}\mathbf{u})(t) - (\widehat{\boldsymbol{\varphi}}\mathbf{u})(t)| &= |\mathbf{g}(\mathbf{x}(t)) - \widehat{\mathbf{g}}(\widehat{\mathbf{x}}(t))| \\ &\leq |\mathbf{g}(\mathbf{x}(t)) - \mathbf{g}(\widehat{\mathbf{x}}(t))| + |\mathbf{g}(\widehat{\mathbf{x}}(t)) - \widehat{\mathbf{g}}(\widehat{\mathbf{x}}(t))| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned}$$

□

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

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

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

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

2.4 Non-uniqueness of the representation

We make the following important remarks:

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

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

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

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 (15) – their equivalent counterparts $(\widetilde{\mathbf{f}},\widetilde{\mathbf{g}},\widetilde{\mathbf{x}}_0)$, 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 performance of the optimization algorithm. Nevertheless, it may be seen also as an opportunity: we can indeed contain the size of the spaces $\widehat{\mathcal{F}}$, $\widehat{\mathcal{G}}$ and $\widehat{\mathcal{X}}$ by imposing specific constraints on the solution to a priori choose a representative solution for a given class of equivalent solutions. In such a way we can restrict the design space for the optimization problem without ruling out possible solutions, thus reducing its complexity. We now show two possible ways of performing this task.

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

Consider a model $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0} \in \Phi$. Consider then the (invertible) state transformation $\widetilde{\mathbf{x}} = \mathbf{h}_1(\mathbf{x}) = \mathbf{x} - \mathbf{x}_0$. By applying the transformation (15), we get an equivalent model $\boldsymbol{\varphi}_{\widetilde{\mathbf{f}},\widetilde{\mathbf{g}},\widetilde{\mathbf{x}}_0}$, where $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}},\mathbf{u}) = \mathbf{f}(\widetilde{\mathbf{x}}+\mathbf{x}_0,\mathbf{u})$, $\widetilde{\mathbf{g}}(\widetilde{\mathbf{x}}) = \mathbf{g}(\widetilde{\mathbf{x}}+\mathbf{x}_0)$ and $\widetilde{\mathbf{x}}_0 = \mathbf{0}$. Thanks to this property, when we look for the

solution of the best-approximation problem (7) we 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 the cost functional J under the constraint $\mathbf{x}_0 = \mathbf{0}$. The statement of the best-approximation problem improves since the number of design variables decreases (the design variables are now just $\mathbf{f}$ and $\mathbf{g}$) and we have disambiguated among a number of equivalent solutions, without ruling out possible solutions.

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

Consequence of Remark 1 is that the state variables $\mathbf{x}$ are just auxiliary variables to track the time evolution of the internal state of the model, not necessarily inferring a clear physical interpretation. There is thus large freedom in their choice. Consider the case when $n \geq N_y$: one could possibly decide a priori to force the first N_y state variables $x_1, \ldots, x_{N_y}$ to coincide with the outputs $y_1, \ldots, y_{N_y}$. 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 affirmative, then we can restrict ourselves to models such that $\mathbf{g}(\mathbf{x})$ is the function extracting the first N_y component of a vector, which we denote by $\boldsymbol{\pi}^{N_y}(\mathbf{x}) = (x_1\,,\, x_2\,,\, \ldots\,,\, x_{N_y})^T$.

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

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

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

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

where the state is written in the form $\mathbf{x} = (\mathbf{y}^T, \mathbf{z}^T)^T$, where $\mathbf{z}(t)$ are the other auxiliary variables. We have not ruled out yet all the ambiguity since, being the output transparent to 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 in fact equivalent to a model in the form of (17).

Clearly, this hypothesis is not fulfilled by any model $\boldsymbol{\varphi}_{\mathbf{f},\mathbf{g},\mathbf{x}_0}$: if, for instance, the outputs $y_1, \ldots, y_{N_y}$ are linearly dependent, then $\mathbf{h}_2$ cannot be invertible. However, this case does not have any practical interest since in such case the dimension of the output can be reduced. Hence, 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 further 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

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

1. $\widehat{\mathcal{X}} = \{\mathbf{0}\}$, which we refer to as *output-outside-the-state* (the output is a function of the state, but it is not part of the state);

2. $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$ and $\widehat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$, which we refer to as *output-inside-the-state* (the first N_y state variables coincide with the output variables).

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

- Collecting the input-output observations (6) of the HF model to be approximated (i.e. to be reduced or to be identified).

- Selecting a suitable state dimension $n \geq 1$ for the reduced model, the subset $\widehat{\mathcal{F}}$ and, in the output-outside-the-state case, the subset $\widehat{\mathcal{G}}$.

- Solving the abstract problem (7); this reads, in the output-outside-the-state case:

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

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

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

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

2.6 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 case, $\widehat{\mathcal{G}}$) 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 training set (6), but this typically results into overfitting (see e.g. Rasmussen and Ghahramani 2001). The compromise stays in the middle, where the so-called Occam's hill is located (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 Solution strategy

The unknowns of problem (19) and, more in general, (18) are the functions $\mathbf{f}$ and, in the output-outside-the-state case, also $\mathbf{g}$. We are thus performing optimization in 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 (18). Thus, we introduce a family of Lagrange multipliers $\mathbf{w}_j \in \mathcal{C}^0([0,T];\mathbb{R}^n)$, for $j = 1,\dots,N_s$,

associated to the constraints given by the state equations:

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

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

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

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

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

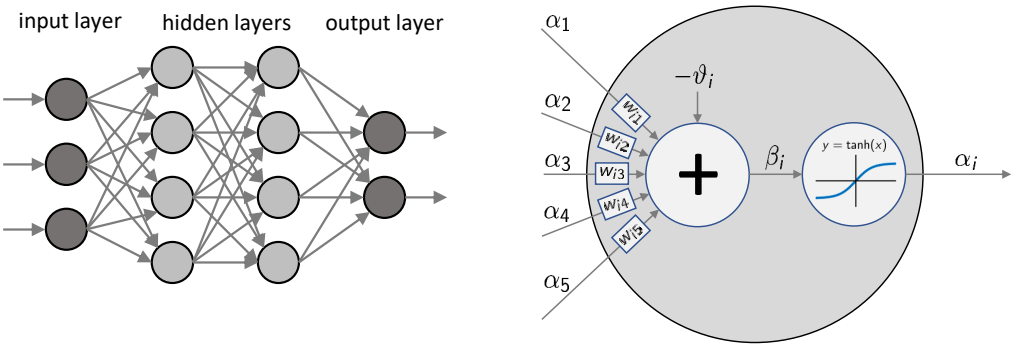
In equations (21)–(22), which state the conditions the unknowns $\mathbf{f}$ and $\mathbf{g}$ must fulfil to be a (local) minimum of the cost functional J, $\mathbf{f}$ and $\mathbf{g}$ are evaluated just in $\mathbf{x}_j(t)$ and $\widehat{\mathbf{u}}_j(t)$ for $j=1,\ldots,N_s$ and $t\in[0,T]$. However, the trajectories of $\mathbf{x}_j(t)$ and $\widehat{\mathbf{u}}_j(t)$ do not fill the whole spaces $\mathbb{R}^n$ and U. Therefore, the problem of fullfilling Eqs. (21)–(22) is ill–posed, being underdetermined. This has to be compensated for by proper regularization of the unknown themselves, by means of Occam's razor principle (see Sec. 2.6). A consequence of this is that the differentials in (22) cannot be written in gradient form and thus gradient descent strategies cannot be applied for an iterative optimization procedure.

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

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

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

The parametrization of $\mathbf{f}$ and $\mathbf{g}$ can be obtained in different manners, such as by polynomials, piecewise polynomials, truncated Fourier series, splines, NURBS, etc. Here we choose to represent them by ANNs, which can be seen as nonlinear maps parametrized by a finite number of real parameters (see Sec. 3.1). Our choice is driven by the universal approximation properties of ANNs (see Cybenko 1988, 1989) and their well recognized properties of learning from data.



(a) Feed-forward ANN

(b) Action of the i-th neuron

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

3 Optimization strategy

In Sec. 2 we have formulated the model reduction problem as a the constrained optimization problem (18). In this section we address its numerical approximation. In Sec. 3.1, we first recall ANNs and their universal approximation property and then we show how ANNs can be employed to approximate time-dependent models.

3.1 Artificial Neural Networks

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); these 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. 2a).

Each connection in the network is endowed with a weight (we denote by $w_{ij} \in \mathbb{R}$ the weight of the connection from the j-th the i-th neuron). Each neuron, except 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 preceding layer, shifted by the threshold value ϑ_i (i.e. $\beta_i = \sum_{j \in \mathbb{I}_i} w_{ij}\alpha_j - \vartheta_i$, where $\mathbb{I}_i$ and $\mathbb{O}_i$ are the sets of the neurons of the previous and next layer respectively) and then applies a nonlinear function (i.e. $\alpha_i = f_{\text{act}}(\beta_i)$). The activation function f_{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. 2b a schematic picture of the action of a generic neuron is shown.

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 α_i associated to the input layer coincide with the inputs of the ANN, while the outputs of the ANN consist in the values β_i of the output layer (notice that the activation function is not applied to the last layer). 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 $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$ is the vector collecting both the weights w_{ij} and the thresholds ϑ_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, $\alpha_{i_h} = p_h$ for $h = 1, \dots, n_I$ and $q_h = \beta_{o_h}$ for $h = 1, \dots, n_O$.

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 (see e.g. Yegnanarayana 2009), a measure of the performance of the network is available, so that gradient-based optimization algorithms can be applied to determine the parameters $\boldsymbol{\mu}$ in an iterative manner, until a convergence criterion is satisfied.

By a classical result by Cybenko (Cybenko 1989), known as *universal approximation theorem*, ANNs with a single hidden layer can approximate with arbitrarily small error any continuous function on a compact set, provided that a sufficient number of hidden neurons are employed.

In the following, we recall 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 $\mathbf{q}$ with respect to the input $\mathbf{p}$ 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 them, we need first to compute the sensitivity of the network output $\mathbf{q}$ with respect to the neurons outputs α_j. Thus, we have, for $h = 1, \dots, n_O$, and for any j belonging to the input layer or to the hidden layers:

$$\frac{\partial q_h}{\partial \alpha_j} = \begin{cases} w_{o_h j} & \text{if } j \in \mathbb{I}_{o_h} \\ \displaystyle\sum_{k \text{ s.t. } j \in \mathbb{I}_k} \frac{\partial q_h}{\partial \alpha_k} f'_{\text{act}}(\beta_k) w_{kj} & \text{otherwise.} \end{cases}$$

Notice that to derive the sensitivities associated to a given layer, the sensitivities of the next layer are required, due to the chain rule. For this reason, the calculation should be performed first for the last hidden layer and then traced-back to the input layer (whence the name of *back-propagation*). Then, we have, for $h = 1, \dots, n_O$, for $k = 1, \dots, n_I$, for $i \notin \{i_1, \dots, i_{n_I}\}$ and $j \in \mathbb{I}_i$:

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

3.1.2 Compact representation of the ANN

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

$$\mathbf{f}(\mathbf{p}; \boldsymbol{\mu}) = \mathrm{W}_{n_L - 1}\, f_{\text{act}}\left(\dots \mathrm{W}_2\, f_{\text{act}}\left(\mathrm{W}_1\, \mathbf{p} - \boldsymbol{\vartheta}_1\right) - \boldsymbol{\vartheta}_2 \dots\right) - \boldsymbol{\vartheta}_{n_L - 1}, \tag{24}$$

where the application of the activation function f_{act} must be interpreted component-wise. We notice that the parameters vector $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$ collects the entries of the matrices W_L and of the vectors $\boldsymbol{\vartheta}_L$.

3.1.3 Transforming the ANN through affine changes of variables

Thanks to the compact representation (24), 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 $\mathbf{q} = f(\mathbf{p}; \boldsymbol{\mu})$, i.e. $\widetilde{\mathbf{p}} = \mathrm{A}\mathbf{p} + \mathbf{b}$, $\widetilde{\mathbf{q}} = \mathrm{C}\mathbf{q} + \mathbf{d}$. By simple calculations it turns out that the weights $\widetilde{\boldsymbol{\mu}}$ such that $\widetilde{\mathbf{q}} = f(\widetilde{\mathbf{p}}; \widetilde{\boldsymbol{\mu}})$ are obtained by substituting the weight and the thresholds associated to the first and last layers of synapses as follows:

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

3.1.4 The crucial role of normalization

Even if ANNs can work with data values spanning different order of magnitudes, their performance is optimized when both input $\mathbf{p}$ and output $\mathbf{q}$ data are normalized. Therefore, before training the network, we normalize both the input $\mathbf{u}$ and the output $\mathbf{y}$ so that their components take values in the interval $[-1, 1]$. Moreover, since the output of the ANN representing $\mathbf{f}$ are derivatives with respect to time, we also normalize time with respect to the fastest time scale associated with the HF model (1), which can be estimated by considerations on the training set. A rough estimation of the order of magnitude suffices in most of the cases. Once 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 inputs, outputs and time is not enough to ensure that also the state variables (or just the hidden variables in the output-inside-the-state case) are in the range $[-1, 1]$, because of their hidden nature (see again Sec. 2.4). Therefore, at each optimization epoch, if the mean square value of a component of the state exceeds a lower or an upper bound (that we set to 0.1 and 2 respectively), we renormalize it by means of the affine transformation formulas of Sec. 3.1.3.

3.2 Representation of the unknowns in terms of ANN

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

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

Proof. ANNs are by construction Lipschitz continuous, being the composition of Lipschitz continuous functions, thus we have $\mathcal{F}_n^{\text{ANN}} \subset \mathcal{F}_n$ and $\mathcal{G}_n^{\text{ANN}} \subset \mathcal{G}_n$. Moreover, properties (10) and (11) holds by the universal approximation theorem (Cybenko 1989). Therefore, the thesis is a corollary of Prop. 3. □

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

3.3 Discretization of the state equation and of the objective functional

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

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

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

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

We notice that, after time discretization, the ANN model of Eq. (26) has a structure resembling that of a *recurrent neural network* (RNN), that is an ANN whose internal state is fed back to the input after each evaluation of the network (see e.g. Haykin 2009). RNNs are widely used for machine learning tasks involving sequential inputs, such as handwriting recognition, speech recognition and natural language processing, and also for time series prediction (Connor, Martin, and Atlas 1994; Graves 2013; Hinton et al. 2012). However, the approach presented in the present work is more general (indeed, RNNs can be interpreted as particular cases of our formulation, see Sec. 3.2) and it allows to recast the MOR problem within the setting of a best-approximation problem (7). Moreover, by addressing the problem at the continuous level and thanks to the setting of (14), the reduced model that we obtain by solving the best-approximation problem is independent of the RNN structure used in the training stage. In other words, once the optimization problem in the discrete setting (26) has been solved (i.e. the ANN has been trained), the reduced model is available in the form of Eq. (2). This allows more flexibility in the choice of the time discretization scheme and time step size than with RNN, wherein the forward Euler scheme is exclusively used with fixed Δt. A further advantage is the possibility of coupling the ROM problem with other mathematical models.

3.4 Training the ANN: optimization algorithm

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

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

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 minimization problem we employ the Levenberg-Marquardt method, which is designed for least-squares problems in the form (3.4) (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 in (Nocedal and Wright 2006).

The Levenberg-Marquardt descent direction can be seen as a combination of the steepest-descent direction (which is recovered for $\lambda^{(k)} \gg 1$) with the Gauss-Newton direction ($\lambda^{(k)} = 0$), which is an approximation of the Newton direction obtained by neglecting the quadratic term in the computation of the Hessian (that is the second term in $D^2(\frac{1}{2}\mathbf{r}^T\mathbf{r}) = \nabla^T\mathbf{r}\nabla\mathbf{r} + \sum_j r_j D^2 r_j$, where we denote by D^2 the second derivative operator). To exploit 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 progressively switch to the Gauss-Newton direction when the solution approaches a minimum point. We chose $\lambda^{(k)} = \min\left\{|\mathbf{r}^{(0)}|^2, |\nabla^T \mathbf{r}^{(k)} \mathbf{r}^{(k)}|\right\}$, which ensures superlinear convergence for the method, provided that the objective functional is twice continuously differentiable (see Grippo and Sciandrone 2011).

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

3.5 Computation of sensitivities

The Levenberg-Marquardt method requires the computation of the sensitivities of the residuals $r^{j,k,h}$ with respect to the unknown parameters $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Since the map from $\boldsymbol{\mu}$ to the variables

$\mathbf{x}_j^k$ is implicitly given by the state equation in (26), we employ the Lagrange multipliers method to compute the sensitivities. Generally speaking, suppose that one needs to compute the sensitivity of a quantity $Q = Q(\{\mathbf{x}_j^k\}_{j,k})$ with respect to $\boldsymbol{\mu}$. By introducing a family of Lagrange multipliers $\mathbf{w}_j^k$, the Lagrangian associated to the problem reads:

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

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

$$\begin{cases} \mathbf{w}_j^{M_j} = \dfrac{\partial Q}{\partial \mathbf{x}_j^{M_j}} \\ \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}_j^{k+1} + \dfrac{\partial Q}{\partial \mathbf{x}_j^k}, \qquad \text{for } k = 0, \dots, M_j - 1. \end{cases} \tag{27}$$

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

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

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

$$\begin{cases} \mathbf{w}^k = \mathbf{w}^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}^{k+1}, \qquad \text{for } k = 0, \dots, m-1 \\ \mathbf{w}^m = \sqrt{\Delta t}\, \nabla_{\mathbf{x}}^T \mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu})\, \mathbf{e}_h. \end{cases} \tag{28}$$

The line-search algorithm for determining 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

$$\begin{cases} \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu})\, \mathbf{w}_j^{k+1} \\ \qquad + \Delta t\, \nabla_{\mathbf{x}}^T \mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu})\, (\mathbf{g}(\mathbf{x}_j^k; \boldsymbol{\nu}) - \mathbf{y}_j^k), \qquad \text{for } k = 0, \dots, M_j - 1 \\ \mathbf{w}_j^{M_j} = \mathbf{0}, \end{cases} \tag{29}$$

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

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

Table 1: Test cases list.

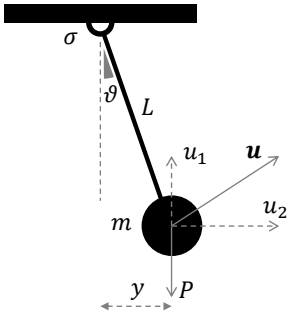


Figure 3: Nonlinear pendulum problem considered in Sec. 4.1

4 Numerical Results

In this section we assess the capabilities of the proposed reduction approach through three test cases, summarized in Table 1. First, in Sec. 4.1, we consider the nonlinear pendulum problem. Thanks to the modest complexity of this test case ($N = 2$), we can perform a sensitivity analysis on the network architecture, highlighting the Occam's hill (Sec. 2.6). Moreover, we can directly compare the internal dynamics of the reduced models with that of the HF model.

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

4.1 *Test case 1*: Nonlinear pendulum

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

$$\begin{cases} \ddot{\vartheta}(t) = \frac{1}{Lm}\left(u_2(t)\cos\vartheta(t) + (u_1(t) - P)\sin\vartheta(t)\right) - \sigma\,\vartheta(t) & t \in (0, T], \\ \vartheta(0) = 0,\ \dot{\vartheta}(0) = 0. \end{cases} \tag{30}$$

We set the constants values to $m = L = 1$, $P = 2$, $c = 3$. Moreover, we set the external force $\mathbf{u} \in [-1, 1]^2$. Suppose that we are interested in predicting the horizontal displacement of the mass $y(t) = L\sin\vartheta(t)$ (in this case we drop the bold notation being the output a scalar), given the input $\mathbf{u}(t)$. The input-output map given by (30) fall within the concept of model introduced in Sec. 2.1, where $N_u = 2$, $N_y = 1$, $U = [-1, 1]^2$, $Y = [-1, 1]$. Moreover, this model can be written in the form (1), by setting $\mathbf{X} = (\vartheta, \dot{\vartheta})^T$ and thus we have $N = 2$.

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

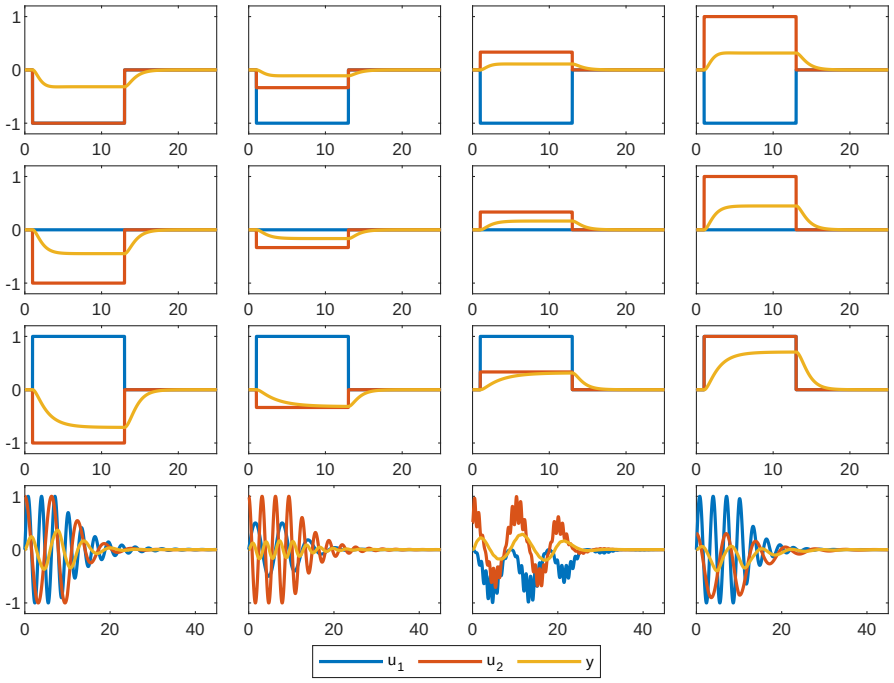


Figure 4: *Test case 1*: Training set.

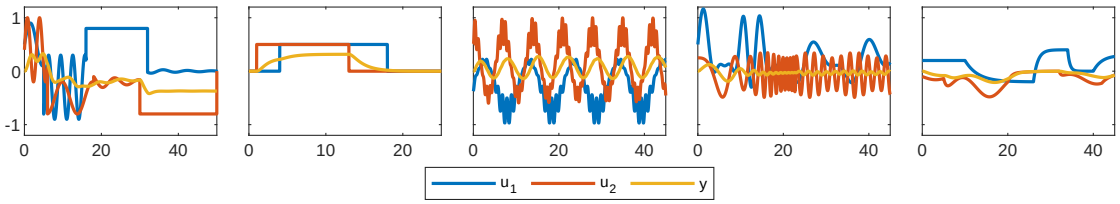


Figure 5: *Test case 1*: Validation set.

4.1.1 Training, validation and test sets

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

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

To monitor the ANN learning process, at each optimization epoch, we evaluate the performance of the ANN on a further set, that we call *validation set*. The comparison between the relative L^2 error on the training set (which we denote by $\mathcal{E}_{\text{train}}$) and the error on the validation set (which we denote by $\mathcal{E}_{\text{val}}$) allows to perform *regularization by early stopping* (see e.g. Yegnanarayana 2009): as long as the error on the validation set start increasing, we stop the optimization loop since this is an indication that overfitting occurs. In the validation set we place 5 samples, comprising several working regimes of the system (see Fig. 5). Notice that in this set we also switch from a regime to another inside the same sample, to monitor the capability of the model to cope with it.

Finally, when the optimization loop is completed, we test the performance of the reduced model

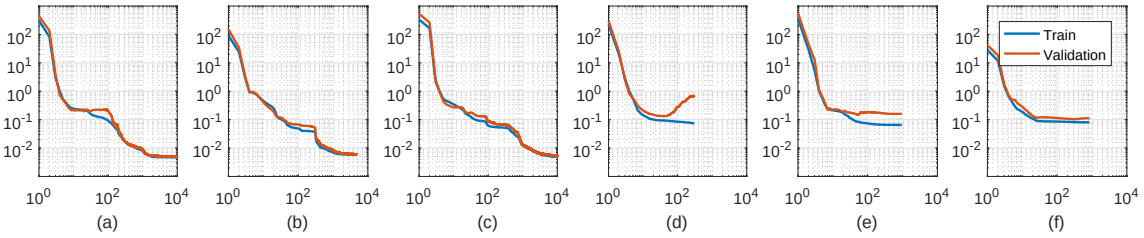


Figure 6: *Test case 1*: examples of evolution of the error on the training ($\mathcal{E}_{\text{train}}$) and validation sets ($\mathcal{E}_{\text{val}}$).

on a large-size *test set*, comprising step responses, oscillatory inputs and randomly generated ones. The test set amounts to 126 samples.

4.1.2 Sensitivity analysis of the network complexity

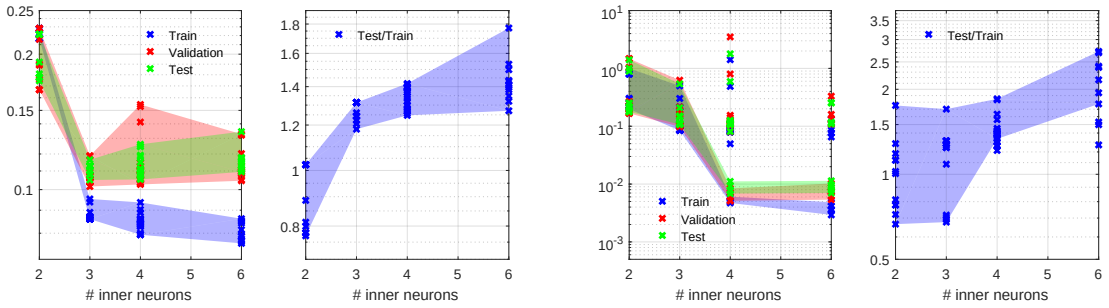
The objective functional $\mathcal{J}$ is non-convex and features several many local minima. As a consequence, the optimized ANN obtained by means of the strategy of Sec. 3.4 may depend on the initialization of the ANN itself. Since we adopt a random initialization for the ANN, by running the proposed strategy several times with the same ANN architecture, we actually end up with very different results. This is intimately linked to the non-uniqueness of representation of a given model (see Remark 1): very different ANNs may represent the same model. We will go back to this issue in Sec. 4.1.3.

Fig. 6 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 random initializations of the ANN . 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, thus indicating good performances of the learning process; 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 stop the optimization process.
- In (e), instead, a more subtle case of overfitting occurs since it originates in the early stages of the optimization and 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 selected to handle with this issue, even if this is beyond the scope of the present work.

In the following we perform a sensitivity analysis of the performance of the proposed reduction approach w.r.t. the complexity of the network, i.e. the number of hidden neurons, which we denote by N_{neurons}. For simplicity, we consider only the case of ANNs with a single hidden layer. Figure 7 shows the dependency of the performance of the proposed reduction approach on the number of hidden neurons, in the cases $n = 1$ and $n = 2$. Each cross is associated to a single test, while the 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).

Let us 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. 7a). However, by further increasing N_{neurons}, we cross the Occam's hill (see Sec. 2.6) and, even if $\mathcal{E}_{\text{train}}$ keeps decreasing, $\mathcal{E}_{\text{test}}$ and $\mathcal{E}_{\text{val}}$ start increasing. This phenomenon is also evident from the right figure, showing the ratio $\mathcal{E}_{\text{test}}/\mathcal{E}_{\text{train}}$ increasing with the



(a) $n = 1$ (b) $n = 2$

Figure 7: *Test case 1*: errors vs number of hidden neurons N_{neurons}, in the cases (a) $n = 1$ and (b) $n = 2$. For both cases, the left plot shows the errors $\mathcal{E}_{\text{train}}$, $\mathcal{E}_{\text{val}}$ and $\mathcal{E}_{\text{test}}$, while the right plot shows the ratio $\mathcal{E}_{\text{test}}/\mathcal{E}_{\text{train}}$. Each cross represents the final result of a test (including when the tests gets stuck into a "bad" local minimum). Coloured regions represent the areas spanned by the tests, excluding the tests which got stuck into a "bad" local minimum.

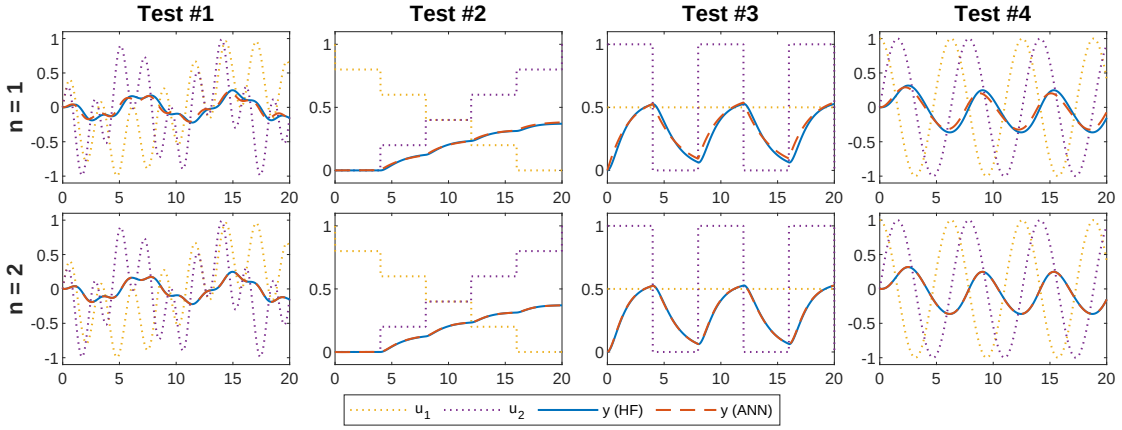


Figure 8: *Test case 1*: comparison of the exact solution (blue line) and the solution obtained with the ANN model (red line) in four different test cases. First row: one-variable model; Second column: two-variables model.

network complexity. Even if the issue of designing the training set falls beyond the purposes of the present work, we notice that with a richer training set the top of the Occam's hill would probably move towards larger values of N_{neurons}, thus allowing to reach a better performance of the reduced model.

We then consider the case $n = 2$ (see Fig. 7b). The introduction of a further state variable in the system, w.r.t. the case $n = 1$, translates in significant increment of the model ability to faithfully reproduce the input-output map given by the HF model, as expected according to Prop. 2. Indeed, even if for $N_{\text{neurons}} \leq 3$ the errors are similar to the case $n = 1$, by increasing the network complexity the errors drop by one order of magnitude for $N_{\text{neurons}} = 4$, before slowly diverging (the training error $\mathcal{E}_{\text{train}}$ decreases, while the validation $\mathcal{E}_{\text{val}}$ and test error $\mathcal{E}_{\text{test}}$ slightly increase). We also notice the occurrence of tests which got stuck in "bad" local minima (highlighted in Fig. 7b 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 apparently located in the two cases. Figure 8 shows the comparison of the results obtained in four test cases with the HF and the two selected reduced models (the first row refers to the model with $n = 1$, the second to the model obtained with $n = 2$).

4.1.3 What did the ANN actually learn?

The feasibility of the reduction approach proposed in this work is strictly related to the possibility of faithfully representing the state of the HF model (which we denote by $\mathbf{X}(t) \in \mathbb{R}^N$) by means of a lower-dimensional state $\mathbf{x}(t) \in \mathbb{R}^n$. If this is the case, then the knowledge of $\mathbf{x}(t)$ and $\mathbf{u}(t)$ provides enough information to compute, with just a little approximation, both the output $\mathbf{y}(t)$ and the evolution of the state of the system (that is to say the value of $\frac{d}{dt}\mathbf{x}(t)$).

We may thus interpret the learning process as that of implicitly building a map from the reduced-order state $\mathbf{x}$ to the full-order state $\mathbf{X}$, which is not directly observed, but it is observed only through its effects to $\frac{d}{dt}\mathbf{y}(t)$. 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 linear, here 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 for example may extract features from the snapshots which do not provide a significant contribution to the input-output map.

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

Let us consider the example addressed in this Section, for $n = N = 2$. This case may not be interesting in a MOR perspective, but it helps to shed light on the machinery behind the proposed reduction approach. Indeed, thanks to the choice $\mathbf{g}(\mathbf{x}) = \boldsymbol{\pi}^{N_y}$, we have, in principle, $x_1(t) = L \sin\vartheta(t)$. Consequently, we may speculate that the knowledge of the first variable of the reduced model x_1 provides all the information needed to reconstruct the first variable of the HF model ϑ. 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 when a trajectory in the $(\vartheta, \dot{\vartheta})$ plane crosses, then the same should happen for the trajectory in the (x_1, x_2) plane. In other words, all the trajectories of the reduced models, corresponding to the a given input $\mathbf{u}$, should be homotopic to the corresponding trajectory of the HF model.

To verify these conjectures, we consider the oscillating input also employed in the first column of Fig. 8 (i.e. $u_1 = \sin(t)\cos(1.3\,t)$, $u_2 = \cos(1.8\,t)\sin(t)$) and we compare the trajectories of the HF model (grey background in Fig. 9) with those of four different reduced models (white background in Fig. 9), obtained with four different random initializations of the ANN. In the first line of Fig. 9 we show the time evolution of the two state variables of the systems. By looking at the the figure it is apparent how different the obtained ANN models can be. However, by observing – in the second row of Fig. 9 – the trajectories in the phase space (x_1, x_2), one can see that all the trajectories are approximatively homotopic to the trajectory of the HF model in its phase space $(\vartheta, \dot{\vartheta})$. Finally, in order to better visualize the role of the second variable, we decouple it from x_1 in the following way. First, we collect the values of the two variables at the discrete times $t_0, t_1, \ldots$ in two vectors, which we denote by $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively. Next, 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 $\mathbf{z}$). Next, we subtract to $\mathbf{z}'_2$ the component parallel to $\mathbf{z}'_1$ (that is to say $\mathbf{z}''_2 = \mathbf{z}'_2 - (\mathbf{z}'_2 \cdot \mathbf{z}'_1)/\|\mathbf{z}'_1\|^2\,\mathbf{z}'_1$). Finally, we normalize $\mathbf{z}''_2$ (that is to say $\mathbf{z}'''_2 = \mathbf{z}''_2/\|\mathbf{z}''_2\|$). The third row of Fig. 9 shows the trajectories in the plane (x_1, x'''_2). For the original model, we show the trajectory in the plane $(\vartheta, \dot{\vartheta}''')$, where the coordinate $\dot{\vartheta}'''$ is computed by the same procedure as x'''_2.

4.2 *Test case 2*: Nonlinear transmission line circuit

In order to assess the capability of the proposed reduction approach to reduce the complexity of large-scale nonlinear systems, we consider a popular benchmark in MOR, namely the nonlinear transmission line circuit represented in Fig. 10 (see e.g. Chen and White 2000; Rewieński and White 2001). It is an electrical network comprising a current source, $N = 1000$ unitary resistors, N unitary capacitors and N nonlinear diods with law $i = e^{40\,v} - 1$. The input of the model is

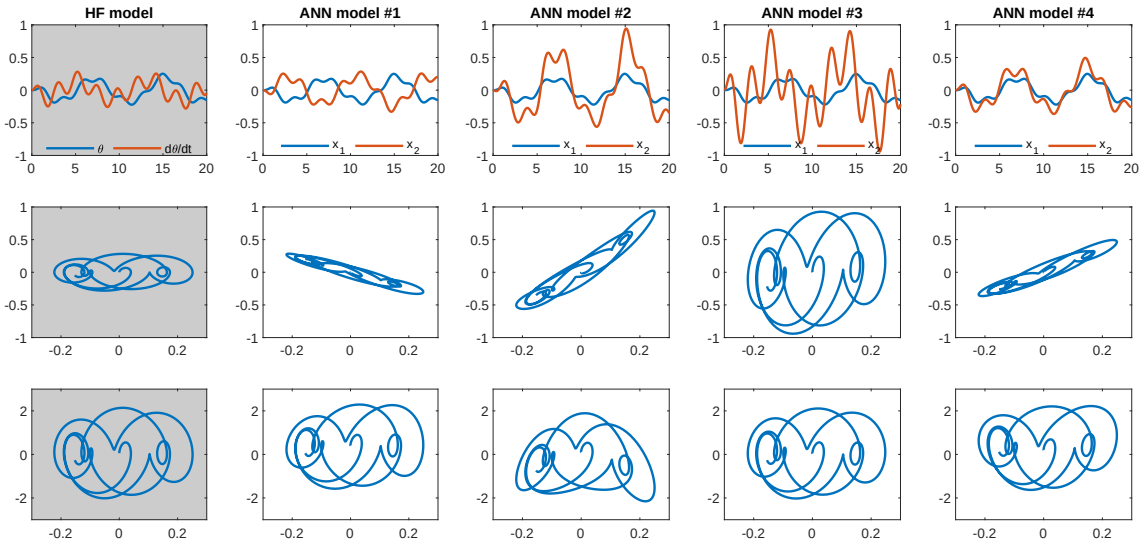


Figure 9: *Test case 1*: results of the test case considered in Sec. 4.1.3. Each column refers to a different model (first column: HF model; other columns: four different ANN models). First line: time evolution of the two state variables of the system (first column: ϑ and $\dot{\vartheta}$; other columns: x_1 and x_2). Second line: trajectories in the phase space (first column: $(\vartheta, \dot{\vartheta})$; other columns: (x_1, x_2)). Third line: trajectories in the $(\vartheta, \dot{\vartheta}''')$ phase space (first column) and (x_1, x_2''') phase space (other columns).

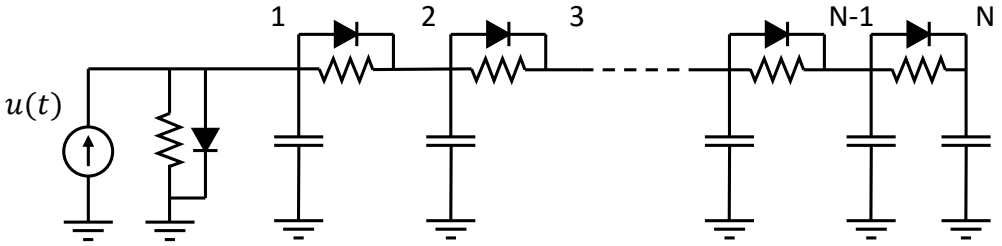


Figure 10: *Test case 2*: scheme of the nonlinear transmission line electric circuit considered in Sec. 4.2.

the current source $u(t)$, taking values in $U = [0, 1]$, and the output is the tension at the first node $v_1(t)$. By denoting by v_i the voltage at the i-th node, the HF model reads as follows:

$$\begin{cases} \dot{v}_1(t) = -2\, v_1(t) + v_2(t) + 2 - e^{40\, v_1(t)} - e^{40(v_1(t) - v_2(t))} + u(t) \\ \dot{v}_i(t) = -2\, v_i(t) + v_{i-1}(t) + v_{i+1}(t) + e^{40(v_{i-1}(t) - v_i(t))} - e^{40(v_i(t) - v_{i+1}(t))}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } i = 2, \dots, N-1 \\ \dot{v}_N(t) = -v_N(t) + v_{N-1}(t) - 1 + e^{40(v_{N-1}(t) - v_N(t))}, \end{cases} \tag{31}$$

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

We consider the training set represented in Fig. 11, comprising 5 step inputs and 20 randomly generated input signals, each of 1 s duration. In this test case we consider both the output-inside-the-state and the output-outside-the-state cases, for $n = 1, 2, 3$ and we compare the results. In all the cases we employ ANNs with a single hidden layer, with respectively 8 and 3 neurons in the ANN for $\mathbf{f}$ and $\mathbf{g}$. For the time discretization, we employ a time step of $\Delta t = 5 \cdot 10^{-3}$.

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

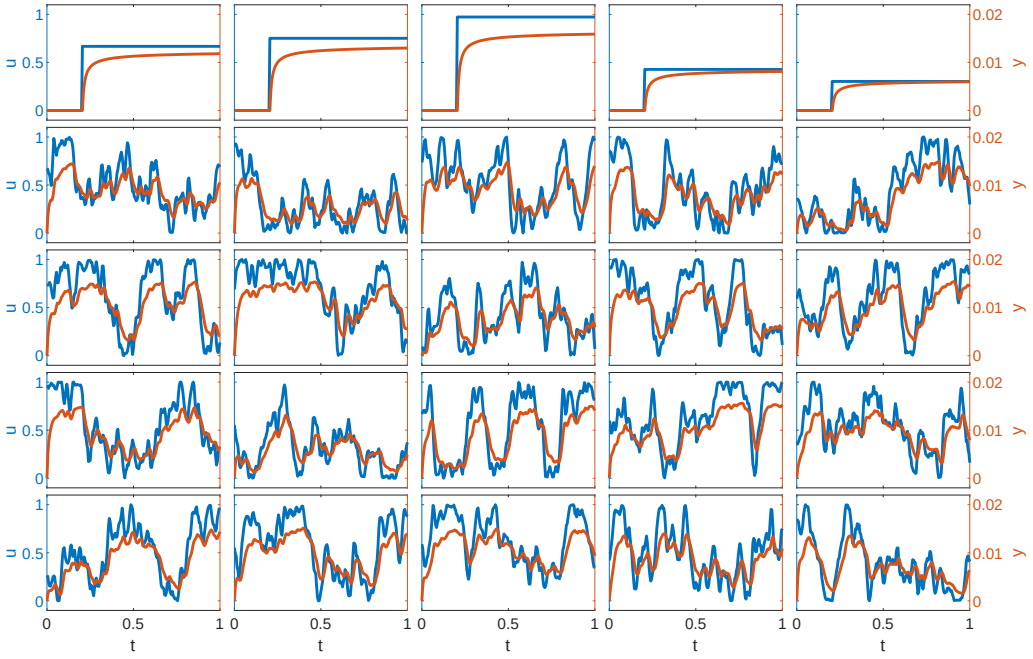


Figure 11: *Test case 2*: training set. Blue lines (axis on the left) represent the time evolution of the input, red lines (axis on the right) represent the time evolution of the output.

Therefore we expect the output-inside-the-state case to provide similar results than the output-outside-the-state case.

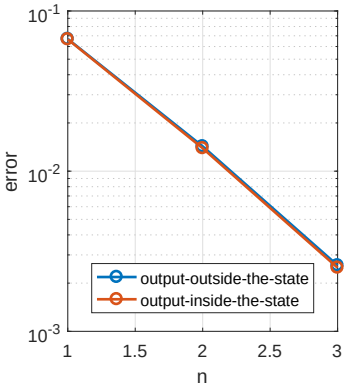
In Fig. 12a we show $\mathcal{E}_{\text{test}}$, the relative L^2 error on the test set (which comprises 25 step responses and 80 randomly generated inputs), for the different cases considered. We notice that, as expected, the two approaches provide, for a given value of n, very similar results. The output-inside-the-state case is thus preferable since it is more efficient both in the offline phase (since the number of design variables is lower) and in the online phase (since $\mathbf{g}$ does not need to be evaluated at each time step). We also notice that, coherently with Prop. 2, the error $\mathcal{E}_{\text{test}}$ decreases as n increases, reaching, for $n = 3$, a remarkably good approximation level (nearly $2.5 \cdot 10^{-3}$). In Fig. 12b we compare the response of the HF model with that of the three reduced models obtained with the output-inside-the-state case in the time-domain.

4.3 *Test case 3*: Heat equation (PDE)

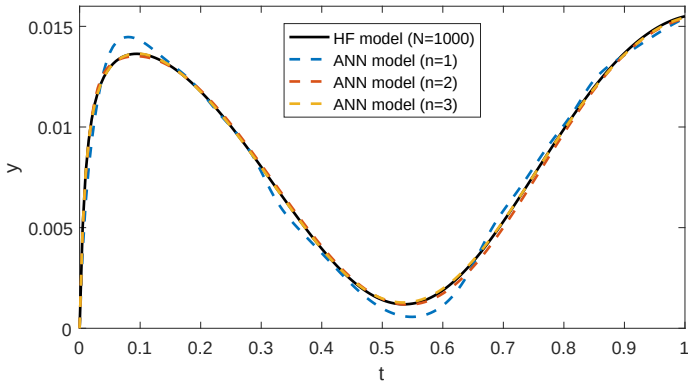
We consider now the application of the proposed reduction approach to the MOR of a parabolic PDE problem, which extends the benchmark problem considered in Manzoni, Pagani, and Lassila 2016, where its Reduced Basis (RB) reduction has been considered. Consider the spatial domain $\Omega = (0, 1.5)^2$, whose boundary $\partial\Omega$ is partitioned into the top border Γ_t, in contact with a heat reservoir with zero temperature, the bottom border Γ_b, with a constant inward heat flux $\varphi = 1$, and in the wall borders Γ_w, characterized by no-flux boundary conditions (see Fig, 13). The time evolution of the spatially distributed temperature $\psi(\mathbf{x}, t)$ in the domain Ω is thus described by the heat equation:

$$\begin{cases} \frac{\partial}{\partial t}\psi(\mathbf{x},t) - \text{div}\left(k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\right) & \text{in } \Omega, \text{ for } t > 0 \\ k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\cdot\mathbf{n} = 0 & \text{on } \Gamma_w, \text{ for } t > 0 \\ k(\mathbf{x},\mathbf{u}(t))\,\nabla\psi(\mathbf{x},t)\cdot\mathbf{n} = \varphi & \text{on } \Gamma_b, \text{ for } t > 0 \\ \psi(\mathbf{x},t) = 0 & \text{on } \Gamma_t, \text{ for } t > 0 \\ \psi(\mathbf{x},0) = 0 & \text{on } \Omega. \end{cases} \tag{32}$$

Let us partition the domain Ω into 9 subdomains Ω_i of equal size, for $i = 1, \dots, 9$, as in Fig. 13. Let us consider the piecewise constant thermal conductivity coefficient k, parametrized by the



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

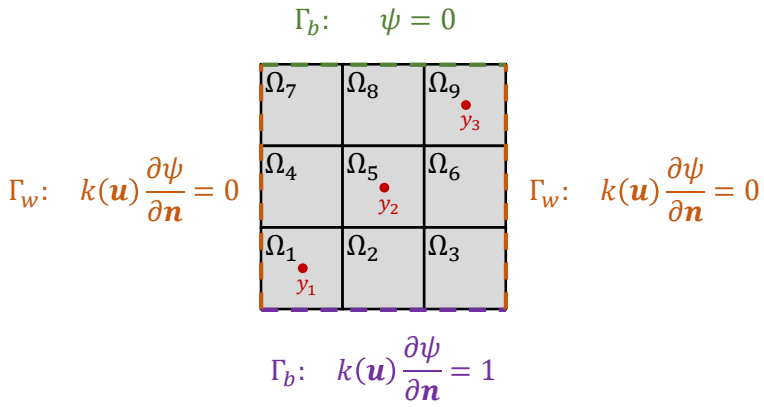


Figure 13: *Test case 3*: domain and boundary conditions.

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 Ω_i. Consider then three probes, located at the centre of the subdomains Ω_1, Ω_5 and Ω_9, measuring the time evolution of the temperature in such points. The output $\mathbf{y}(t) \in \mathbb{R}^3$ is the vector collecting the three temperature values.

For the HF solution of (32), we consider the P2 Finite Element approximation on a 30 by 30 uniform square elements grid and we employ the Forward Euler scheme, with $\Delta t = 10^{-2}$, for the time discretization, implemented in the MATLAB finite element library `feamat` (Pegolotti 2019). The HF model has dimension $N = 3721$.

In this test case we compare the results obtained with the proposed reduction approach with those obtained with a popular MOR method in the field of PDEs, namely the RB method, which exploits the linearity of Eq. (32) and the affine dependence on $\mathbf{u}$ (Quarteroni, Manzoni, and Negri 2015). 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 case, 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 15 reports a subset of the training 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

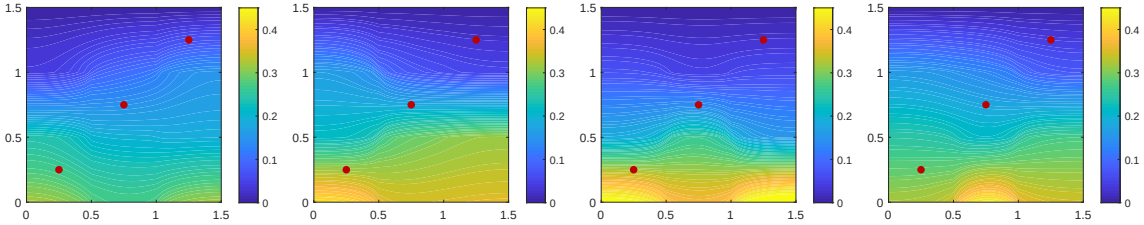


Figure 14: *Test case 3*: 4 examples of snapshots obtained at different times with different inputs.

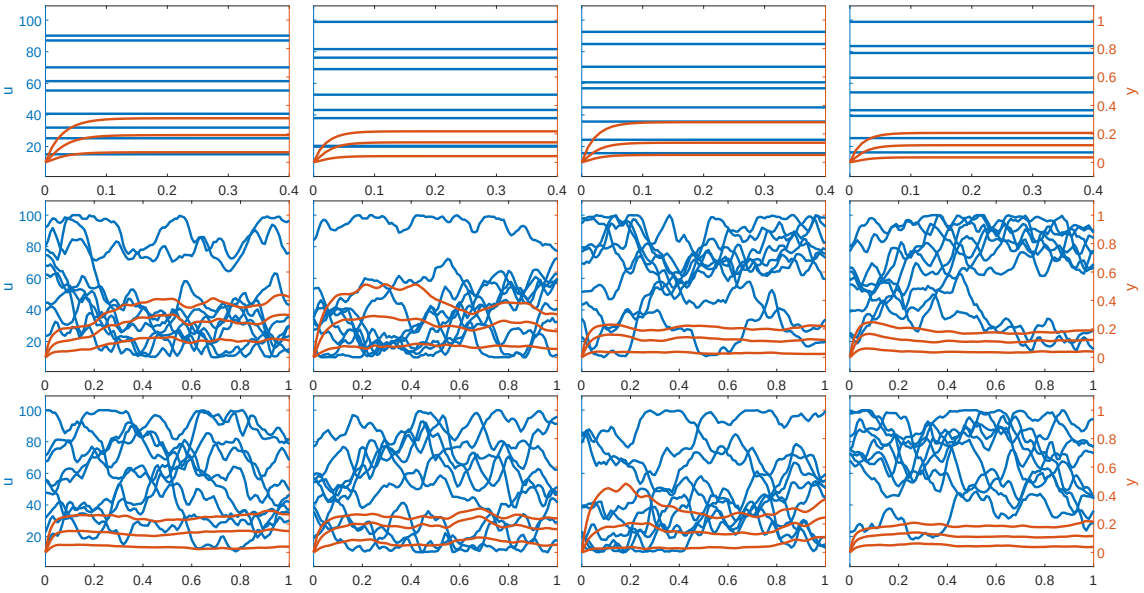


Figure 15: *Test case 3*: a subset of the training set. Blue lines (axis on the left) show the time evolution of the 9 inputs (i.e. the thermal conductivities of the 9 subdomains), red lines (axis on the right) show the time evolution of the three outputs (i.e. the temperature in the location of the three probes).

2000), and 50 random transients of duration 1 s. As of the RB method, we build the basis by POD of the snapshot matrix obtained by sampling every 0.1 s the same 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. 17 we compare the results obtained by the two methods, by evaluating the error on a test set composed by 20 steady-state inputs, 50 random input signals of duration 1 s (an example is reported in Fig. 16) and 10 random input signals of duration 10 s. The purpose of the latter choice is to asses the capability of the proposed reduction approach to approximate the evolution of the HF model also for longer time horizons than the ones used in the training phase. As is shown in Fig. 17a, 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 approach too. Even if this result is implementation dependent, it can be ascribed to the fact that the online phase of the RB method requires, at each time step, 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 reduction approach lies in the offline phase, which requires the training of one or two ANNs (for $\mathbf{f}$ and $\mathbf{g}$), while for the RB method we just need to compute the SVD decomposition of the snapshot matrix and to project both the model matrices and right-hand side. Moreover, the computational complexity of the training rapidly grows with n and, for high n, large training sets are needed to avoid overfitting, thus preventing the applicability of the

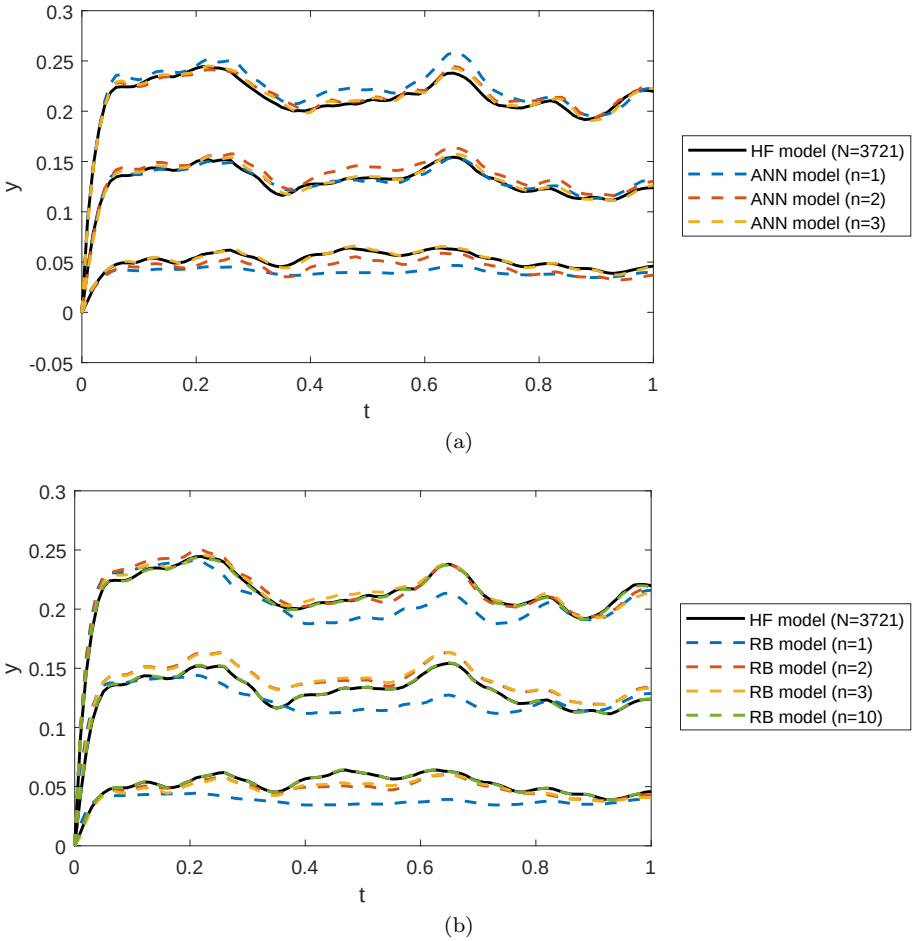


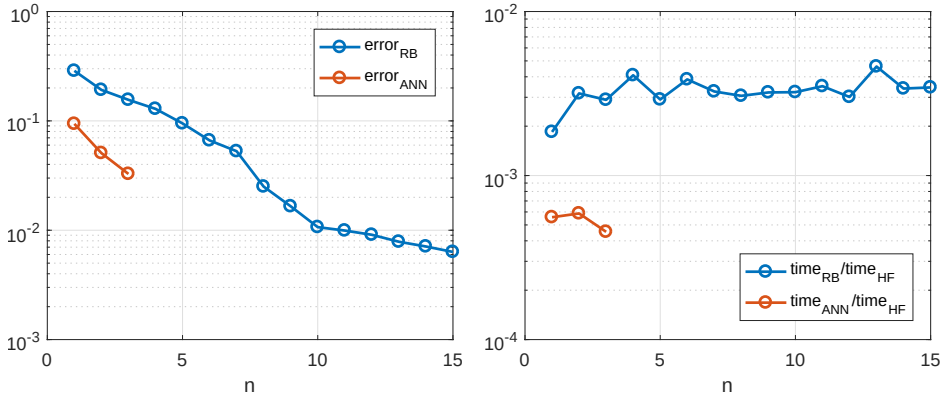
Figure 16: *Test case 3*: time-domain comparison of the result of the HF model with those of the different reduced models for a random input: (a) ANN models vs HF model, (b) RB models vs HF model.

proposed approach to large n. For this reason, we limit ourselves to the cases $n = 1, 2, 3$. However, from the presented results it seems that with the proposed approach it is possible to get a good approximation of complex models (like the PDE model considered in this section) just with a few variables, so we do not actually need to increase n if the approximation level is satisfactory for the application. We also notice that in this work we employed very basic tools to train the ANNs, while the offline phase may be considerably speeded up with the application of more advanced techniques (see Sec. 5) which, besides decreasing the training time also minimize overfitting, thus reducing the number of samples needed to train the ANN.

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

5 Conclusions and perspectives

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



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

Figure 17: *Test case 3*: comparison between the proposed reduction approach and the RB method. (a) Relative L^2 error versus number of variables of the reduced model n; (b) Mean computational time required by the reduced model, normalized w.r.t. that required by the HF model to simulate the same amount of physical time, versus n.

simpler models. The latter consists in a class of ODE models described by means of an ANN (or by two of them, in the output-outside-the-state case), which is fed by input-output pairs originated from the HF system. Thanks to this formulation, it is possible to compute the sensitivity of the model error with respect to the parameters of the ANNs, and to exploit standard optimization techniques to make the ANN learn the underlying physics of the system. The proposed reduction approach can be applied to a wide class of dynamical models with time dependent inputs, subject to some minimal requirements (see Sec. 2.1).

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

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

We have assessed the effectiveness of the proposed approach on a simple case study (namely the nonlinear pendulum) and by investigating the reduction of two large-scale problems (a nonlinear system of ODEs and a parabolic PDE), featuring thousands of degrees of freedom. 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 state variables. We have also compared the performance of the proposed reduction approach with that of the RB method, one of the most popular MOR methods in the field of PDEs. For a given reduced model size, despite a more expensive offline phase, the proposed reduction approach yielded much more accurate reduced models than the RB method, also featuring a lower online cost in terms of computational time.

Some crucial aspects, where there is possibility for improvement of the proposed reduction approach, have not been fully explored in the present study. In particular, we have not dealt with the issue of the training set design, where it is possible to employ automatic procedures to select an optimized training set, in a similar manner to the selection of snapshots in the RB framework (see e.g. 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 training set (SGD, see e.g. Bottou 2010), dropout (Srivastava et al. 2014), batch normalization (Ioffe and Szegedy 2015) and progressive layers freezing (Brock et al. 2017).

Finally, we notice that the formulation of the MOR problem in terms of minimization problem

potentially allows to easily incorporate into the learning stage some a priori knowledge on the HF model, by suitably accounting for a penalization term.

Acknowledgements

The authors gratefully thank A. Menafoglio and S. Pagani (MOX, Politecnico di Milano) for the interesting and useful discussions about MOR and statistical learning and L. Pegolotti (École Polytechnique Fédérale de Lausanne) for kindly sharing the library `feamat` (Pegolotti 2019).

References

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

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

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

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

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

Barrault, M., Y. Maday, N. C. Nguyen, and A. T. Patera (2004). "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus Mathematique* 339.9, pp. 667–672.

Baur, U., C. Beattie, P. Benner, and S. Gugercin (2011). "Interpolatory projection methods for parameterized model reduction". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2489–2518.

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

Benner, P., V. Mehrmann, and D. C. Sorensen (2005). *Dimension reduction of large-scale systems*. Vol. 35. Springer.

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

Brock, A., T. Lim, J. M. Ritchie, and N. Weston (2017). "FreezeOut: Accelerate Training by Progressively Freezing Layers". In: *arXiv preprint arXiv:1706.04983*.

Brunton, S. L., J. L. Proctor, and J. N. 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, G. and R. L. Berger (2002). *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA.

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

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

Connor, J. T., R. D. Martin, and L. E. Atlas (1994). "Recurrent neural networks and robust time series prediction". In: *IEEE transactions on neural networks* 5.2, pp. 240–254.

Cybenko, G. (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, D., B. Haegeman, and T. 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, M., B. Haasdonk, and M. 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, R. and L. Sirovich (1995). “Karhunen–Loeve procedure for gappy data”. In: *Journal of the Optical Society of America A* 12.8, pp. 1657–1664.

Fink, J. P. and W.C. 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.

Freno, B. A. and K. T. Carlberg (2018). “Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations”. In: *arXiv preprint arXiv:1808.02097*.

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

Gosea, I. V. and A. 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.

Graves, A. (2013). “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850*.

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

Gu, C. (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.

Guo, M. and J. S. Hesthaven (2019). “Data-driven reduced order modeling for time-dependent problems”. In: *Computer methods in applied mechanics and engineering* 345, pp. 75–99.

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

Haykin, S. S. (2009). *Neural networks and learning machines*. Vol. 3. Pearson Upper Saddle River.

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

Hesthaven, J. S., G. Rozza, and B. Stamm (2016). *Certified reduced basis methods for parametrized partial differential equations*. Springer.

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath (2012). “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

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

Ioffe, S. and C. Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167*.

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

Krige, D. 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.

Lee, K. and K. Carlberg (2018). “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders”. In: *arXiv preprint arXiv:1812.08373*.

Lefteriu, S. and A. 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, L. (1998). “System identification”. In: *Signal Analysis and Prediction*. Springer, pp. 163–173.

Loeve, M. (1978). “Probability theory, vol. ii”. In: *Graduate Texts in Mathematics* 46, pp. 0–387.

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

Maday, Y., N. C. Nguyen, A. T. Patera, and G. S. Pau (2007). “A general, multipurpose interpolation procedure: the magic points”. In:

Manzoni, A., S. Pagani, and T. 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, A.J. and A.C. 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, M. D., R. J. Beckman, and W. 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, A., P. Secchi, and M. Dalla Rosa (2013). "A Universal Kriging predictor for spatially dependent functional data of a Hilbert Space". In: *Electronic Journal of Statistics* 7, pp. 2209–2240.

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

Narendra, K. S. and K. 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, F., A. Manzoni, and D. Amsallem (2015). "Efficient model reduction of parametrized systems by matrix discrete empirical interpolation". In: *Journal of Computational Physics* 303, pp. 431–454.

Nelles, O. (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*. second. New York, NY, USA: Springer Science & Business Media.

Pearson, K. (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, L. (2019). *Feamat*. URL: `https://github.com/lucapegolotti/feamat`.

Peherstorfer, B., S. Gugercin, and K. Willcox (2017). "Data-Driven Reduced Model Construction with Time-Domain Loewner Models". In: *SIAM Journal on Scientific Computing* 39.5, A2152–A2178.

Peherstorfer, B. and K. Willcox (2015a). "Dynamic data-driven reduced-order models". In: *Computer Methods in Applied Mechanics and Engineering* 291, pp. 21–41.

— (2015b). "Online adaptive model reduction for nonlinear systems via low-rank updates". In: *SIAM Journal on Scientific Computing* 37.4, A2123–A2150.

Peterson, J. 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, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1986). *Numerical recipes: the art of scientific computing*. Cambridge Univ. Press, New York.

Prud'Homme, C., D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici (2002). "Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods". In: *Journal of Fluids Engineering* 124.1, pp. 70–80.

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

Quarteroni, A. and G. Rozza (2014). *Reduced order methods for modeling and computational reduction*. Vol. 9. Springer.

Quarteroni, A. and A. 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, M. and George E. Karniadakis (2018). "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357, pp. 125–141.

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2017a). "Machine learning of linear differential equations using Gaussian processes". In: *Journal of Computational Physics* 348, pp. 683–693.

— (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*.

— (2018). "Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems". In: *arXiv preprint arXiv:1801.01236*.

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707.

Rasmussen, C. E. (2004). "Gaussian processes in machine learning". In: *Advanced lectures on machine learning*. Springer, pp. 63–71.

Rasmussen, C. E. and Z. Ghahramani (2001). "Occam's razor". In: *Advances in neural information processing systems*, pp. 294–300.

Regazzoni, F., L. Dedè, and A. 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, M. and J. 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.

San, O. and R. Maulik (2018). "Neural network closures for nonlinear model order reduction". In: *Advances in Computational Mathematics*, pp. 1–34.

Sirovich, L. (1987). "Turbulence and the dynamics of coherent structures. I. Coherent structures". In: *Quarterly of Spplied Mathematics* 45.3, pp. 561–571.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Trehan, S., K. T. Carlberg, and L. J. Durlofsky (2017). "Error modeling for surrogates of dynamical systems using machine learning". In: *International Journal for Numerical Methods in Engineering* 112.12, pp. 1801–1827.

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