Automatica 196 (2027) 113333

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Manifold meta-learning for reduced-complexity neural system identification☆

Marco Forgione [a,*], Ankush Chakrabarty [b], Dario Piga [a], Matteo Rufolo [a], Alberto Bemporad [c]

[a] SUPSI, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland
[b] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA
[c] IMT School for Advanced Studies Lucca, Lucca, Italy

ARTICLE INFO

Article history:
Received 31 March 2025
Received in revised form 3 June 2026
Accepted 18 August 2026

ABSTRACT

System identification has greatly benefited from deep learning techniques, particularly for modeling complex, nonlinear dynamical systems with partially unknown physics where traditional approaches may not be feasible. However, deep learning models often require large datasets and significant computational resources for both training and inference due to their high-dimensional parameterizations. To address this challenge, we propose a meta-learning framework that discovers a low-dimensional manifold within the parameter space of an over-parameterized neural network architecture. This manifold is learned from an abundant meta-dataset of input–output sequences generated by a simulator of the class of systems of interest, and thereafter enables efficient model training while preserving the network's expressive power for the considered system class. Unlike bilevel meta-learning approaches, our method employs an auxiliary neural network to map datasets directly onto the learned manifold, eliminating the need for costly second-order gradient computations during meta-training and generally reducing the number of first-order updates required in inference, which could be expensive for large models. Finally, we derive a probabilistic bound that quantifies the performance gap between training on the learned low-dimensional manifold and the full parameterization. We validate our approach on a family of Bouc–Wen oscillators, which is a well-studied nonlinear system identification benchmark. We demonstrate that we are able to learn accurate models even in small-data scenarios.

© 2026 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, system identification has advanced significantly with the adoption of deep learning tools (Bemporad, 2025; Champneys et al., 2024; Deshpande et al., 2023; Pillonetto et al., 2025). These methods have proven effective in modeling complex, nonlinear dynamical systems where the underlying physics is partially or entirely unknown, relying on flexible and expressive black-box components such as neural networks. However, a fundamental challenge of black-box system identification is the need for large datasets to ensure model reliability across a wide range of operating conditions. Even when such datasets are available, training deep models with numerous parameters can be computationally demanding, limiting their practical applicability. This contrasts with physics-based modeling, which, when feasible, enables data-efficient learning and typically results in models that generalize well beyond the training regime.

To bridge the gap between black-box and physics-based approaches, recent research has explored the opportunity to include domain knowledge into deep-learning architectures and training algorithms, while preserving their flexibility. These efforts include designing custom model structures that satisfy known physical constraints by design (Deshpande et al., 2023; Forgione & Piga, 2021; Moradi et al., 2023) and introducing regularization terms in the loss function to promote the satisfaction of known dynamics, as seen in physics-informed machine learning (Nghiem et al., 2023). While highly successful, these approaches still require expert knowledge and manual tuning for the design of appropriate architectures and regularizers.

This paper addresses the challenge of learning simplified model architectures in an automated manner using meta-learning techniques (Hospedales et al., 2021). In line with the meta-learning

☆ This work was supported as a part of NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 51NF40_225155). This research was supported by the European Research Council (ERC), Advanced Research Grant COMPACT (Grant Agreement No. 101141351). The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Simone Formentin under the direction of Editor Alessandro Chiuso.

* Corresponding author.
E-mail addresses: marco.forgione@supsi.ch (M. Forgione), achakrabarty@ieee.org (A. Chakrabarty), dario.piga@supsi.ch (D. Piga), matteo.rufolo@supsi.ch (M. Rufolo), alberto.bemporad@imtlucca.it (A. Bemporad).

https://doi.org/10.1016/j.automatica.2026.113333
0005-1098/© 2026 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

setting for system identification (Chakrabarty et al., 2025; Forgione et al., 2023), we assume access to a *meta-dataset*, namely, a collection of datasets generated by different, yet related, dynamical systems representative of a target population that we aim to describe. The proposed method requires an abundant meta-data regime: in practice, the meta-dataset is generated by a simulator of the class of systems of interest, with parameters randomized within physically plausible ranges.

Additionally, we assume that a given black-box architecture, referred to as the *base architecture*, is sufficiently expressive to capture the dynamics of each system in the meta-dataset for at least one (system-dependent) choice of its tunable parameters. However, this base architecture is highly over-parameterized, meaning it has significantly more degrees of freedom than the actual factors governing variability across systems in the meta-dataset. Given the meta-dataset and the base architecture, our goal is to meta-learn a *low-dimensional manifold* embedded in the base architecture's parameter space such that the base architecture, with parameters restricted to this manifold, can still describe all the systems within the class of interest. Restricting the search space to this manifold effectively reduces the degrees of freedom of the base architecture, thereby enhancing generalization of the estimated models.

Conceptually, our approach is inspired by classic approaches for model-based meta-learning, which aim at *optimizing the algorithm* that estimates a model within a given base architecture (Hospedales et al., 2021). For instance, model-agnostic meta-learning (MAML) (Finn et al., 2017) seeks an *initialization* of the base architecture's parameters such that, with a few steps of gradient descent starting from that point, effective adaptation to each dataset in the meta-dataset is achieved. This meta-learning problem is tackled through bilevel optimization, where the initial parameters are optimized at the outer level, while adaptation to the specific datasets is performed via gradient descent at the inner level. Recently, the MAML framework has been adopted for the design of adaptive model-based control schemes in Richards et al. (2023). Unlike MAML, our methodology meta-learns a low-dimensional manifold in the base architecture's parameter space instead of providing an initialization for optimization, and can be related to learning an effective *inductive bias* to facilitate the system identification procedure.

An existing meta-learning algorithm that even more closely resembles our approach is latent embedding optimization (LEO) (Rusu et al., 2019), which meta-learns a low-dimensional embedding of the datasets. Compared to LEO (and MAML), we circumvent the need to perform bilevel optimization by introducing an *auxiliary neural network* that directly maps each dataset onto its low-dimensional representation, bypassing the need for gradient-based inner-loop optimization.

Meta-learning has been recently applied for system identification, among other works, in Chakrabarty et al. (2025), Du et al. (2023), Forgione et al. (2023). However, Du et al. (2023), Forgione et al. (2023) build a unified *meta-model* for the entire meta-dataset, relying on the *in-context* learning capabilities of transformer architectures. Conversely, (Chakrabarty et al., 2025) is based on the MAML algorithm and its variants; unlike the current approach, MAML-like approaches involve a bilevel training procedure. While this is effective, a major drawback is the complexity of the bilevel training procedure. To avoid this, workarounds such as implicit MAML (iMAML) have been investigated by Yan et al. (2024), which leverages the implicit function theorem to simplify the inner-loop training. Also, (Chakrabarty et al., 2025) demonstrates the potential of almost-no-inner-loop (ANIL), which restricts the inner-loop training to a small set of user-chosen layers or components in the network. In other words, ANIL restricts model adaptation to a linear subspace, namely the span of the weights in the layers that are chosen to remain learnable. In contrast, our method specifies only the dimensionality of the adaptation manifold, while allowing its structure to be meta-learned from data. As a result, our approach can recover ANIL as a very special case, but is not limited to it. Indeed, it can also discover more general manifolds, potentially involving nonlinear parameter combinations rather than a manually predefined subset of layers.

2. Framework

2.1. Data stream and distribution

In line with the meta-modeling framework introduced in Forgione et al. (2023), we assume to have access to a large (possibly unlimited) collection $\mathcal{D}$ of datasets referred to as the meta-dataset, described by

$$\mathcal{D} = \{D^{(i)}\}_{i\in\mathbb{N}}, \text{ where } D^{(i)} = (\mathbf{u}^{(i)}_{0:N-1}, \mathbf{y}^{(i)}_{0:N-1}). \tag{1}$$

Each dataset $D^{(i)}$ comprises N inputs $\mathbf{u}^{(i)}_k \in \mathbb{R}^{n_u}$ and corresponding targets $\mathbf{y}^{(i)}_k \in \mathbb{R}^{n_y}$, $k = 0, 1, \dots, N-1$. In the context of system identification, $\mathbf{y}^{(i)}_{0:N-1}$ is the sequence of outputs generated by a dynamical system $S^{(i)}$ driven by the input sequence $\mathbf{u}^{(i)}_{0:N-1}$. This results in a causal dependency $\mathbf{u}^{(i)}_{0:k} \to \mathbf{y}^{(i)}_k$. In the case of static regression, the dependency is simply $\mathbf{u}^{(i)}_k \to \mathbf{y}^{(i)}_k$.

In the following, we adopt the shorthand notation $\mathbf{u} = \mathbf{u}_{0:N-1}$, $\mathbf{y} = \mathbf{y}_{0:N-1}$ to denote the sequence of N input and output samples, respectively, contained in a dataset. The inputs $\mathbf{u}^{(i)}$ of each dataset are assumed to be drawn from a distribution with density $p(\mathbf{u})$, while the dependency $\mathbf{u}^{(i)} \to \mathbf{y}^{(i)}$ is different, yet related across datasets. Formally, the collection of datasets (1) may be interpreted as samples drawn from a distribution (in the space of datasets) with density:

$$p(D) = p(\mathbf{u}, \mathbf{y}) = p(\mathbf{u}) \int_z p(z) p(\mathbf{y} \mid \mathbf{u}, z)\, dz, \tag{2}$$

where $z \in \mathbb{R}^{n_z}$ is an unknown or hidden variable that characterizes the data-generating mechanism $p(\mathbf{y} \mid \mathbf{u}, z)$. The variable z should be interpreted as an abstract representation of the factors of variation underlying the data-generating process across datasets. In the considered synthetic data generation setting, z may correspond to a subset of the simulator's physical parameters, intentionally randomized to capture plausible variations of the input/output system dependency expected in the real-world scenarios of interest, such as different masses attached to the end-effector of a robot, environment temperature in a chemical reaction, etc.

Remark 1. The latent variable z only captures the aspects of the data-generating system that vary across datasets. For instance, in the case of the robot where only the end effector mass changes, a scalar z representing such mass suffices, even though the robot itself is characterized by many other physical parameters, that, however, remain fixed for all simulations. Those non-varying parameters may be thought of as part of the functional form of the conditional distribution $p(\mathbf{y} \mid \mathbf{u}, z)$ in (2).

2.2. Base architecture

Consider the base architecture given by the black-box model structure

$$\hat{\mathbf{y}} = F(\mathbf{u}; \theta), \ \theta \in \mathbb{R}^{n_\theta}, \tag{3}$$

which we assume is prescribed to describe the input/output dependency $\mathbf{u} \to \mathbf{y}$ for each dataset D in the support of $p(D)$ with

practically no bias, for some system-dependent (unknown) value of the parameter θ. We argue that this assumption is not particularly restrictive given the availability of expressive function approximators such as deep neural networks, that, with appropriate selection of hyperparameters, can describe wide classes of nonlinear functions with high accuracy, and have been reported to exhibit competitive performance across different system identification benchmarks (Champneys et al., 2024).

The model structure (3) is typically characterized by a number of parameters $n_\theta \gg n_z$ for two reasons. First, (3) has a black-box structure, not necessarily tailored to a specific class of system. For example, accurate description of robot dynamics with $l = 8$ links may require thousands of black-box parameters ($n_\theta \approx 10^3$) with neural-network models (Bemporad, 2025), whereas a physics-based representation only requires $10l = 80$ terms (Gaz et al., 2019). Second, just a subset of these physical parameters may be varying across the simulations within the meta-dataset. For instance, as in the example of Remark 1, if the only physical parameter varying across dataset is the robot's end-effector mass, then $n_z = 1$ suffices. Consequently, black-box architectures used to describe physical systems are often heavily over-parameterized.

For a given dataset $D = (\mathbf{u}, \mathbf{y})$, the optimal parameter that best describes the input/output relation is typically obtained by minimizing a supervised learning criterion:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}(\theta)\right), \tag{4}$$

where the training loss $\mathcal{L}(\cdot,\cdot)$ penalizes the mismatch between measurements $\mathbf{y}$ and predictions $\hat{\mathbf{y}}(\theta) = F(\mathbf{u}, \theta)$. A common choice for $\mathcal{L}(\cdot,\cdot)$ is the mean squared error $\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_{k=0}^{N-1} \left\|\mathbf{y}_k - \hat{\mathbf{y}}_k\right\|_2^2$. A regularization term $r(\theta)$ may be added to (4). A typical choice is $r(\theta) = \rho\|\theta\|_2^2$, where $\rho \geq 0$ controls the regularization strength.

However, when the number of parameters n_θ and/or the number of samples N in the training dataset D are large, learning the dynamics represented by F can be computationally intensive. Conversely, when N is small, achieving good *generalization* to unseen data becomes difficult.

2.3. Meta-learning manifolds in the parameter space

To address the computational and generalization challenges of system identification within the base architecture (3), we consider the following problem:

Problem 1. Given the base architecture (3) and a collection $\mathcal{D}$ of datasets sampled from $p(D)$, learn a lifting function $P : \mathbb{R}^{n_\phi} \mapsto \mathbb{R}^{n_\theta}$, with $n_\phi \ll n_\theta$ that defines a n_ϕ-dimensional manifold $M = \{\theta \in \mathbb{R}^{n_\theta} \colon \theta = P(\phi),\ \phi \in \mathbb{R}^{n_\phi}\}$ such that each dataset $D = (\mathbf{u}, \mathbf{y})$ in the support of $p(D)$ can be described accurately by a model $\hat{\mathbf{y}} = F(\mathbf{u}; P(\phi))$ for some coordinate vector $\phi \in \mathbb{R}^{n_\phi}$.

We label Problem 1 as a *meta-learning* problem, as our goal is to learn a unique manifold M for the *entire meta-dataset*. Once the manifold is learned, the user can fit a model to any dataset $D = (\mathbf{u}, \mathbf{y})$ contained in the support of $p(D)$ by solving a lower-dimensional system identification problem with $n_\phi \ll n_\theta$ optimization variables.

Problem 1 admits at least one solution for $n_\phi \geq n_z$. Indeed, we hypothesize in Section 2.2 that each data-generating system in the distribution (equivalently, each $z \in \mathbb{R}^{n_z}$) is represented by (at least) a $\theta \in \mathbb{R}^{n_\theta}$. Then, there exists a function $P^\dagger : \mathbb{R}^{n_z} \mapsto \mathbb{R}^{n_\theta}$ mapping each data-generating system (characterized by a $z \in \mathbb{R}^{n_z}$) to a parameter $\theta \in \mathbb{R}^{n_\theta}$ of the full black-box model structure.[1] Therefore, the model structure $\hat{\mathbf{y}} = F(\mathbf{u}; P^\dagger(z)),\ z \in \mathbb{R}^{n_z}$, solves Problem 1 with $P = P^\dagger$, $\phi = z$. However, the function $P^\dagger$, as well as n_z, are not generally known a priori. Moreover, there is no guarantee that $P^\dagger$ is sufficiently regular to be learnable from data with practical numerical algorithms.

Notwithstanding this potential limitation, and therefore accepting that the resulting solution may be suboptimal, we set the dimensionality $n_\phi \ll n_\theta$ of the reduced model parameter space and describe the function P by a vector $\gamma \in \mathbb{R}^{n_\gamma}$ of tunable parameters, denoting it by P_γ. We assume that each function P_γ in the *user-chosen* search space $\{P_\gamma : \gamma \in \mathbb{R}^{n_\gamma}\}$ induces a differentiable manifold of dimension n_ϕ embedded in $\mathbb{R}^{n_\theta}$. For instance, P_γ could be parameterized as an affine function or as a neural network with smooth activation functions.

As we will detail in Section 3, our meta-learning goal is to learn a lifting parameter vector $\hat{\gamma} \in \mathbb{R}^{n_\gamma}$ for the entire set $\mathcal{D}$ to define the reduced-complexity architecture

$$\hat{\mathbf{y}} = F(\mathbf{u}; P_{\hat{\gamma}}(\phi)),\ \phi \in \mathbb{R}^{n_\phi}, \tag{5}$$

given by the functional composition of the base architecture with the *learned* lifting function $P_{\hat{\gamma}} : \mathbb{R}^{n_\phi} \mapsto \mathbb{R}^{n_\theta}$.

Based on the learned reduced-complexity architecture (5), and given a new dataset $D^* = (\mathbf{u}^*, \mathbf{y}^*)$ that is contained in the support of $p(D)$, the following reduced-complexity optimization problem

$$\hat{\phi} = \arg\min_{\phi} \mathcal{L}(\mathbf{y}^*, F(\mathbf{u}^*, P_{\hat{\gamma}}(\phi))) \tag{6}$$

solves the identification problem of fitting a model to D^*. Given the reduced number n_ϕ of optimization variables, problem (6) can generally be solved more efficiently than (4), e.g., by taking advantage of quasi-Newton (Bemporad, 2023, 2025) and even global optimization approaches that may not be computationally feasible in the full-complexity structure (3). Furthermore, the risk of overfitting is mitigated by the lower number of degrees of freedom of the architecture (5) with respect to (3).

3. Meta-learning procedure

In line with the standard meta-learning setting, we split each dataset $D^{(i)}$ into training and test portions, i.e.,

$$D^{(i)} = (\mathbf{u}_{\text{tr}}^{(i)}, \mathbf{y}_{\text{tr}}^{(i)}), (\mathbf{u}_{\text{te}}^{(i)}, \mathbf{y}_{\text{te}}^{(i)}),\ i = 1, 2, \ldots, \tag{7}$$

where both $\mathbf{y}_{\text{tr}}^{(i)}$ and $\mathbf{y}_{\text{te}}^{(i)}$ are generated by the *same* system $S^{(i)}$ for different input signals $\mathbf{u}_{\text{tr}}^{(i)}$ and $\mathbf{u}_{\text{te}}^{(i)}$, respectively. Such a data split allows one to learn an optimal manifold M such that, when training is performed on $(\mathbf{u}_{\text{tr}}^{(i)}, \mathbf{y}_{\text{tr}}^{(i)})$, performance measured on test data $(\mathbf{u}_{\text{te}}^{(i)}, \mathbf{y}_{\text{te}}^{(i)})$ is good.

A direct way of learning an optimal parametric function $P_{\hat{\gamma}}$ is to solve the following bilevel optimization problem:

$$\hat{\gamma} = \arg\min_{\gamma} \mathbb{E}_{p(D)}\left[\mathcal{L}(\mathbf{y}_{\text{te}}, \hat{\mathbf{y}}_{\text{te}})\right] \tag{8a}$$

$$\hat{\mathbf{y}}_{\text{te}} = F\left(\mathbf{u}_{\text{te}};\ P_\gamma(\hat{\phi})\right) \tag{8b}$$

$$\hat{\phi} = \text{Alg}(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}}, \gamma), \tag{8c}$$

where the inner-level optimization $\text{Alg}(\cdot,\cdot,\gamma)$ is the reduced system identification algorithm:

$$\text{Alg}(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}}, \gamma) = \arg\min_{\phi} \mathcal{L}(\mathbf{y}_{\text{tr}}, F(\mathbf{u}_{\text{tr}};\ P_\gamma(\phi))). \tag{9}$$

Essentially, (8c) is a parametric version of (6) where the coefficients γ of the lifting function P_γ are free. In fact, γ is optimized at the outer level (8a) to minimize the expected test set loss, according to the nested structure of the overall optimization problem (8).

[1] The relationship $z \to \theta$ might be one-to-many. The function $P^\dagger$ here arbitrarily chooses one of the multiple equivalent solutions.

3.1. Encoder–decoder architecture

To circumvent the numerical intricacies related to the nested optimization problem formulation (8), we replace the inner optimization with an auxiliary neural network E_ψ, which may be interpreted as a *learned* black-box system identification algorithm:

$$\hat{\gamma}, \hat{\psi} = \arg\min_{\gamma,\psi} \mathbb{E}_{p(D)} \overbrace{\left[\mathcal{L}(\mathbf{y}_{\text{te}}, \hat{\mathbf{y}}_{\text{te}})\right]}^{J(\gamma,\psi)} \tag{10a}$$

$$\hat{\mathbf{y}}_{\text{te}} = F\left(\mathbf{u}_{\text{te}};\ P_\gamma(\hat{\phi})\right) \tag{10b}$$

$$\hat{\phi} = E_\psi(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}}). \tag{10c}$$

The pair (10b)–(10c) may also be seen as an encoder–decoder architecture. The encoder E_ψ ingests the input–output pair $(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ and outputs reduced-complexity parameters $\hat{\phi}$. The decoder, defined by the functional composition $F(\cdot, P_{\hat{\gamma}}(\cdot))$, first lifts the reduced parameters $\hat{\phi}$ to the full-complexity dimension n_θ, and then applies the base architecture with the lifted parameters to the input $\mathbf{u}_{\text{te}}$ to predict the output $\hat{\mathbf{y}}_{\text{te}}$. Fig. 1 is a graphical representation of (10b)–(10c) focusing on this encoder–decoder interpretation.

To solve (10), we replace the intractable expectation over $p(D)$ by a Monte Carlo average. In other words, $J(\gamma, \psi)$ in (10) is approximated as a mean over b sampled datasets $D^{(i)}$ from the meta-dataset $\mathcal{D}$ leading to:

$$\tilde{J}(\gamma, \psi) = \frac{1}{b}\sum_{i=1}^{b} \mathcal{L}\left(\mathbf{y}_{\text{te}}^{(i)}, F\left(\mathbf{u}_{\text{te}}^{(i)}, P_\gamma\left(E_\psi(\mathbf{u}_{\text{tr}}^{(i)}, \mathbf{y}_{\text{tr}}^{(i)})\right)\right)\right) \tag{11}$$

The loss $\tilde{J}(\gamma, \psi)$ is minimized with standard gradient-based optimization. In case the datasets $D^{(i)}$ are resampled at each iteration, b plays the role of the batch size.

We emphasize that the main objective of solving (10) is to find the lifting function $P_{\hat{\gamma}}$ to be plugged into (5), and, therefore, the manifold M of dimension n_ϕ that best reduces the complexity of identifying a model over the dataset distribution $p(D)$. Once $P_{\hat{\gamma}}$ is learned, given a new dataset $D^* = (\mathbf{u}^*, \mathbf{y}^*)$, the user may prefer to solve the identification problem (6) explicitly, instead of relying on the learned $E_{\hat{\psi}}$. In fact, (6) can be solved numerically using standard algorithms, typically with (local) convergence guarantees. These algorithms are arguably more trustworthy than the neural network $E_{\hat{\psi}}$, which is a black-box from datasets to vectors of dimension n_ϕ and only learns to approximate algorithmic behavior through the meta-learning procedure, thereby inheriting the usual generalization and representation challenges of learning-based methods. Nonetheless, the learned algorithm $E_{\hat{\psi}}$ may be used to initialize an iterative optimization algorithm for (6), thereby speeding up the reduced-complexity model learning task.

3.2. Specialization for neural state-space models

We now specialize the methodology presented in the previous sections for datasets $D^{(i)}$ that are input/output time series generated by dynamical systems, and the base architecture is a state-space model:

$$x_{k+1} = f(x_k, u_k; \theta) \qquad y_k = g(x_k, u_k; \theta). \tag{12}$$

In (12), f and g are learnable mappings, such as feed-forward neural networks. In this case, the output sequence $\mathbf{y}$, given an input sequence $\mathbf{u}$, also depends on an initial state x_0. Formally, we have: $\hat{\mathbf{y}} = F(\mathbf{u}, x_0; \theta)$. Thus, in applying the previously mentioned methodology, care has to be taken in the initialization of this state, notably for a correct computation of the loss in (10a).

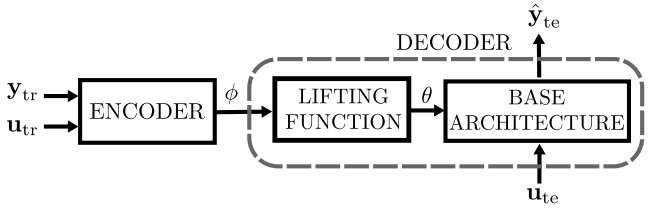


Fig. 1. The autoencoder architecture for reduced-complexity architecture learning.

A few practical solutions have been introduced in recent system identification literature to tackle the initial state issue. Similarly to Beintema et al. (2023), Bemporad (2025), an additional neural network can be included to process the first n_{init} samples of $(\mathbf{u}_{\text{te}}, \mathbf{y}_{\text{te}})$ and provide an estimate $\hat{x}_{n_{\text{init}}}$ of the state at time step n_{init}. Starting from this state estimate, the loss (10a) may be computed, excluding the first n_{init} samples from the loss evaluation. We note that, in the case of synthetic data, the two datasets $(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}})$, $(\mathbf{u}_{\text{te}}, \mathbf{y}_{\text{te}})$ may be chosen to have the *same* (random) initial state. In this case, the encoder network E_ψ may be designed to provide initial state estimates, together with the reduced-complexity model parameters, thereby simplifying the architecture.

Finally, an even simpler approach may be followed when the systems to be modeled are known to have a fading memory. In this case, the state can be initialized to an arbitrary value (such as zero), and the effect of the system's "natural response" can be removed from the loss by ignoring the contribution from the first n_{skip} samples, where n_{skip} is selected large enough. In the numerical examples presented in Section 5, for simplicity, problem (10) is instantiated relying on such a "zero" state initialization scheme. More advanced schemes may be integrated with a moderate effort.

4. Probabilistic bound on the performance gap

In this section, we derive a probabilistic bound on the gap between training the full-complexity model (3) and the reduced-complexity architecture (5).

For each dataset D drawn from the dataset distribution $p(D)$, let $(D_{\text{tr}}, D_{\text{te}})$ be the split of D that partitions D into a training part $D_{\text{tr}} = (\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ and a test part $D_{\text{te}} = (\mathbf{u}_{\text{te}}, \mathbf{y}_{\text{te}})$. Note that D_{tr} and D_{te} are generated by the same underlying system (i.e., they share the same latent z in (2)). Given D_{tr}, let the trained parameters of the full- and reduced-complexity models be, respectively,

$$\hat{\theta}(D_{\text{tr}}) = \arg\min_\theta \mathcal{L}(\mathbf{y}_{\text{tr}}, F(\mathbf{u}_{\text{tr}}; \theta))\,, \tag{13}$$

$$\hat{\phi}(D_{\text{tr}}) = \arg\min_\phi \mathcal{L}\big(\mathbf{y}_{\text{tr}}, F(\mathbf{u}_{\text{tr}}, P_{\hat{\gamma}}(\phi))\big)\,. \tag{14}$$

We define the fit index over a set of length N as

$$\text{fit}(\mathbf{y}, \hat{\mathbf{y}}) = 100 \cdot \left(1 - \frac{\sqrt{\sum_{k=0}^{N-1}\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right)^2}}{\sqrt{\sum_{k=0}^{N-1}\left(\mathbf{y}_k - \bar{\mathbf{y}}\right)^2}}\right)(\%), \tag{15}$$

where $\bar{\mathbf{y}}$ is the sample average of the measured output $\mathbf{y}$.

Then, we define the *performance gap* as the difference between the test fit index obtained using the full- and the reduced-complexity models, i.e.,

$$\Psi(D) = \text{fit}\Big(\mathbf{y}_{\text{te}}, F(\mathbf{u}_{\text{te}}; \hat{\theta}(D_{\text{tr}}))\Big) - \text{fit}\Big(\mathbf{y}_{\text{te}}, F(\mathbf{u}_{\text{te}}; P_{\hat{\gamma}}(\hat{\phi}(D_{\text{tr}})))\Big)\,. \tag{16}$$

The gap $\Psi(D)$ thus measures the difference in the performance on a test set between full- and reduced-complexity models, identified based on the same training set D_{tr}. Note that, If $\Psi(D) > 0$, the full-complexity model achieves a higher fit than the reduced-complexity model, i.e., training in the full space yields better performance on D_{te}. Conversely, if $\Psi(D) < 0$, the reduced-complexity model performs better on D_{te}.

Theorem 1 defined below establishes a probabilistic upper bound for the performance gap Ψ. Such a bound is constructed using *m additional* independent and identically distributed (i.i.d.) datasets $\{D^{(i)}\}_{i=1}^{m}$ extracted from $p(D)$, independent of the datasets used to learn γ and ψ in the previous meta-learning phase.

Theorem 1 (*Bound on the Performance Gap*). *Let $\{D^{(i)}\}_{i=1}^{m}$ be i.i.d draws from $p(D)$, independent of the datasets previously used in the meta-learning phase. For each i let $\left(D_{\text{tr}}^{(i)}, D_{\text{te}}^{(i)}\right)$ be the corresponding training/test split. Define $\Psi^{(i)} = \Psi\left(D^{(i)}\right)$ and let $\Psi_{(1)} \leq \cdots \leq \Psi_{(m)}$ be the ordered statistics of the samples $\{\Psi^{(i)}\}_{i=1}^{m}$, i.e., the sequence of $\{\Psi^{(i)}\}_{i=1}^{m}$ arranged in non-decreasing order.*

For any $\alpha, \delta \in (0, 1)$, set
$\varepsilon_m = \sqrt{\frac{\log(2/\delta)}{2m}}$ *and* $k^\star = \left\lceil m\left(1 - \alpha + \varepsilon_m\right)\right\rceil$.
Suppose m is large enough so that $k^\star \leq m$ (equivalently, $\alpha \geq \varepsilon_m$). Let $D^{\text{new}} \sim p(D)$ be a new draw independent of $\{D^{(i)}\}_{i=1}^{m}$, with training/test split $\left(D_{\text{tr}}^{\text{new}}, D_{\text{te}}^{\text{new}}\right)$, and define

$$\Psi^{\text{new}} = \Psi\left(D^{\text{new}}\right). \tag{17}$$

Then, with probability at least $1-\delta$ over the draw of the training/test pairs $\{D^{(i)}\}_{i=1}^{m}$, it holds:

$$\mathbb{P}\left(\Psi^{\text{new}} \leq \Psi_{(k^\star)}\right) \;\geq\; 1 - \alpha, \tag{18}$$

where $\mathbb{P}$ denotes the probability measure induced by the sampling distribution $p(D)$.

Proof. The proof builds upon the same rationale adopted in Busetto et al. (2027, Section 4.2) in the context of amortized optimization to derive bounds on the gap between solving an optimization problem and its reduced-dimensional counterpart learned from data. A proof is provided in the following for self-consistency of the paper.

First, since the datasets $\{D^{(i)}\}_{i=1}^{m}$ are i.i.d. draws from $p(D)$ and the mapping $D \mapsto \Psi(D)$ is deterministic (given the training/test procedure), the performance gaps $\{\Psi^{(i)}\}_{i=1}^{m}$ are i.i.d. random variables.

Let G be the *cumulative distribution function* (CDF) of $\Psi(D)$ and let $\hat{G}_m$ be its empirical CDF defined as:

$$\hat{G}_m(t) \;=\; \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\left(\Psi^{(i)} \leq t\right), \qquad t \in \mathbb{R}, \tag{19}$$

with $\mathbf{1}$ being the indicator function.

By the Dvoretzky–Kiefer–Wolfowitz–Massart inequality (Massart, 1990), the following condition holds with probability at least $1 - \delta$ over the sets $\{D^{(i)}\}_{i=1}^{m}$:

$$\sup_t \left|\hat{G}_m(t) - G(t)\right| \;\leq\; \varepsilon_m. \tag{20}$$

For $\alpha \in (0, 1)$, let $Q_{1-\alpha} = \inf\{t : G(t) \geq 1 - \alpha\}$ be the $(1-\alpha)$-quantile of the distribution of $\Psi(D)$. Let us choose $k^\star = \left\lceil m\left(1 - \alpha + \varepsilon_m\right)\right\rceil \leq m$. Then,

$$\hat{G}_m\left(\Psi_{(k^\star)}\right) \;\geq\; \frac{k^\star}{m} \;\geq\; 1 - \alpha + \varepsilon_m, \tag{21}$$

where the first inequality follows from the definition of the empirical CDF $\hat{G}_m$ in (19), evaluated at the $k^\star$-th order statistic $\Psi_{(k^\star)}$, and the second inequality follows from the definition of $k^\star$.

Combining (20) and (21), with probability at least $1 - \delta$ we obtain

$$G\left(\Psi_{(k^\star)}\right) \;\geq\; \hat{G}_m\left(\Psi_{(k^\star)}\right) - \varepsilon_m \;\geq\; 1 - \alpha, \tag{22}$$

so that $Q_{1-\alpha} \leq \Psi_{(k^\star)}$ by the definition of $Q_{1-\alpha}$.

Finally, for a new draw $D^{\text{new}} \sim p(D)$, we have

$$\mathbb{P}\left(\Psi^{\text{new}} \leq Q_{1-\alpha}\right) = 1 - \alpha \tag{23}$$

by definition of $Q_{1-\alpha}$. Since we proved that $Q_{1-\alpha} \leq \Psi_{(k^\star)}$, then

$$\mathbb{P}\left(\Psi^{\text{new}} \leq \Psi_{(k^\star)}\right) \;\geq\; \mathbb{P}\left(\Psi^{\text{new}} \leq Q_{1-\alpha}\right) = 1 - \alpha, \tag{24}$$

which proves (18). □

Note that, although Theorem 1 is stated for the performance gap defined in (16), the same argument applies to other performance metrics, for instance, the loss $\mathcal{L}$ of a new dataset D^{new}.

Remark 2. To ensure that the index $k^\star$ is well-defined, one needs $m \geq k^\star$. This yields the requirement $m \geq \dfrac{\log(2/\delta)}{2\alpha^2}$, showing that the minimum number of datasets m needed to obtain certification bounds scales logarithmically with $1/\delta$ and quadratically with $1/\alpha$. Nevertheless, it is worth noting that the m datasets used to compute the bound on the performance gap can often be generated in simulation, by sampling i.i.d. synthetic datasets from the same dataset distribution $p(D)$ employed in the meta-training phase (Section 3).

Remark 3. It is worth clarifying the different roles played by the datasets D used in the meta-learning phase (Section 3) and in the certification phase (Section 4). In the *meta-training* phase, at each optimization step of the meta-objective 11, we sample b datasets from $p(D)$ to approximate the expectation in (10a) by the empirical average in 11. This procedure is used to learn the shared representation, namely the encoder E_ψ and the lifting function P_γ. In the *certification* phase, after (γ, ψ) have been learned and fixed, we draw an independent collection of m previously unseen i.i.d. datasets from $p(D)$ to compute the performance gap $\Psi(D)$ and derive probabilistic guarantees (Theorem 1).

5. Numerical illustration

We apply our approach to the Bouc–Wen system identification benchmark (Noel & Schoukens, 2016). The software was developed in Python using the JAX automatic differentiation library and it is available at the repository https://github.com/forgi86/sysid-neural-manifold. All computations were performed on a server equipped with an AMD EPYC 9754 CPU and an Nvidia 4090 GPU.

5.1. The Bouc–Wen benchmark

The Bouc–Wen benchmark entails the vibrations of a 1-DoF hysteretic mechanical system. The system dynamics are described in state-space form by:

$$\begin{bmatrix} \dot{p}(t) \\ \dot{v}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{1}{m_L}\left(u(t) - k_L p(t) - c_L v(t) - z(t)\right) \\ \alpha v(t) - \beta\left(\gamma |v(t)||z(t)|^{\nu-1} + \delta v(t)|z(t)|^{\nu}\right) \end{bmatrix} \tag{25a}$$

$$y(t) = p(t), \tag{25b}$$

where $p(t)$ (m) is the measured output position; $v(t)$ (m/s) is the unmeasured velocity; $z(t)$ (N) is an unmeasured hysteretic force term; and $u(t)$ (N) is the known input force. The nominal value of the model coefficients are reported in the first row of Table 1.

The benchmark provides a test dataset where the input $u(t)$ is a random-phase multisine of length 8192 sampled at frequency

Table 1
Coefficients of the Bouc–Wen system. Nominal values (first row) and minimum/maximum values in the dataset distribution (second/third row).

	m_L	c_L	k_L	α	β	γ	δ	ν
nom	2	10	5.0×10^4	5.0×10^4	1000	0.8	−1.1	1
min	1	5	2.5×10^4	2.5×10^4	500	0.5	−1.5	1
max	3	15	7.5×10^4	7.5×10^4	4500	0.9	−0.5	1

$f_s = 750$ Hz that excites the frequency range $[5, 150]$ Hz and has a root mean square of 50 N. System (25) is simulated with its nominal coefficients and the test output $y(t) = p(t)$ is provided noise-free. A "default" training dataset is also provided, comprising 40960 samples generated by exciting the system with a multisine input signal with the same characteristic as the test input. In the training dataset, the output is corrupted by an additive Gaussian noise with bandwidth $[0 - 375]$ Hz and amplitude $8 \cdot 10^{-3}$. Moreover, the benchmark provides simulation code and specifies that the user is free to choose different input signals to excite the system and generate arbitrary training datasets.

In this paper, along the lines of Chakrabarty et al. (2025), we extend the original scope of the benchmark (Noel & Schoukens, 2016) and use the simulator to generate additional datasets with perturbed coefficient values, aiming at finding a reduced-complexity architecture that provides high performance over the entire *class* of Bouc—Wen systems with parameters comprised in the min–max range reported in Table 1. Importantly, meta-learning of the reduced-complexity architecture is performed exclusively on these additional datasets. The training and test datasets provided by the benchmark developers and generated using the nominal coefficients are only used at a later stage to *assess* the performance of the meta-learned architecture. This strategy prevents the risk of *meta-overfitting* to a specific Bouc–Wen configuration. In meta-learning terminology, the new datasets generated by the authors of this work constitute the *meta-training* set, while the benchmark-provided datasets serve as the *meta-test* set.

5.2. Base architecture

As base architecture, we consider the neural state-space:

$$x_{k+1} = Ax_k + Bu_k + N_f(x_k, u_k; W_f) \tag{26a}$$

$$y_k = Cx_k + N_g(x_k; W_g), \tag{26b}$$

where $n_u = 1$, $n_y = 1$, $n_x = 3$, N_f, and N_g are feed-forward neural networks with one hidden layer and 16 hidden units with weights W_f and W_g, respectively, and `tanh` activation function; and A, B, and C are matrices of proper size. The base architecture parameters are thus $\theta = \text{vec}(W_f, W_g, A, B, C) \in \mathbb{R}^{n_\theta}$, with $n_\theta = 244$.

5.3. Metrics and baselines

We evaluate performance in terms of the fit index (15). Fitting the model on the default 40960-length training dataset, Schoukens (2021) reported the state-of-the-art result on this benchmark (fit = 98.9%) on the test set. We could match their results by minimizing, in line with the full-complexity training setting (4), the MSE loss of the base architecture (26) initialized with random parameters over 40000 iterations of `AdamW` with learning rate 10^{-3} followed by 10000 iterations of `BFGS`. The training time was 1.4 h. We also train as a baseline a third-order linear model that provides fit = 77.2%, underlining the relevance of non-linearities in this benchmark.

5.4. Full-complexity model training

In this section, we assess the performance of full-complexity model learning in the low-data regime. To this aim, we fit full-complexity models according to (4) over sequences of length $L = 100, 200, 400, 500, 600, 800, 1000, 2000, 3000, 4000, 5000$. As in the previous section, the base architecture is trained from scratch starting from a random initialization, thus without leveraging any prior information from the dataset distribution $p(D)$. For each length L, we execute a Monte Carlo study involving $n_{\text{mc}} = 100$ subsequences of length L extracted from the official 40960-length dataset, training distinct models on each sequence.

For optimization, we use the same `AdamW` + `BFGS` approach with the same settings detailed in the previous section. As shown in Fig. 2, full-complexity model learning proves highly inefficient in the low-data regime. In particular, for sequence length $L \leq 400$, the median fit is below the linear baseline. The median performance consistently improves for increasing sequence length, with average fit above 97% for $L \geq 3000$ samples. However, even for $L = 2000, 3000, 4000$, and 5000, a small number of runs (2, 1, 1, and 1, respectively) fail due to numerical issues in the `BFGS` solver. While these issues might be mitigated with a careful tuning of the solver settings, they highlight the challenges of full-complexity model training.

Fig. 3 illustrates the time required to train in parallel the 100 models vs. the sequence length L. We report both the run time for the 40000 `AdamW` iterations and the total training time, which includes the 10000 `BFGS` iterations. The results show that `BFGS` dominates the overall training time.

5.5. Meta-learning the reduced-complexity architecture

To enable higher performance in the low-data regime, we learn a manifold with $n_\phi = 20$ parameters, embedded within the full-complexity architecture (26) characterized by $n_\theta = 244$ parameters, by applying the methodology described in Section 3. To this aim, we introduce the parameterized lifting network $P_\gamma : \mathbb{R}^{n_\phi} \mapsto \mathbb{R}^{n_\theta} = V\phi + \theta_{\text{bias}}$ with $V \in \mathbb{R}^{n_\theta \times n_\phi}, \theta_{\text{bias}} \in \mathbb{R}^{n_\theta}$, which expands the reduced-complexity dimension n_ϕ to the full-complexity n_θ. Its tunable parameters are collected in the vector $\gamma = \text{vec}(V, \theta_{\text{bias}}) \in \mathbb{R}^{n_\gamma}$, with $n_\gamma = (n_\phi + 1)n_\theta = 5124$.

To learn the optimal lifting network parameters $\hat{\gamma}$, we adopt the encoder–decoder methodology and solve the optimization problem (10), using the Monte Carlo approximation $\hat{J}(\gamma, \psi)$ in 11 of the loss $J(\gamma, \psi)$. The meta-dataset $\mathcal{D}$ is obtained by randomly sampling Bouc–Wen systems with coefficients uniformly distributed within the min–max ranges in Table 1, and feeding as input random multisine sequences of length $N = 2000$ with the same properties as the default training input.

The encoder E_ψ is a sequence-to-vector architecture obtained by combining a bi-directional Gated Recurrent Unit (GRU) with a feed-forward neural network. The GRU is configured with $n_u + n_y = 2$ input channels and a single hidden layer with $n_h = 128$ hidden units. It processes the sequences $(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ and produces a sequence of activations $\mathbf{h}_{0:N-1}, \mathbf{h}_i \in \mathbb{R}^{n_h}$. This sequence is reduced to a single vector h_m through *average pooling*: $h_m = \frac{1}{N}\sum_i \mathbf{h}_i$ and further processed by the feed-forward neural network configured with $n_h = 128$ inputs, a hidden layer with 128 units, and an output layer with $n_\phi = 20$ units, and `tanh` activation functions. Overall, the network E_ψ has $n_\psi = 136340$ parameters corresponding to the GRU and the feed-forward neural network weights. It takes as input the sequences $(\mathbf{u}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ and produces as output a single vector of size n_ϕ.

For gradient-based minimization of the loss 11, the `Adam` algorithm is applied with a batch size $b = 128$ and initial learning rate of $2 \cdot 10^{-4}$, decreased to $2 \cdot 10^{-5}$ with cosine scheduling over

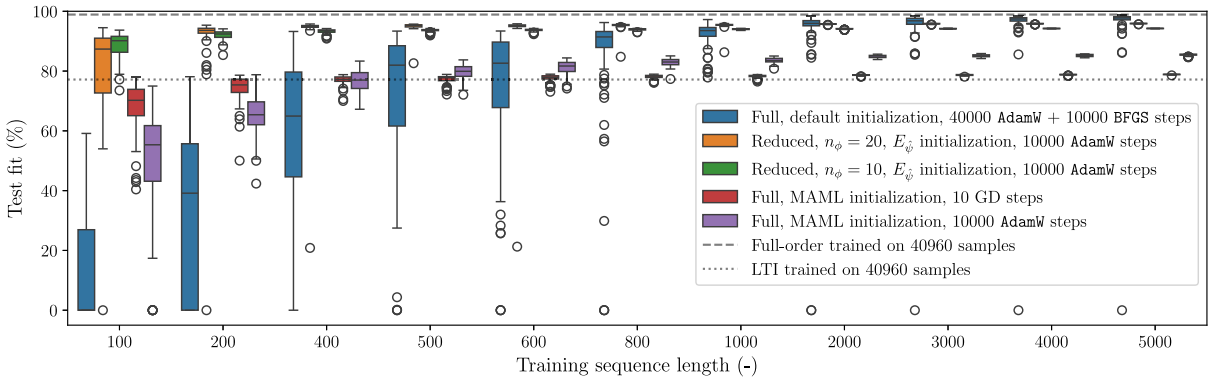


Fig. 2. Full- and reduced-complexity models: test fit vs. training sequence length L.

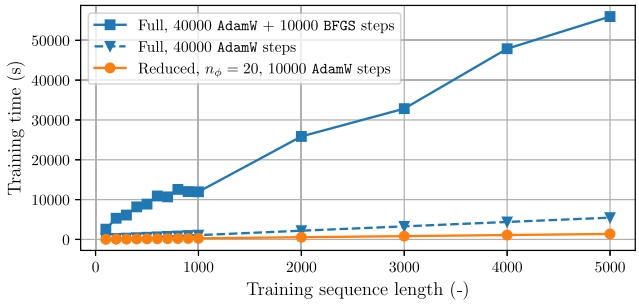


Fig. 3. Training time vs. training sequence length L.

200000 iterations. The datasets $D^{(i)}$ are resampled at each meta-optimization iteration. Thus, overall, 256 million trajectories are used for meta training. The optimization takes approximately 25.3 h. As a result, we obtain the reduced-complexity architecture (5) and, as a byproduct, the encoder network $E_{\hat{\psi}}$ which can be used to initialize the reduced-complexity model learning (6).

5.6. Reduced-complexity model training

We repeat the Monte Carlo study of Section 5.4 involving $n_{\text{mc}} = 100$ datasets per sequence length using the reduced-complexity architecture meta-learned in Section 5.5, instead of the base architecture.

In all the runs, the reduced-complexity MSE loss is minimized according to (6). Specifically, minimization is performed over 10000 iterations of AdamW with learning rate 10^{-3}, using the encoder network $E_{\hat{\psi}}$ to obtain an initial guess of ϕ. Results in Fig. 2 show that the reduced-complexity training is significantly more effective than the full-complexity one for sequences of length up to 2000 samples. Very good results are obtained with just 500 samples (median fit of 95.2%). For training sequences longer than 2000 samples, full-complexity training generally yields superior results, given the higher flexibility of the model structure.

We remark that none of the reduced-complexity training runs exhibited numerical difficulties, and the obtained performance is largely insensitive to the choice of the optimizer and its settings. This underscores the stability and reliability of the reduced-complexity model learning problem.

The run time of reduced-complexity training with 10000 AdamW iterations is shown in Fig. 3 and amounts to approximately 25% of the 40000 AdamW iterations executed in the full-complexity case. This highlights that the cost of a *single* optimization iteration remains nearly identical for both cases, as evaluating the lifting function P is negligible compared to computing the training loss. Indeed, P is a lightweight static function evaluated only once per iteration, while the neural state-space (3) is executed iteratively over the L time steps.

Due to the lower problem dimensionality and to the near-optimal parameter initialization using $E_{\hat{\psi}}$, 10000 AdamW iterations are sufficient for reduced-complexity optimization. It is worth reminding that this efficiency gain is achieved after the expensive meta-training phase, which however has to be executed only once for the whole class of identification problems described by the probability distribution $p(D)$.

To analyze the sensitivity of the methodology to the choice of the manifold dimensionality, we repeat the reduced-complexity architecture meta-learning and the Monte Carlo study for $n_\phi = 10$. Results in Fig. 2 highlight that this choice leads to the best results for the very short sequences of length $L = 100$, while it leads to worse performance against $n_\phi = 20$ for all other lengths.

5.7. Empirical evaluation of the performance gap bound

In this section, we numerically assess the relevance of Theorem 1, namely that the bound (18) on the performance gap between the full- and reduced-complexity training schemes holds with a (high) probability of at least $1 - \delta$. To this end, we set $\delta = 0.01$, $\alpha = 0.05$, and perform $m = 1900$ full and reduced trainings on m datasets of length $L = 500$ and $L = 3000$ extracted from $p(D)$. For each run, we record the performance gap statistic Ψ defined in (16). The results are sorted in non-decreasing order and denoted by $\{\Psi_{(i)}\}_{i=1}^{m}$.

Given the chosen values of α, δ, and m, we obtain $\epsilon_m = 0.037$ and $k^\star = 1876$. The element $\Psi_{(k^\star)}$ in the ordered set $\{\Psi_{(i)}\}_{i=1}^{m}$ then serves as a probabilistic upper bound on the performance gap, as established by (18). Fig. 4 illustrates this bound for both sequence lengths, together with the empirical cumulative distribution function (CDF) of Ψ obtained from an additional 100 Monte Carlo simulations. We observe that the empirical $(1 - \alpha)$-quantile of Ψ in these additional simulations is indeed bounded by $\Psi_{(k^\star)}$, thereby illustrating the practical relevance of the theoretical result. For $L = 500$, the bound $\Psi_{(k^\star)} = -1.5\%$ is negative, which confirms that reduced-complexity models achieve better performance on short sequences. In contrast, for $L = 3000$, the bound $\Psi_{(k^\star)} = 2.7\%$ is positive, and upper bounds (with probability at least $1-\delta$) the potential performance loss of reduced-complexity model training.

5.8. Comparison with MAML

We perform a comparison with the MAML meta-learning approach. In the meta-learning phase, MAML solves an explicit bi-level optimization, similar to (8), to find an initial full-complexity parameter $\theta_{\text{maml}} \in \mathbb{R}^{n_\theta}$ for subsequent gradient-based optimization. In its inner level, MAML executes plain gradient descent

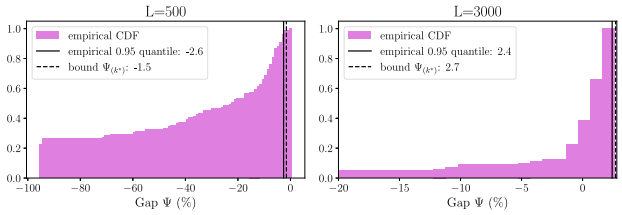


Fig. 4. Performance gap (fit index difference) between full and reduced models: empirical cumulative distribution function (pink), empirical 0.95 quantile (solid black line) and theoretical upper bound $\Psi_{(k^\star)}$ on such quantile. Results for $L = 500$ (left panel) and $L = 3000$ (right panel).

initialized from θ_{maml}. In our implementation, MAML is configured with 10 inner gradient-descent steps at an inner learning rate of 0.1. Note that these are typical settings for this algorithm, and that the number of inner gradient-descent steps is limited by computational and memory requirements. The outer MAML optimization is solved with the same settings used for reduced-complexity architecture meta learning, i.e. with 200000 `Adam` iterations and cosine learning rate scheduling. The MAML meta-learning optimization takes 3.3 days.

The meta-learned parameter θ_{maml} is then used as an initializer in the Monte Carlo study. Specifically, full-complexity trainings (4) starting from θ_{maml} are executed both with 10 steps of plain gradient descent (matching the MAML inner-level meta-training, that is however limited by computational reasons) and with 100000 iterations of `AdamW` with learning rate $1 \cdot 10^{-3}$ (matching the reduced-complexity model trainings). All results are reported in Fig. 2. We observe that MAML initialization with 10 GD steps is rather effective for the short sequences (though not as effective as training in the low-dimensional meta-learned manifold), but its performance barely surpasses the linear baseline for longer ones. Conversely, training with 10000 `AdamW` steps reaches a higher performance for the longer sequences, but it is less effective on the shorter ones. Overall, the reduced-complexity model training approach presented in this work dominates the MAML implementation.

The lower performance of MAML may be partly due to computational limits on the number of inner-optimization steps and the fact that its hyper-parameters were not extensively tuned for this benchmark. Moreover, the considered setting of an infinite meta-dataset likely mitigates the weaker inductive bias of our approach based on auxiliary black-box maps, which in turn may mask one of its drawbacks and contribute to its observed advantage over MAML.

6. Conclusions

This paper introduced a meta-learning framework for system identification that discovers a low-dimensional manifold within the parameter space of an over-parameterized neural architecture. Our results on the Bouc–Wen benchmark highlight the potential of this approach, particularly in extremely data-scarce scenarios.

Several extensions of this framework can be explored. While our methodology was introduced in the context of a black-box base architecture, it could be extended to models, constraints, and fitting criteria informed by physical knowledge. Additionally, integrating variational learning techniques into the meta-learning process could enable the discovery of not only a deterministic low-dimensional representation but also a prior distribution over this space.

Finally, it would be interesting to study meta-training in the regime of small *meta*-datasets, that is, when only a limited number of trajectories from a class of systems is available. In this context, statistical efficiency at the meta-learning procedure becomes critical, and a probabilistic formulation may play a key role by regularizing it through a suitable prior over the low-dimensional representation, thereby improving sample efficiency.

References

Beintema, G. I., Schoukens, M., & Tóth, R. (2023). Deep subspace encoders for nonlinear system identification. *Automatica*, *156*, Article 111210.

Bemporad, A. (2023). Training recurrent neural networks by sequential least squares and the alternating direction method of multipliers. *Automatica*, *156*, Article 111183.

Bemporad, A. (2025). An L-BFGS-B approach for linear and nonlinear system identification under ℓ_1- and group-lasso regularization. *IEEE Transactions on Automatic Control*, *70*(7), 4857–4864.

Busetto, R., Mejari, M., Forgione, M., Bemporad, A., & Piga, D. (2027). Learning low-dimensional embeddings for black-box optimization. *Automatica*, *195*, Article 113325.

Chakrabarty, A., Wichern, G., Deshpande, V. M., Vinod, A. P., Berntorp, K., & Laughman, C. R. (2025). Meta-learning for physically-constrained neural system identification. *Neurocomputing*, *651*, Article 130945. http://dx.doi.org/10.1016/j.neucom.2025.130945.

Champneys, M., Beintema, G. I., Tóth, R., Schoukens, M., & Rogers, T. J. (2024). Baseline results for selected nonlinear system identification benchmarks. *IFAC-PapersOnLine*, *58*(15), 474–479. http://dx.doi.org/10.1016/j.ifacol.2024.08.574, 20th IFAC Symposium on System Identification.

Deshpande, V. M., Chakrabarty, A., Vinod, A. P., & Laughman, C. R. (2023). Physics-constrained deep autoencoded Kalman filters for estimating vapor compression system states. *IEEE Control Systems Letters*, *7*, 3483–3488.

Du, Z., Balim, H., Oymak, S., & Ozay, N. (2023). Can transformers learn optimal filtering for unknown systems? *IEEE Control Systems Letters*, *7*, 3525–3530.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135). PMLR.

Forgione, M., & Piga, D. (2021). Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, *59*, 69–81.

Forgione, M., Pura, F., & Piga, D. (2023). From system models to class models: An in-context learning paradigm. *IEEE Control Systems Letters*.

Gaz, C., Cognetti, M., Oliva, A., Giordano, P. R., & De Luca, A. (2019). Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, *4*(4), 4147–4154.

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(9), 5149–5169.

Massart, P. (1990). The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality. *The Annals of Probability*, *18*(3), 1269–1283.

Moradi, S., Jaensson, N., Tóth, R., & Schoukens, M. (2023). Physics-informed learning using Hamiltonian neural networks with output error noise models. *IFAC-PapersOnLine*, *56*(2), 5152–5157.

Nghiem, T. X., Drgoňa, J., Jones, C., Nagy, Z., Schwan, R., Dey, B., Chakrabarty, A., Di Cairano, S., Paulson, J. A., Carron, A., et al. (2023). Physics-informed machine learning for modeling and control of dynamical systems. In *American control conference* (pp. 3735–3750).

Noel, J.-P., & Schoukens, M. (2016). Hysteretic benchmark with a dynamic nonlinearity. In *Workshop on nonlinear system identification benchmarks* (pp. 7–14).

Pillonetto, G., Aravkin, A., Gedon, D., Ljung, L., Ribeiro, A. H., & Schön, T. B. (2025). Deep networks for system identification: a survey. *Automatica*, *171*, Article 111907.

Richards, S. M., Azizan, N., Slotine, J.-J., & Pavone, M. (2023). Control-oriented meta-learning. *The International Journal of Robotics Research*, *42*(10), 777–797. http://dx.doi.org/10.1177/02783649231165085.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., & Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *International conference on learning representations*. URL https://openreview.net/forum?id=BJgklhAcK7.

Schoukens, M. (2021). Improved initialization of state-space artificial neural networks. In *2021 European control conference* (pp. 1913–1918). IEEE.

Yan, J., Chakrabarty, A., Rupenyan, A., & Lygeros, J. (2024). MPC of uncertain nonlinear systems with meta-learning for fast adaptation of neural predictive models. In *2024 IEEE 20th international conference on automation science and engineering* (pp. 1910–1915). IEEE.



Marco Forgione received his M.Sc. degree in Computer Engineering from the University of Pavia, Italy, in 2009, and his Ph.D. in Systems and Control Engineering from Delft University of Technology, The Netherlands, in 2014. He was a Postdoctoral Researcher at École Centrale de Lyon, France (2014–2015), and a Control Engineering Consultant at Whirlpool EMEA, Italy (2015–2018). He is currently a Senior Researcher at the IDSIA Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland. His main research interests are at the intersection of system identification, meta-learning, and control.



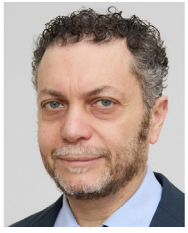
Ankush Chakrabarty received the Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, in 2016. He held a research position as a Postdoctoral Fellow at Harvard University (2016–2018), where he worked on embedded predictive control and convolutional networks for automated insulin delivery. Since then, he is at Mitsubishi Electric Research Labs (MERL), focusing on machine learning (ML) and uncertainty-aware decision-making for smart buildings, autonomous vehicles, robotics, and factory automation. His research interests include ML for system identification, time-series, predictive control, and knowledge transfer mechanisms for real-world applications. He has authored over 100 peer-reviewed publications and holds more than 25 patents. He is a Senior Member of the IEEE since 2016, and has an Erdős number of 4.



Dario Piga received his Ph.D. in Systems and Control Engineering from Politecnico di Torino, Italy. He is currently Professor at the Dalle Molle Institute for Artificial Intelligence (IDSIA USI-SUPSI) in Lugano, Switzerland, where he is the founder and head of the LEarning for Optimization and coNtrol (LEON) research group. His research interests lie at the intersection of system identification, control theory, machine learning, and nonlinear optimization. He coordinates and serves as Principal Investigator in several research projects and collaborates with international companies on the development of innovative AI-based modeling and control solutions for applications in manufacturing, transportation, robotics, biomedical engineering, and the chemical industry. He is an Associate Editor of Automatica and Chair of the IEEE Control Systems Society Technical Committee on System Identification and Adaptive Control.



Matteo Rufolo received the M.Sc. degree cum laude in Mathematical Engineering from the Polytechnic University of Turin, Italy, in 2023. Since 2024, he has been pursuing a Ph. D. degree in Scientific Machine Learning at the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland. His research focuses on meta- learning, sequential modeling, and efficient neural architectures, with an emphasis on Transformer-based in-context learning and state-space models for dynamical systems. He develops reduced-order models and hardware-aware machine learning methods for fast adaptation and real-time inference. Previously, he was a Research Intern at CERN (2022–2023), working on high-performance simulations and inverse problems for the Large Hadron Collider. He also served as a Machine Learning Research Intern at the Basque Center for Applied Mathematics (BCAM), focusing on robust classification algorithms.



Alberto Bemporad received his M.S. degree cum laude in Electrical Engineering (1993) and his Ph.D. in Control Engineering (1997) from the University of Florence, Italy. He held research positions at Washington University in St. Louis (1996–1997) and ETH Zurich (1997–2002). He served at the University of Siena (1999–2009) and at the University of Trento (2010–2011). Since 2011, he has been Full Professor at the IMT School for Advanced Studies Lucca, where he was Director from 2012 to 2015. He co-founded ODYS S.r.l. in 2011, a company specialized in industrial model predictive control systems, and has held visiting appointments at Stanford University, the University of Michigan, and Zhejiang University. He has authored over 400 publications and 21 patents in model predictive control, hybrid systems, numerical optimization, nonlinear system identification, and automotive control, and co-developed the Model Predictive Control Toolbox (MathWorks) for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001–2004 and Chair of the IEEE CSS Technical Committee on Hybrid Systems during 2002–2010. He is an IEEE Fellow (since 2010) and IFAC Fellow (since 2025). His awards include the IFAC High-Impact Paper Award (2011–14), IEEE CSS Transition to Practice Award (2019), SAE Environmental Excellence in Transportation Award (2021), the Beale–Orchard-Hays Prize (2024), and an ERC Advanced Grant (2024).