

In-Context Learning for System Identification: Probabilistic Formulation, Long-Sequence Handling, and Benchmark Validation*

Matteo Rufolo^{1,2}

Dario Piga¹

Filippo Pura¹

Marco Forgione¹

¹SUPSI, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland

²USI, Faculty of Informatics, Lugano, Switzerland

{matteo.rufolo, dario.piga, filippo.pura, marco.forgione}@supsi.ch

Please cite this version of the paper:

M. Rufolo, D. Piga, F. Pura and M. Forgione “In-Context Learning for System Identification: Probabilistic Formulation, Long-Sequence Handling, and Benchmark Validation,” in *Automatica*, 2026. In press.

You may use the following bibtex entry:

```
@article{rufolo2026incontext,
  title={In-Context Learning for System Identification: Probabilistic Formulation, Long-Sequence Handling, and Benchmark Validation},
  author={Rufolo, Matteo and Piga, Dario and Pura, Filippo and Forgione, Marco},
  journal={Automatica},
  year={2026},
  note={in press}
}
```

Abstract

Traditional system identification typically involves two steps: initially, training data is processed to estimate a model of the system of interest; then, the model is used to predict outputs for new input queries. In contrast, recent in-context learning approaches learn a meta model that maps directly from training data and query inputs to outputs in a zero-shot fashion, bypassing explicit model estimation. This approach leverages a meta model trained offline, on large sets of synthetically generated data, to emulate the behavior of a system identification algorithm. In this work, we advance this meta-learning framework for system identification in three directions. First, we develop a unified probabilistic meta-identification architecture that combines a Transformer-based encoder–decoder with tailored mechanisms for handling long training sequences, enabling calibrated predictions over long horizons. Second, we show how a pre-trained meta model can act as an uncertainty-aware synthetic data generator for classical system identification, providing task-specific data that regularize parametric models when real data are scarce. Third, we systematically evaluate

*This work was partly supported by the Hasler Foundation project “FALCON - Foundation models for zero-shot adaptive control and prediction for dynamical systems” and as a part of NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 51NF40_225155).

the framework on standard nonlinear system identification benchmarks and on a challenging out-of-distribution scenario, demonstrating competitive accuracy with state-of-the-art methods and strong generalization, including cases where input-only adaptation is sufficient to handle significant distribution shifts.

Keywords: Meta learning; Artificial neural networks; System identification; Synthetic data.

1 Introduction

1.1 System identification and meta learning

System identification (SYSID) provides data-driven methods to construct mathematical models of dynamical systems from measured input–output data. Given a dataset, the goal is to estimate a model that captures the underlying behavior with sufficient accuracy for tasks such as simulation, prediction, and control design. Over the past decades, a broad range of identification techniques has been developed, ranging from prediction-error and subspace methods for linear to deep-learning-based approaches for nonlinear model learning; see, e.g., [16, 42, 3, 25, 39, 28]. In the classical setting, each identification task is posed for a single system, and a model is fitted using exclusively the data available for that system.

Traditional identification limitations In standard practice, the estimation pipeline relies heavily on expert intervention. The practitioner must select a suitable model structure (e.g., linear *vs* nonlinear, time-invariant *vs* time-varying, state-space *vs* input/output representation, etc.), choose an identification approach (e.g., prediction error methods, subspace methods, kernel-based approaches, etc.), and tune hyper-parameters in the numerical optimization routines used to solve the identification problem (e.g., learning rate in gradient-based methods, regularization weights, stopping criteria to prevent overfitting, etc.). These design and tuning choices are typically repeated from scratch for each new system of interest.

In many applications, however, the goal is not to model a single isolated plant, but to repeatedly identify related systems (for example, members of a given benchmark family or class of processes). In such settings, traditional pipelines treat each system independently: for every new dataset, the modeling and tuning steps described above are repeated, and a separate model is estimated from scratch. This per-system workflow does not exploit information gained from previously identified systems and can be computationally demanding when many systems must be modeled. Moreover, classical approaches typically yield a different model for each system, which complicates downstream tasks such as deploying a shared controller across a fleet of systems or transferring prior knowledge to new tasks. While uncertainty-aware identification methods and advanced long-sequence models exist, they are not generally designed to provide class-level, zero-shot predictions for new systems from a short context trajectory.

Proposed meta-learning approach To address these challenges, we adopt a meta-learning viewpoint in which the basic object of interest is a *class* of dynamical systems rather than a single plant. Building on the in-context system identification paradigm introduced in [18], this paper presents a *novel* SYSID paradigm inspired by meta learning techniques [20]. We consider a meta model that is trained once, offline, on a large synthetic corpus generated from a distribution over systems. At inference time, this pre-trained meta model receives a context input–output sequence from a new system and a query input, and directly predicts the corresponding output in a zero-shot fashion eliminating the need to implement any system identification algorithm and avoiding the selection of model structure and hyper-parameters.

This class-level meta-modeling addresses the limitations discussed above in the following ways:

- *Reuse of knowledge across systems:* a single meta model is shared by all systems in the class, so information gained during offline training is amortized across tasks instead of being discarded after each identification.
- *Reduced per-system design effort:* architecture and training hyper-parameters are chosen once at the class level. For a new system, the meta model is applied without revisiting model-structure selection or hyper-parameter tuning for each trajectory.
- *Zero-shot prediction from short contexts:* given a limited context from a new system, the meta model can produce output predictions for new inputs without any task-specific gradient-based adaptation, in contrast to classical pipelines that require solving a new optimization problem per system.

This enables knowledge reuse across systems and reduces the need to repeatedly perform full identification procedures—including structure and hyper-parameter selection—for every new trajectory.

We emphasize that addressing the above limitations comes at its own cost. In particular, the proposed meta-learning paradigm relies on a substantial offline pre-training phase on synthetic data, and a performance gap with carefully tuned classical system identification methods may still appear when abundant task-specific data are available or when strict requirements on parsimonious models for control design are imposed. These trade-offs, and the regimes in which the meta-modeling approach is most advantageous, are examined in detail in the subsequent sections.

The proposed meta learning approach is analogous to *Large Language Models* (LLMs) like GPT-4, LLaMA, Gemini, and DeepSeek, which are pre-trained on vast text corpora to understand and generate text. Similarly, our meta learning approach enables users to generate output predictions by providing context and a related query, leveraging knowledge acquired in pre-training to efficiently handle various system identification tasks.

1.2 Related Literature

Meta learning, also referred to as learning to learn, is a branch of machine learning introduced in the late 1980s [36] that has gained significant momentum in recent years. In a broad sense, the goal of meta learning is to enhance model and algorithm efficiency by leveraging experience from multiple related tasks. While standard machine learning trains models for a single task, meta learning approaches aim to capture common patterns and similarities across tasks, enabling rapid adaptation to new ones with minimal data and computational resources.

One of the most popular meta learning algorithms is *Model-Agnostic Meta Learning* (MAML) [15], which aims to find model parameters that can quickly adapt to new tasks. MAML has proven particularly effective in few-shot learning scenarios across various domains and applications, including computer vision for few-shot image classification tasks [2], reinforcement learning to accelerate policy learning in new environments [15], materials science for predicting material properties and accelerating materials discovery [46], as well as in system identification, optimization and control [21, 8, 33, 7].

Another instance of meta learning is represented by in-context learning, where large models are trained on diverse and extensive datasets, enabling them to process a context—a demonstration of the task at hand—along with a query input, and directly predict the corresponding output without further training [22, 12]. In this framework, zero-shot prediction emerges as a powerful feature of in-context learning, where models leverage their extensive pre-training on diverse datasets to adapt to new tasks without requiring additional fine-tuning or retraining. Typical application of in-context learning include Natural Language Processing [45], vision [43], speech recognition [11], and robotics [44], just to cite a few. In the field of system identification, in-context learning has recently demonstrated promising results for zero-shot prediction,

leveraging knowledge gained from estimating the behavior of systems similar to the target one [18, 34, 14].

1.3 Paper Contribution

This paper builds upon the preliminary work on in-context learning for system identification introduced in [18] and further developed in [30, 35]. The aim is to provide a unified framework that integrates and extends these previous contributions, establishing a comprehensive and probabilistic formulation of in-context learning for system identification. Within this unified setting, the present work introduces the following technical contributions:

- **Extended methods for long-context processing:** A novel ensemble-based strategy with probabilistic weighting is proposed to efficiently handle long input–output context sequences. Compared with the patching approach of [35], the new method achieves higher in-distribution accuracy and reduced meta-training time, at the cost of a slight increase in meta-inference time.
- **Probabilistic synthetic data generation:** The data generation strategy of [30] is reformulated within the probabilistic framework, enabling the production of uncertainty-aware synthetic datasets for training classical system identification algorithms.
- **Benchmark validation:** Where [18, 35] focused mainly on simulated scenarios, we systematically evaluate the full proposed architecture on multiple nonlinear system identification benchmarks from the nonlinear benchmark initiative [1], demonstrating robustness and generalization on real experimental data.
- **Model adaptation study:** We assess the framework in an out-of-distribution (OOD) scenario in which both the dynamical systems and the input signals differ from those used during meta model training. We show that fine-tuning by matching only the inputs, while keeping the family of training systems fixed, achieves performance comparable to full adaptation. This result is practically relevant for scenarios where the system dynamics are uncertain but the excitation signals, chosen by the user, are known in advance.

Taken together, these contributions go beyond our previous works [18, 30, 35] in several ways. First, the probabilistic formulation of synthetic data generation yields uncertainty-aware synthetic datasets and a principled mechanism to down-weight unreliable synthetic samples, addressing the growing need for more reliable synthetic data in machine learning. Second, the adaptation study shows that fine-tuning only the input side of the meta model can already recover most of the benefits of full adaptation, which is particularly relevant in applications where excitation signals can be designed but the system dynamics remain uncertain. Finally, by combining these elements with long-context processing in a single architecture and by carrying out a systematic evaluation on standard nonlinear benchmarks, including systems that differ from the WH training class, the paper validates the sim-to-real capabilities of the in-context approach and provides some of the first concrete tools for using meta-learning in system identification, considering the emerging interest in this research direction.

The rest of the paper is organized as follows. In Section 3, we review the meta modeling framework originally introduced in [18]. We first formulate a point estimation problem of system outputs as in [18], and then extend this formulation to the more general case of probabilistic predictions. Model architectures and training algorithms are discussed in Section 4, followed by novel approaches to manage long context sequences. The first one is based on sequence patching and inspired by the Vision Transformer [13] for image classification. The second one relies on an ensembling of the meta model predictions associated with different (short) context sequences. Section 5 details how the pre-trained meta model can generate synthetic data useful for classical

system identification to estimate a parametric model of a dynamical system of interest when the training data are scarce or insufficiently informative. Examples showcasing the effectiveness of the meta learning paradigm are reported in Section 6, including benchmark performance and model adaptation analyses. Concluding remarks are drawn in Section 7, along with directions for future research.

2 Notation

Vectors are denoted by lowercase letters (e.g., u). Subscripts indicate the time index (e.g., u_i). For compactness, $u_{a:b}$ denotes the sequence $\{u_a, u_{a+1}, \dots, u_b\}$. A superscript in parentheses, such as $u^{(i)}$, denotes the i -th sample drawn from a dataset or distribution. A superscript without parentheses, such as u^i , denotes the i -th subsequence extracted from a longer sequence. The same notation applies to the corresponding model outputs associated with each subsequence or sample. $\mathbb{E}_p[\cdot]$ denotes the expectation with respect to a probability distribution p . The Euclidean-norm of a vector u is denoted as $\|u\|$. Estimated quantities are indicated by a hat, e.g., $\hat{y}$ is an estimate of y . $\mathcal{N}(\cdot; \mu, \sigma^2)$ denotes a Gaussian probability density function with mean μ and variance σ^2 .

3 Meta learning framework

3.1 System Class

Let us consider a class $\mathcal{S}$ of dynamical systems. This class could correspond, for example, to Linear Time-Invariant (LTI) systems, fading memory nonlinear systems, or a more restricted subset of dynamics describing specific real-world systems. Examples of such restricted classes include the set of all robotic manipulators with a maximum number of degrees of freedom or the set of all vehicles with specific characteristics, to name just a few.

We assume to have access to a (potentially infinite) stream of input/output datasets generated by different systems $\{S^{(i)}, i = 1, 2, \dots\}$ extracted from a probability distribution $p(S)$ defined over the class $\mathcal{S}$. For each system $S^{(i)}$, a context dataset $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$ and a query dataset $(\tilde{u}_{1:n}^{(i)}, \tilde{y}_{1:n}^{(i)})$ are available, with $u_k, \tilde{u}_k \in \mathbb{R}^{n_u}$, $y_k, \tilde{y}_k \in \mathbb{R}^{n_y}$. These two datasets are obtained by processing two randomly generated input sequences $u_{1:m}^{(i)}, \tilde{u}_{1:n}^{(i)}$ through the same system $S^{(i)}$, starting from different random initial conditions.

The systems $S^{(i)}$ may be instances of a high-fidelity simulator of a process of interest, with physical parameter randomly distributed according to domain knowledge. Overall, we assume there is an underlying *distribution* $p(D)$ of datasets generated by the system distribution $p(S)$ and the random input distribution. The distribution $p(D)$ may not be explicitly known, but samples $\{D^{(i)} = (u_{1:m}^{(i)}, y_{1:m}^{(i)}), (\tilde{u}_{1:n}^{(i)}, \tilde{y}_{1:n}^{(i)}), i = 1, 2, \dots\}$ can be drawn from $p(D)$.

3.2 Classic system identification vs meta modeling

Classic system identification aims to estimate a different model for each system $S^{(i)}$ based on the available dataset $D^{(i)}$. A classic approach might require choosing a model structure and a learning algorithm, using the context data $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$. Then, the system state at a time step $c < n$ of the query may be estimated from the input/output samples $(\tilde{u}_{1:c}^{(i)}, \tilde{y}_{1:c}^{(i)})$ up to time c . By simulating the identified system model starting from the estimated initial condition and for the remaining query input sequence $\tilde{u}_{c+1:n}^{(i)}$, one could make predictions $\hat{y}_{c+1:n}^{(i)}$ of the output $y_{c+1:n}^{(i)}$:

$$\hat{y}_{c+1:n}^{(i)} = \mathcal{M}(\tilde{u}_{c+1:n}^{(i)}, \tilde{u}_{1:c}^{(i)}, \tilde{y}_{1:c}^{(i)}, u_{1:m}^{(i)}, y_{1:m}^{(i)}), \quad (1)$$

where $\mathcal{M}$ represents the combined identification, system state estimation, and simulation procedure applied to the dataset $D^{(i)}$.

In the meta learning approach discussed in this paper, a single meta model is estimated for all systems in the class $\mathcal{S}$. Given a context dataset generated by a specific system $S^{(i)}$ and a query input, the meta model directly provides an estimate of the corresponding query output. Specifically, we introduce a (black-box) map $\mathcal{M}_\phi$, parameterized by ϕ , which processes the same inputs as in (1):

$$\hat{y}_{c+1:n}^{(i)} = \mathcal{M}_\phi(\tilde{u}_{c+1:n}^{(i)}, \tilde{u}_{1:c}^{(i)}, \tilde{y}_{1:c}^{(i)}, u_{1:m}^{(i)}, y_{1:m}^{(i)}). \quad (2)$$

The map $\mathcal{M}_\phi$ is trained to behave as a meta model for the system class $\mathcal{S}$ by minimizing the expected simulation error over the dataset distribution $p(D)$ in an end-to-end supervised manner, i.e.,

$$\hat{\phi} = \arg \min_{\phi} \overbrace{\mathbb{E}_{p(D)} [\|\tilde{y}_{c+1:n} - \mathcal{M}_\phi(X)\|^2]}^{=J(\phi)}, \quad (3)$$

where $J(\phi)$ is the loss function, and $X = (\tilde{u}_{c+1:n}, \tilde{u}_{1:c}, \tilde{y}_{1:c}, u_{1:m}, y_{1:m})$ is introduced to compact the notation and represents all the input variables to the meta model $\mathcal{M}_\phi$.

The expectation over $p(D)$ is approximated by a Monte Carlo average over b system instances from the class $\mathcal{S}$ and corresponding measured context $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$ and query sequences $(\tilde{u}_{1:n}^{(i)}, \tilde{y}_{1:n}^{(i)})$, i.e.,

$$J(\phi) \approx \frac{1}{b} \sum_{i=1}^b \|\tilde{y}_{c+1:n}^{(i)} - \mathcal{M}_\phi(X^{(i)})\|^2. \quad (4)$$

3.3 Probabilistic Prediction

The meta model in (2) provides point estimates of the future outputs. While informative, these estimates lack the expressiveness of a full probabilistic description, which would also quantify predictive uncertainty. To address this limitation, the meta model $\mathcal{M}_\phi$ is modified to output, at each time step, the parameters of a probability distribution over future outputs.

Training of the meta model $\mathcal{M}_\phi$ is then performed with the aim of approximating the unknown conditional probability distribution $p(\tilde{y}_{c+1:n}|X)$ with a parameterized distribution $q_\phi(\tilde{y}_{c+1:n}|X)$. To this aim, we introduce the Kullback-Leibler (KL) divergence $\text{KL}(p \parallel q_\phi)$ between p and q_ϕ :

$$\text{KL}(p \parallel q_\phi) = \mathbb{E}_{p(\tilde{y}_{c+1:n}|X)} \left[\log \frac{p(\tilde{y}_{c+1:n}|X)}{q_\phi(\tilde{y}_{c+1:n}|X)} \right]. \quad (5)$$

The parameters ϕ defining $\mathcal{M}_\phi$ (and hence q_ϕ) are chosen to minimize the expected KL divergence over the conditioning variables X :

$$\begin{aligned} \mathbb{E}_{p(X)} [\text{KL}(p \parallel q_\phi)] &= \mathbb{E}_{p(D)} \left[\log \frac{p(\tilde{y}_{c+1:n}|X)}{q_\phi(\tilde{y}_{c+1:n}|X)} \right] \\ &= \overbrace{\mathbb{E}_{p(D)} [-\log q_\phi(\tilde{y}_{c+1:n}|X)]}^{J(\phi)} + K, \end{aligned} \quad (6)$$

where $J(\phi)$ is the loss function minimized with respect to ϕ , and K is a constant independent of ϕ .

As in the point-estimate formulation, the expected loss $J(\phi)$ is approximated by a Monte Carlo average:

$$J(\phi) \approx \frac{1}{b} \sum_{i=1}^b -\log q_\phi(\tilde{y}_{c+1:n}^{(i)}|X^{(i)}), \quad (7)$$

where $\tilde{y}_{c+1:n}^{(i)}$ and $X^{(i)}$ (with $i = 1, \dots, b$) are samples from $p(D)$ associated with the system $S^{(i)}$.

In this paper, we parameterize the distribution q_ϕ as a multivariate Gaussian with a diagonal covariance matrix. For notation simplicity, we present the case of single-output systems (i.e., $n_y = 1$). Thus, q_ϕ is factorized as:

$$q_\phi(\tilde{y}_{c+1:n}|X) = \prod_{k=c+1}^n q_\phi(\tilde{y}_k|X), \quad (8a)$$

$$q_\phi(\tilde{y}_k|X) = \mathcal{N}(\tilde{y}_k; \mu_k(X, \phi), \sigma_k(X, \phi)^2) \quad (8b)$$

The vectors of means $\mu_{c+1:n}$ and standard deviations $\sigma_{c+1:n}$, both of dimension $n - c$, are the outputs of the meta model $\mathcal{M}_\phi$ when fed by the query input $u_{c+1:n}$, initial input/output measurements $\tilde{u}_{1:c}$, $\tilde{y}_{1:c}$, and context data $u_{1:m}, y_{1:m}$:

$$\mu_{c+1:n}, \sigma_{c+1:n} = \mathcal{M}_\phi(\tilde{u}_{c+1:n}, \tilde{u}_{1:c}, \tilde{y}_{1:c}, u_{1:m}, y_{1:m}). \quad (9)$$

Here, $\mu_{c+1:n}$ corresponds to the previous point estimate $\hat{y}_{c+1:n}$, while $\sigma_{c+1:n}$ quantifies the associated predictive uncertainty.

Intuitively, the meta model $\mathcal{M}_\phi$ acts as a system identification algorithm that infers the dynamics of each system $S^{(i)}$ from its input/output context $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$. Using this inferred knowledge, it generates predictions and corresponding uncertainty estimates $(\mu_{c+1:n}, \sigma_{c+1:n})$ for the query sequence. Once trained, $\mathcal{M}_\phi$ can produce predictions for previously unseen systems using only an input/output context and a query input $\tilde{u}_{c+1:n}$, without the need to identify a dedicated model for each system.

3.4 Meta model architecture and training

The meta model is implemented using an encoder-decoder Transformer architecture similar to the one introduced in [40] for language translation, see Fig. 1 for a visual representation. With respect to [40], a recurrent patching network, described in details in Section 4.1, transforms the context data $(u_{1:m}, y_{1:m})$ into a shorter sequence $p_{1:M}$, with $p_i \in \mathbb{R}^{d_{\text{model}}}$ and $M = \frac{m}{L}$, where L is an integer greater than 1. The encoder backbone then processes $p_{1:M}$ and maps it into $\zeta_{1:M}$, with $\zeta_i \in \mathbb{R}^{d_{\text{model}}}$ (left side of Fig. 1). Then, the decoder processes the query input $\tilde{u}_{c+1:n}$ causally in time through masked self-attention, to generate the estimated distribution parameter $\mu_{c+1:n}, \sigma_{c+1:n}$, incorporating the knowledge of $\zeta_{1:M}$ through a cross-attention mechanism. Conceptually, the encoder can be interpreted as a system identification module, where $\zeta_{1:M}$ is an implicit representation of the system, and the decoder is a simulator.

Before the decoder block (right side of Fig. 1), the initial conditions (namely, initial input/output measurements) are processed by a linear layer that maps each time step from $\mathbb{R}^{n_u+n_y}$ to $\mathbb{R}^{d_{\text{model}}}$. Similarly, the query input $\tilde{u}_{c+1:n}$ undergoes a separate transformation from $\mathbb{R}^{n_u}$ to $\mathbb{R}^{d_{\text{model}}}$ (bottom-right blocks in Fig. 1). These processed sequences are concatenated to form a unique sequence of length n , with each element having dimension d_{model} . Then, a positional encoding is performed to preserve the temporal structure before the sequence enter the decoder backbone of the Transformer. The final layer of the decoder (upper-right block in Fig. 1) outputs both the mean $\mu_{c+1:n}$ and standard deviation $\sigma_{c+1:n}$ of the probability distribution of the system output at each future time step $c+1, \dots, n$.

Training of $\mathcal{M}_\phi$ is performed via stochastic (or mini-batch) gradient-descent, minimizing the loss $J(\phi)$ in (7). At each iteration, a batch of b systems $S^{(i)}$ (with $i = 1, \dots, b$) is drawn from the class $p(S)$ and the corresponding dataset $D^{(i)}$ is generated to compute the loss.

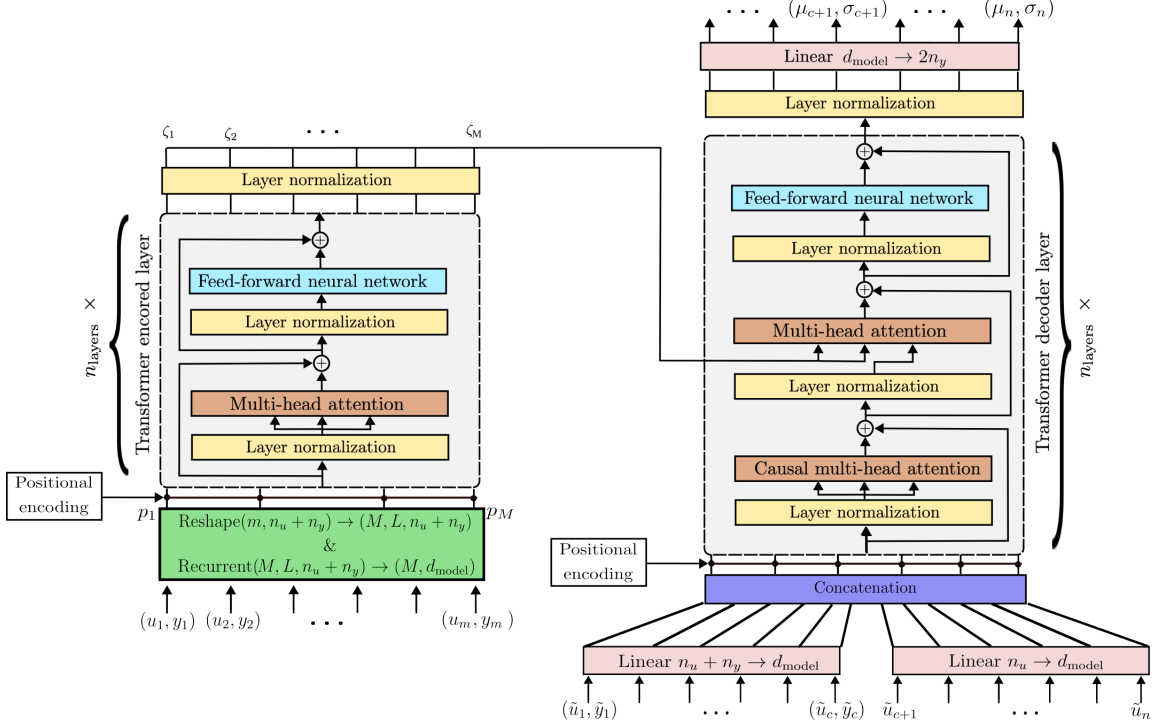


Figure 1: Encoder-decoder Transformer for in-context system identification. The main differences with respect to the original Transformer architecture [40] are highlighted in blue and pink. Additionally, an extra layer preceding the encoder backbone, referred to as recurrent patching neural network and highlighted in green, is incorporated to manage long context sequences.

4 Handling long sequences

An intrinsic limitation of the Transformer architecture is that the attention mechanism has a computational complexity that grows quadratically with the sequence length. This results in a limitation on the length of context sequences that the meta model $\mathcal{M}_\phi$ can efficiently handle. In this section, we propose two different approaches to address these limitations: one is applied during training, and the other during inference.

4.1 Patching

The patching approach, originally introduced in [13] for image processing, was later adapted for time series data in [26]. This method partitions the context sequence into smaller sub-sequences, referred to as *patches*, which are individually processed by a *patching network* to reduce the time dimensionality. This approach significantly reduces the computational cost of training by shortening the sequence length before processing by the Transformer. Specifically, the input/output context sequence $(u_{1:m}, y_{1:m})$ is divided into M non-overlapping patches,¹ each of length $L = \frac{m}{M}$ and dimension $n_u + n_y$. These patches are then processed by a recurrent neural network (RNN), which maps each patch to a single vector of dimension d_{model} . The resulting *patch embedding* sequence, $p_{1:M}$, is enriched with positional encoding to take into account temporal order information before being passed to the encoder backbone (bottom-left green block in Fig. 1). A visual representation of this approach is shown in Fig. 2.

This approach reduces the sequence length processed by the multi-head attention mechanism by a factor of L , leading to a computational complexity reduction of a factor L^2 for the training of the meta model $\mathcal{M}_\phi$.

¹For simplicity, we assume m is a multiple of M .

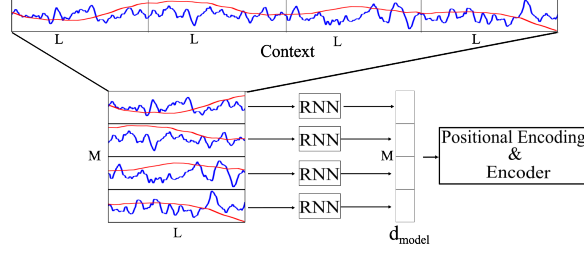


Figure 2: Visual representation of the implemented patching approach. For clarity, a single-input ($n_u = 1$, blue) single-output ($n_y = 1$, red) system is depicted

4.2 Ensembling

Differently from the patching strategy, the ensembling method described in this section is applied only during the inference phase, i.e., after meta model training. The approach is based on averaging multiple predictions to improve robustness. This idea builds on the well-established principle that predictive performance can often be improved through ensembling, which combines the outputs of multiple models rather than relying on a single one [5, Chapter 14].

We consider the case where the Transformer has been trained on context sequences of length m , but a longer sequence $(u_{1:m'}, y_{1:m'})$ with $m' > m$ is available from the real system at inference time. From this longer sequence, we extract shorter overlapping subsequences $\{(u_{1:m}^i, y_{1:m}^i)\}_{i=1}^s$ of length m . Considering all overlapping sub-sequences, $s = m' - m + 1$. Each subsequence $(u_{1:m}^i, y_{1:m}^i)$ is provided as context to the pre-trained Transformer, along with a fixed query input, to generate multiple predictions for the same query output, i.e.,

$$\mu_{c+1:n}^i, \sigma_{c+1:n}^i = \mathcal{M}_\phi(\tilde{u}_{c+1:n}, \tilde{u}_{1:c}, \tilde{y}_{1:c}, u_{1:m}^i, y_{1:m}^i), i = 1, \dots, s, \quad (10)$$

where $\mu_{c+1:n}^i$ and $\sigma_{c+1:n}^i$ represent the predicted mean and uncertainty for the query output based on the i -th context subsequence $(u_{1:m}^i, y_{1:m}^i)$.

A single prediction is then obtained by combining the s predictions through a weighted average. Specifically, at a generic query time step k , the ensembled prediction is computed as:

$$\mu_k = \sum_{i=1}^s \omega_k^i \mu_k^i. \quad (11)$$

where ω_k^i are non-negative weights that determine the contribution of each prediction. The goal is to select the weights to obtain an unbiased estimate of $\tilde{y}_k$ with minimum variance.

From (8b), we may interpret μ_k^i as a “noisy” estimate of the true unknown output $\tilde{y}_k$:

$$p(\mu_k^i) = \mathcal{N}(\mu_k^i; \tilde{y}_k, (\sigma_k^i)^2). \quad (12)$$

Consequently, the ensembled mean μ_k in (11) is also Gaussian, with mean $\bar{\mu}_k = \tilde{y}_k \sum_{i=1}^s \omega_k^i$. To ensure that μ_k is an unbiased estimator of $\tilde{y}_k$, the weights must satisfy the constraint:

$$\sum_{i=1}^s \omega_k^i = 1. \quad (13)$$

Furthermore, assuming that μ_k^i are independent of each others for all $i = 1, \dots, s$, the variance $\sigma_{\mu_k}^2$ of μ_k can be written:

$$\sigma_{\mu_k}^2 = \sum_{i=1}^s (\omega_k^i \sigma_k^i)^2. \quad (14)$$

Under constraint (13), it can be proven that the variance $\sigma_{\mu_k}^2$ in (14) is minimized by the weights:

$$\omega_k^i = \frac{(\sigma_k^i)^{-2}}{\sum_{j=1}^s (\sigma_k^j)^{-2}}. \quad (15)$$

A detailed derivation of (15) is reported in the appendix.

Note that, in the case of σ_k^i equal to each other for all $i = 1, \dots, s$, then $\omega_k^i = \frac{1}{s}$ and μ_k in (11) is given by an unweighted ensemble average.

5 Synthetic data generation for system identification

In this section, we discuss how the pre-trained meta model can be leveraged for classical system identification, where the goal is to estimate a (parsimonious) model G of a dynamical system S_0 based on observed data.

The use of synthetic data in this context is motivated by regimes in which the available real training dataset D_{tr} is small or weakly informative (e.g., limited excitation bandwidth or short experiments). In such cases, flexible parametric models G trained only on D_{tr} tend to overfit and extrapolate poorly. The pre-trained meta model provides a class-level prior over plausible input–output trajectories, which we exploit to generate synthetic data conditioned on D_{tr} that enrich the training set and act as a regularizer for G .

This strategy rests on one key assumption: S_o can be reasonably well described by $\mathcal{M}_\phi$, i.e., D_{tr} , used as context, allows the meta model to meaningfully predict the trajectories of S_o .

In this setting, the meta model does not replace classical system identification but complements it. By generating synthetic trajectories conditioned on the same training data D_{tr} , it creates additional scenarios tailored to the specific system of interest, rather than generic WH realizations. The resulting two-step pipeline, “generate context-aware synthetic data $\rightarrow$ estimate a parametric model”, therefore trains G on a richer dataset composed of D_{tr} together with synthetic trajectories adapted to the target system, instead of training G directly on D_{tr} combined with unconditioned WH simulations.

Let us consider an observed input/output data sequence $(u_{1:m}, y_{1:m})$ (referred so far as context sequence) generated by system of interest S_o . Since this dataset represents the available information about S_o , it can be used either as a context for the meta model or as a training dataset for estimating a model $\mathbb{G}$ of the system S_o , or both. For this reason, $(u_{1:m}, y_{1:m})$ will be interchangeably referred to as the “context dataset” or “training dataset”, and will be denoted as D_{tr} .

We consider a parametric model $\mathbb{G}$ characterized by a parameter vector θ , where $\hat{y}_t(\theta)$ represents the output of $\mathbb{G}$ for a given input sequence up to time t , i.e.,

$$\hat{y}_t(\theta) = \mathbb{G}(u_{1:t}; \theta). \quad (16)$$

For simplicity of exposition, we assume a single-output system.

According to classical parametric system identification, the model parameters θ are typically obtained by minimizing an average loss function ℓ over the training dataset D_{tr} , i.e.,

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{m} \sum_{t=1}^m \ell(y_t, \hat{y}_t(\theta)). \quad (17)$$

To enrich the training set D_{tr} , we aim to generate an additional, potentially infinite, set of *synthetic* input/output sequences $\tilde{u}_{1:\tilde{T}}^i, \tilde{y}_{1:\tilde{T}}^i$ where $i = 1, 2, \dots$ is used to represent the i -th synthetic trajectory and $\tilde{T}$ is the length of the synthetic trajectories, that for simplicity of notation, is assumed to be the same for all synthetic datasets. The synthetic trajectories are drawn from a common probability distribution $p(\tilde{u}_{1:\tilde{T}}, \tilde{y}_{1:\tilde{T}})$, ideally matching the one underlying D_{tr} .

The main idea behind the data augmentation strategy proposed in the paper is to use the pre-trained meta model $\mathcal{M}_\phi$ to generate the synthetic input/output trajectories. Indeed, the meta model has the ability to infer the behavior of the specific query system $\mathcal{S}_o$ directly from the training (or, equivalently, context) dataset D_{tr} , which implicitly encodes key characteristics of the system $\mathcal{S}_o$. This trained meta model can then predict the output $\tilde{y}_{1:\tilde{T}}$ for any synthetic input sequence $\tilde{u}_{1:\tilde{T}}$, thereby facilitating generation of a potentially infinite dimensional set of synthetic data, by implicitly transferring knowledge across systems within the same class.

The following sections discuss two approaches for combining the training dataset D_{tr} with synthetic data generated by the pre-trained meta model $\mathcal{M}_\phi$. These approaches are related, as both incorporate the synthetic dataset as an additional regularization term in the original loss function in (17). However, they differ in how this regularization term is constructed. In the first approach, discussed in Section 5.1 and originally introduced in [30], equal weights in the regularization term are assigned to all synthetic samples. In contrast, the second approach, presented in Section 5.2, derives the regularization term based on a maximum-likelihood formulation and considers the uncertainty of the synthetic samples in constructing the training loss.

5.1 Deterministic regularization

The parameters θ that define the model $\mathbb{G}(\cdot, \theta)$ of the system $\mathcal{S}_o$ are estimated by minimizing an expected loss that incorporates both the real data D_{tr} generated by the system $\mathcal{S}_o$ and synthetic data generated by the pre-trained meta model $\mathcal{M}_\phi$, i.e.,

$$\begin{aligned} \hat{\theta} = \arg \min_{\theta} & \frac{1}{m} \sum_{t=1}^m \ell(y_t, \hat{y}_t(\theta)) + \\ & + \lambda \mathbb{E}_{p(\tilde{u}_{1:\tilde{T}}, \tilde{y}_{1:\tilde{T}})} \left[\frac{1}{\tilde{T}} \sum_{k=1}^{\tilde{T}} \ell(\tilde{y}_k, \hat{y}_k(\theta)) \right], \end{aligned} \quad (18)$$

where $\hat{y}_k(\theta)$ is the model's estimation output for a synthetic input sequence $\tilde{u}_{1:k}$, i.e., $\hat{y}_k(\theta) = \mathbb{G}(\tilde{u}_{1:k}, \theta)$.

The regularization hyper-parameter $\lambda \geq 0$ controls the relative importance of real versus synthetic data. Setting $\lambda = 0$ results in training on real data only, which may cause overfitting, particularly with complex models or limited datasets. A higher λ value emphasizes synthetic data, which can introduce distributional biases from the synthetic data generation process. Thus, λ should be chosen based on the quantity of real training data and on the quality of the synthetic data. In this work, λ is chosen via hold-out validation. Early stopping over a validation set is used to prevent overfitting, which may occur in case of small-size D_{tr} and when synthetic data are not employed. To not use additional data beyond what is already used for model estimation with early stopping, the same validation set can also be used for tuning λ .

The expected value in (18) is approximated with a sample average, by applying q synthetic input sequences $\tilde{u}_{1:\tilde{T}}^i$ (for $i = 1, \dots, q$) as input in the decoder, generating the corresponding outputs $\tilde{y}_{1:\tilde{T}}^i$ ². Consequently, the estimation problem becomes:

$$\begin{aligned} \hat{\theta} = \arg \min_{\theta} & \frac{1}{m} \sum_{t=1}^m \ell(y_t, \hat{y}_t(\theta)) + \\ & + \lambda \frac{1}{q} \frac{1}{\tilde{T}} \sum_{i=1}^q \sum_{k=1}^{\tilde{T}} \ell(\tilde{y}_k^i, \hat{y}_k^i(\theta)). \end{aligned} \quad (19)$$

²As initial conditions in the generation of the output trajectory $\tilde{y}_{1:\tilde{T}}^i$ we can consider the last c samples of the context D_{tr} .

The optimization problem (19) can be solved using mini-batch stochastic gradient descent, with q new synthetic input/output sequences generated at each iteration.

5.2 Maximum-likelihood based regularization

In this section, we present an alternative approach based on maximum likelihood for integrating real training data D_{tr} and synthetic data into a unified loss function for estimating the parameters θ of the model $\mathbb{G}$.

We assume that there exists an unknown “true” parameter θ^o such that, for any given input sequence $u_{1:t}$, the true system output y_t^o at time t is given by:

$$y_t^o = \mathbb{G}(u_{1:t}, \theta^o). \quad (20)$$

The measured training output is then modeled as:

$$y_t = \underbrace{\mathbb{G}(u_{1:t}, \theta^o)}_{y_t^o} + e_t, \quad t = 1, \dots, m, \quad (21)$$

where e_t is *independent and identically distributed* (i.i.d.) Gaussian noise, i.e., $e_t \sim \mathcal{N}(0, \sigma_e^2)$.

Thus, the likelihood function of the model parameters θ given the training dataset D_{tr} can be factorized as:

$$p(y_{1:m}|u_{1:m}, \theta) = \prod_{t=1}^m \mathcal{N}(y_t; \mathbb{G}(u_{1:t}, \theta), \sigma_e^2). \quad (22)$$

The predicted mean provided by the meta model is used as synthetic output $\tilde{y}_k$, namely:

$$\tilde{y}_k(\tilde{u}_{1:k}, D_{\text{tr}}) = \mu_k(\tilde{u}_{1:k}, D_{\text{tr}}), \quad k = 1, \dots, \tilde{T}, \quad (23)$$

where we emphasize the dependence of μ_k (and consequently $\tilde{y}_k$) on the training dataset D_{tr} (or, equivalently, the context).

According to the meta modeling assumption in (8), the true output $\tilde{y}_k^o = \mathbb{G}(\tilde{u}_{1:k}, \theta^o)$ of the system of interest $\mathcal{S}_o$ for a query input $\tilde{u}_{1:k}$ follows a Gaussian distribution:

$$p(\tilde{y}_k^o|\tilde{u}_{1:k}, D_{\text{tr}}) = \mathcal{N}(\tilde{y}_k^o; \mu_k, \sigma_k^2), \quad (24)$$

and

$$p(\tilde{y}_{1:\tilde{T}}^o|\tilde{u}_{1:\tilde{T}}, D_{\text{tr}}) = \prod_{k=1}^{\tilde{T}} \mathcal{N}(\tilde{y}_k^o; \mu_k, \sigma_k^2). \quad (25)$$

In the above equations, the dependence of μ_k and σ_k^2 is omitted not to burden the notation.

From the choice of the synthetic output $\tilde{y}_k$ in (23), and due to the symmetry of the Gaussian distribution, the likelihood function of the model parameters θ given the single synthetic output sequence $\tilde{y}_{1:k}$ and the training dataset D_{tr} is:

$$p(\tilde{y}_{1:\tilde{T}}|\tilde{u}_{1:\tilde{T}}, D_{\text{tr}}, \theta) = \prod_{k=1}^{\tilde{T}} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(\mu_k - \mathbb{G}(\tilde{u}_{1:k}, \theta))^2}{2\sigma_k^2}\right). \quad (26)$$

Thus, the joint likelihood function of the model parameters θ given both the training and synthetic output is:

$$\begin{aligned}
& p(y_{1:m}, \tilde{y}_{1:\tilde{T}} | u_{1:m}, \tilde{u}_{1:\tilde{T}}, \theta) = \\
& p(y_{1:m} | u_{1:m}, \theta) p(\tilde{y}_{1:\tilde{T}} | \tilde{u}_{1:\tilde{T}}, \underbrace{u_{1:m}, y_{1:m}}_{D_{\text{tr}}}, \theta) = \\
& \prod_{t=1}^m \mathcal{N}(y_t; \mathbb{G}(u_{1:t}, \theta), \sigma_e^2) \prod_{k=1}^{\tilde{T}} \mathcal{N}(\tilde{y}_k; \mathbb{G}(\tilde{u}_{1:k}, \theta), \sigma_k^2) = \\
& \prod_{t=1}^m \frac{1}{\sqrt{2\pi\sigma_e^2}} \exp\left(-\frac{(y_t - \mathbb{G}(u_{1:t}, \theta))^2}{2\sigma_e^2}\right) \times \\
& \prod_{k=1}^{\tilde{T}} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(\mu_k - \mathbb{G}(\tilde{u}_{1:k}, \theta))^2}{2\sigma_k^2}\right). \tag{27}
\end{aligned}$$

Thus, the log-likelihood function is given by:

$$\begin{aligned}
& \log p(y_{1:m}, \tilde{y}_{1:\tilde{T}} | u_{1:m}, \tilde{u}_{1:\tilde{T}}, \theta) = \\
& -\sum_{t=1}^m \frac{(y_t - \mathbb{G}(u_{1:t}, \theta))^2}{2\sigma_e^2} - \sum_{k=1}^{\tilde{T}} \frac{(\mu_k - \mathbb{G}(\tilde{u}_{1:k}, \theta))^2}{2\sigma_k^2} + C, \tag{28}
\end{aligned}$$

where C is a constant term that does not depend on the model parameters θ .

Maximizing the log-likelihood with respect to θ is equivalent to solving the following optimization problem:

$$\min_{\theta} \sum_{t=1}^m \frac{(y_t - \mathbb{G}(u_{1:t}, \theta))^2}{2\sigma_e^2} + \sum_{k=1}^{\tilde{T}} \frac{(\mu_k - \mathbb{G}(\tilde{u}_{1:k}, \theta))^2}{2\sigma_k^2}. \tag{29}$$

Since the noise variance σ_e^2 is constant (and might also be unknown), problem (29) is equivalently rewritten by introducing a hyper-parameter λ that weighs the second summation, i.e.,

$$\min_{\theta} \sum_{t=1}^m (y_t - \mathbb{G}(u_{1:t}, \theta))^2 + \lambda \sum_{k=1}^{\tilde{T}} \frac{(\mu_k - \mathbb{G}(\tilde{u}_{1:k}, \theta))^2}{\sigma_k^2}, \tag{30}$$

with λ tuned through cross-validation to balance the contribution of the synthetic data term, as also discussed in Section 5.1.

The extension to the case of multiple synthetic sequences is straightforward, leading to:

$$\min_{\theta} \sum_{t=1}^m (y_t - \mathbb{G}(u_{1:t}, \theta))^2 + \lambda \sum_{i=1}^q \sum_{k=1}^{\tilde{T}} \frac{(\mu_k^i - \mathbb{G}(\tilde{u}_{1:k}^i, \theta))^2}{(\sigma_k^i)^2}. \tag{31}$$

Summarizing, the optimization problem (31) aims at minimizing the sum of squared errors, where the first term accounts for the discrepancy between the model predictions and the observed real data, while the second term incorporates synthetic data. The synthetic data contributions are weighted inversely to their associated uncertainty, as reflected by the variance terms $(\sigma_k^i)^2$. Consequently, synthetic samples with high uncertainty have a lower influence on the loss, ensuring that the estimation process prioritizes more reliable data points.

The effectiveness of synthetic data is therefore not assumed a priori, but controlled at the point of use through this uncertainty weighting. The probabilistic meta model provides a per-sample uncertainty σ_k , and the maximum-likelihood loss weights synthetic contributions inversely to σ_k^2 , so that synthetic samples deemed uncertain have limited impact on the estimation of G , while high-confidence synthetic predictions carry more weight. In addition, before

relying heavily on synthetic trajectories, the practitioner can perform simple consistency checks on their distribution (e.g., input spectrum, amplitude range, operating region) relative to the available real data, and restrict the use of synthetic data to regimes where the meta model is both confident and the synthetic inputs remain within the domain explored by the real system.

6 Examples

In this section, we demonstrate the capabilities of the meta modelling paradigm through several simulation examples, as well as system identification benchmarks using real data.

Trainings and inferences are performed using an Nvidia RTX 4090 GPU with 24GB of RAM. To ensure the reproducibility of all experiments, the complete codebase, including dataset generation, training routines, and trained meta models, has been made publicly accessible in a GitHub repository [32].

6.1 Meta model training

6.1.1 System class

The meta model $\mathcal{M}_\phi$ is trained on synthetic datasets generated by simulating Wiener-Hammerstein (WH) systems characterized by the structure $G_1 - F - G_2$, where F represents a static nonlinearity positioned between two LTI blocks, G_1 and G_2 . The discrete-time LTI blocks G_1 and G_2 are randomly generated, with their orders uniformly sampled in the range $[1, 10]$, ensuring variability in their dynamic behavior. The pole locations of G_1 and G_2 are parameterized by magnitudes and phases, which are independently drawn from uniform distributions over the intervals $(0.5, 0.97)$ and $(0, \pi/2)$, respectively. The nonlinear function F is implemented as a feedforward neural network with a single hidden layer that contains 32 hidden units. The network weights are drawn from a Gaussian distribution with Kaiming scaling [19].

This choice of training class is principled rather than arbitrary. Any fading-memory nonlinear system admits an arbitrarily close approximation by a finite-order Volterra series [6], and parallel Wiener-Hammerstein structures can exactly represent finite-order Volterra kernels [4]. As a result, the WH-based synthetic corpus spans a rich family of nonlinear dynamics and provides a domain-informed pre-training distribution, rather than a system-specific simulator.

The inputs $u_{1:m}$, used as context signals, are discrete multisine signals of length m , with unitary amplitude and random phase, resulting in a flat spectral density. The active frequency components are constrained within a passband defined by the spectral index range $[f_{\min}, f_{\max}]$, with $f_{\min} \geq 1$ and $f_{\max} \leq \lceil m/2 \rceil$, with $\lceil \cdot \rceil$ denoting the ceiling operator, and $\lceil m/2 \rceil$ corresponding to the Nyquist frequency for a discrete signal. The upper bound $f_{\max}$ is randomly selected from a uniform distribution over the range $[1, \lceil m/2 \rceil]$. The lower bound $f_{\min}$ is then assigned based on the following scheme:

- with probability 0.5, we set $f_{\min} = 1$, enforcing a low-pass filtering effect;
- with probability 0.5, we sample $f_{\min}$ from a uniform distribution over $[1, f_{\max}]$.

The signals are synthesized in the frequency domain and transformed into the time domain using an inverse Discrete Fourier Transform. Finally, they are normalized to maintain zero mean and unit standard deviation. Query input signals $\tilde{u}_{1:n}$ are constructed similarly.

To simulate measurement errors coming from sensors, white Gaussian noise with a standard deviation of $\sigma_{\text{noise}} = 0.01$ is added to the outputs.

In the experiments presented here, the WH-based synthetic corpus is generated by a parametric model. In practice, the same meta-learning framework can be instantiated using any available simulator or physics-based model of a target application domain (e.g., robotics, powertrain, process control). The essential requirement is access to a simulator that captures a class

of plausible systems, not necessarily a perfect representation of the specific plant of interest. As we will discuss in Section 6.2.3, discrepancies between the simulator and the real system can be compensated by lightweight adaptation using a relatively small amount of data.

Remark 1. In contrast to large-scale foundation models trained on heterogeneous data with billions of parameters, the proposed meta model has a modest parameter count and is trained on synthetic trajectories drawn from a physically motivated class of systems. The WH-based design leverages existing knowledge on universal approximation for fading-memory nonlinear systems, rather than relying solely on brute-force scale.

6.1.2 Meta model architecture

The Transformer architecture representing the meta model $\mathcal{M}_\phi$ comprises $n_{\text{layers}} = 12$ layers, each with $d_{\text{model}} = 128$ hidden units. Attention mechanisms are implemented with $n_{\text{heads}} = 4$ in both the encoder and decoder. The query input/output sequences consists $n = 130$ samples, with the initial $c = 30$ samples used as initial conditions.

To assess performance across different context sequence complexities, two meta models are trained: one for a context length of $m = 400$ and another for $m = 16000$. This setup enables exploration of how the length of the context sequence affects model accuracy and computational efficiency. For the extended context length of $m = 16000$, the Transformer architecture incorporates the recurrent patching approach described in Section 4.1. The context is divided into $M = 400$ patches, each of length $L = m/M = 40$. These patches are sequentially processed by a vanilla RNN with a single hidden layer with $d_{\text{model}} = 128$ units. The final hidden state of each patch is further transformed through a square linear layer of size d_{model} , producing a sequence $p_{1:M}$. This sequence is then used as input to the Transformer’s encoder backbone. Conversely, for the shorter context length of $m = 400$, the recurrent patching method is not employed. Instead, the encoder input sequence $p_{1:M}$, where $M = m$, is directly obtained by processing the input/output samples, $u_{1:m}$ and $y_{1:m}$, through a linear layer that maps $n_u + n_y$ inputs to d_{model} outputs. In both scenarios, the Transformer architecture is described by approximately 5.5 million parameters, with additional 16896 parameters for the RNN layer used in the patching strategy.

The meta model $\mathcal{M}_\phi$ is trained to minimize the negative log-likelihood loss J_ϕ defined in (7) across 3 million epochs, utilizing the AdamW optimizer [23] with minibatches of size $b = 32$. The training times for the meta models with context lengths of $m = 400$ and $m = 16000$ are 28 and 50 hours, respectively.

6.2 Meta model testing

The predictive performance of the pre-trained meta model $\mathcal{M}_\phi$ is assessed not only across realizations of WH systems belonging to the same class used for meta model training, but also on simulated systems from a different class excited with non multisine input signals, as well as on *real* datasets from the nonlinear system identification benchmarks [1].

For the meta model handling a long context length ($m = 16000$), the patching-based approach is adopted, while ensembling is used when the meta model handling a shorter context ($m = 400$) is considered. This allows us to leverage the full context data, in case the available training dataset, at inference, is larger than 400 samples.

6.2.1 In-distribution test data

First, the performance of the pre-trained meta model is evaluated on test samples drawn from the same distribution as the training data. To this aim, 320 random realizations of simulated WH systems are generated for testing.

Table 1: In-distribution systems: Root Mean Square Error (RMSE) and inference time for patching-based and ensembling approach.

Method	RMSE mean (std)	Inference time [s]
Patching	0.048 (0.044)	0.019
Ensembling	0.038 (0.050)	0.039

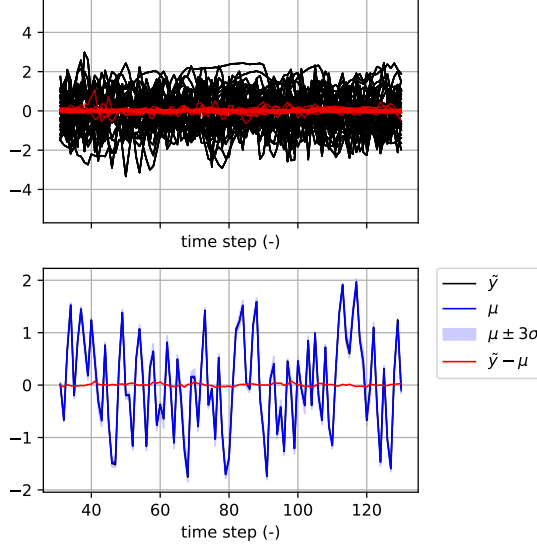


Figure 3: In-distribution test results obtained using the patching-based meta model. *Top*: aggregated prediction results over 320 WH systems, highlighting the consistency and stability of the predictions across many system realizations. *Bottom*: representative WH system showing the true output y , the predicted mean μ , the prediction error $y - \mu$, and the uncertainty band $\mu \pm 3\sigma$. The narrow confidence intervals and low error illustrate accurate prediction.

Table 1 reports the mean and standard deviation of the Root Mean Squared Error (RMSE) achieved over the 320 realizations, along with average inference times to predict the output on a query window of length $n - c = 100$. The patching and ensembling approaches achieve similar predictive performance, with the patching approach being approximately 2 times faster in making inferences.

Figure 3 illustrates the performance of the trained meta model. For visualization purposes, only results from the patching approach are displayed. The top panel shows the aggregated simulation results over the 320 randomly sampled Wiener-Hammerstein systems from the same distribution used for meta model training, highlighting the consistency of the model predictions across systems. The bottom panel provides a detailed view of a single WH realization, showing the true output $\tilde{y}_{c+1:n}$, the predicted mean output $\mu_{c+1:n}$, the corresponding prediction error $\tilde{y}_{c+1:n} - \mu_{c+1:n}$, and the uncertainty intervals computed as $\mu_{c+1:n} \pm 3\sigma_{c+1:n}$. The tight uncertainty bounds and low error in the figure demonstrate that the meta model accurately captures the system dynamics and provides well-calibrated uncertainty estimates.

6.2.2 Benchmarks

The pre-trained meta model $\mathcal{M}_\phi$ is further evaluated on publicly available nonlinear system identification benchmarks, namely the WH [37], cascaded tanks with overflow [38], and coupled electric drives (CED) [41], with CED consisting of 2 input/output trajectories. The adopted benchmarks consist of *real* datasets and thus allow us to investigate the meta model’s ability

to generalize beyond its training domain. Notably, these datasets are used solely for testing, with no additional training performed on the benchmark-specific training data. The available training datasets serve only as context for the pre-trained meta model $\mathcal{M}_\phi$. In our analysis, we prioritize demonstrating the zero-shot capabilities of the meta modeling paradigm, rather than directly comparing predictive performance with existing system identification methodologies.

The benchmark systems are chosen to span a spectrum of structural proximity to the synthetic training distribution, in order to stress-test the framework under varying degrees of distribution shift:

- **Wiener–Hammerstein (WH) benchmark.** This is an *in-distribution* test: the synthetic pre-training data are WH systems, so performance here validates that the framework correctly exploits the pre-training prior on systems drawn from the same class but generated in an experimental setup.
- **Coupled Electric Drives (CED).** The CED system is structurally close to a WH model: it consists of a linear dynamical core followed by a static nonlinearity, which matches the WH cascade structure. Good zero-shot performance on CED therefore confirms that the learned prior transfers to near-distribution systems with similar compositional structure.
- **Cascaded tanks with overflow.** This system exhibits a piecewise-linear overflow nonlinearity embedded in a nonlinear state-space model that does not factorize as a WH cascade, making it a genuinely out-of-distribution test for the meta model. The cascaded tanks benchmark is also characterized by a particularly small real training dataset, a regime where classical identification methods are prone to overfitting and struggle to generalize. In this setting, the inductive bias acquired during WH meta-training acts as a strong regularizer, enabling the framework to achieve competitive, and in some cases state-of-the-art, performance despite the distributional mismatch.

These observations highlight a regime where the proposed meta-modeling framework is particularly effective: when real training data per system are limited, the class-level prior learned from WH-based synthetic data provides a strong regularization effect and improves generalization. Conversely, on benchmarks with abundant task-specific data (e.g., systems with very long and informative training records), carefully tuned classical identification methods can still achieve higher asymptotic accuracy, whereas the meta model primarily offers the advantage of zero-shot prediction with modest performance gaps.

According to the benchmark guidelines, the first 50, 5, and 4 samples of the test data can be used as initial conditions for assessing predictive performance on the WH, cascaded tank, and CED benchmarks, respectively. As outlined in the previous section, the meta model is trained to utilize $c = 30$ initial conditions for inference. For the WH benchmark where 50 initial conditions are provided, only the last 30 samples are used. Conversely, for the cascade tank benchmark, zero-padding is applied to construct the sequence of initial conditions fed to the meta model. For the CED benchmark, where the training and test sequences are known to be contiguous, the sequence of initial conditions is constructed by concatenating the final 26 samples of the training data with the first 4 samples of the test.

Since the meta model is able to simulate the output over a query window of size 100, in order to accommodate the varying test sequence lengths in the benchmarks, an iterative inference approach is adopted. For the first inference step, the initial conditions provided by the benchmark dataset are used. In subsequent iterations, the c final simulated output samples serve as initial conditions for the next test sub-sequence. This iterative approach, combined with the weighted ensembling strategy, significantly increases the computational time required for inference. To address this limitation, the following strategies are implemented:

- **parallelization:** the predictions for the same inference window across all 400-length context sub-sequences, used for the ensembling, are computed in parallel.

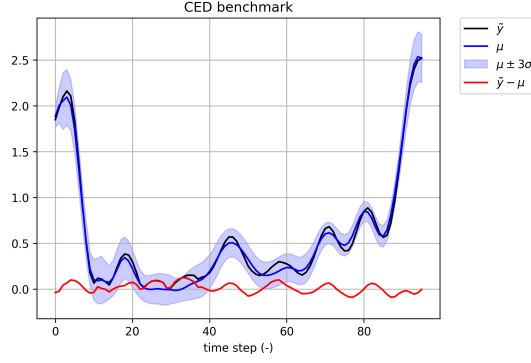


Figure 4: Prediction results on the CED benchmark for the second output signal. Comparison between true output y (black) and meta model’s predicted mean μ (blue), along with simulation error $y - \mu$ (red) and uncertainty region $\mu \pm 3\sigma$ (shaded blue area). Most predictions lie within the confidence bounds, demonstrating robust zero-shot generalization to real data.

- **pre-computation of encoder output:** during each inference iteration, only the decoder inputs change across different iterations. Consequently, the encoder output remains constant, allowing the encoder output to be computed only once for all the inference iterations.

Table 2: Benchmarks: RMSEs and inference time over the test datasets for multiple methods.

Benchmark	Method	RMSE	Training/Meta-inference time
WH	In-context	0.0014	5.91 s
	dynoNet [17]	0.0017	1-60 min
	LSTM [9]	0.0015	1-24 hr
	ARX [9]	0.0434	1-60 min
Cascaded Tanks	In-context	0.429	0.22 s
	dynoNet [17]	0.421	1-60 min
	LSTM [9]	0.452	1-24 hr
	ARX [9]	0.697	1-60 min
CED	In-context	[0.068, 0.055]	0.17 s
	dynoNet [17]	[0.062, 0.047]	1-60 min
	LSTM [9]	[0.139, 0.111]	1-24 hr
	ARX [9]	[0.433, 0.179]	1-60 min

Table 2 reports the RMSE values obtained on the test datasets for the three considered benchmarks, comparing the proposed approach with several baseline methods. It is important to note that the last column of the table compares the training time of the classical system identification baselines with the *meta-inference* time of the proposed approach, rather than its meta-training time. As discussed in Section 3, the proposed meta model is trained exclusively on synthetic data and does not use benchmark data during training. Consequently, the only stage in which benchmark data are employed is during inference. For this reason, the comparison between baseline training time and meta-inference time provides a fairer assessment of computational efficiency, while the meta-training time, already reported in the paper, represents an offline cost incurred only once. It is worth noting that, although the main advantage of the proposed meta modeling paradigm is its ability to perform prediction in a zero-shot learning fashion, the predictive performance is also in line with state-of-the-art system identification algorithms (see leaderboard in [27]).

Figure 4 shows the second output trajectory of the CED benchmark, where we can see the true output $\tilde{y}_{c+1:n}$ of the test dataset, the predicted mean output $\mu_{c+1:n}$ generated by the meta model; the associated prediction error $\tilde{y}_{c+1:n} - \mu_{c+1:n}$; and the 99.7% uncertainty interval computed as $\mu_{c+1:n} \pm 3\sigma_{c+1:n}$. The simulation error $y - \mu$ (red) remains small and within the uncertainty band for most time steps, indicating reliable generalization to real-world data despite the model being trained solely on simulated systems. This result highlights the zero-shot learning capability of the pre-trained meta model.

6.2.3 Model adaptation and fine-tuning

The performance of the pre-trained meta model is also evaluated in an out-of-distribution scenario characterized by shifts in both the input signals and the system class compared to those used for meta model training. To this end, we employed a simulator of a *continuous stirred tank reactor* (CSTR) with jacket cooling, where a first-order irreversible reaction, $A \rightarrow B$, occurs. The physical dynamics of the system are detailed in [24, Chapter 3]. In our analysis, the input variable is the flow rate of the cooling medium within the jackets, while the output is the concentration of reactant A in the reactor. The output is normalized to have 0 mean and unitary standard deviation. The reactor temperature and the temperature of the cooling medium are considered as system states.

The input to these systems is a *random binary signal* (RBS), used to control the flow of the cooling medium inside the jackets. The context sequence length is set to $m = 16000$, leveraging the recurrent patching-based approach, while the number of initial conditions is set to $c = 30$. To emulate realistic operating conditions, white Gaussian noise with a standard deviation of $\sigma_{\text{noise}} = 0.1$ is added to the normalized outputs.

The analysis is conducted both in *zero-shot* learning and in *few-shot* adaptation setting. In the zero-shot setting, the pre-trained meta model is directly applied to predict the CSTR output. In the *few-shot* setting, a lightweight fine-tuning phase is introduced following the methodology in [29], allowing the meta model to adjust to new conditions with minimal computational effort.

Two distinct fine-tuning scenarios are investigated:

- **Scenario 1: Input adaptation only.** The pre-trained meta model, originally trained on WH systems, is adapted by changing only the excitation signal from multisine to RBS, while keeping the same system class. Fine-tuning is performed for 1000 epochs.
- **Scenario 2: Input and system adaptation.** Adaptation data are generated by simulating CSTR systems with randomly perturbed physical parameters and using RBS inputs. To mimic a realistic few-shot setting, fine-tuning is carried out over a small set of 25 batches, each containing 32 systems, for a total of 1000 training epochs. Although the optimization runs for multiple epochs, the model repeatedly sees the same limited set of systems, making the adaptation effectively few-shot.

While the second scenario allows adaptation to both the excitation signal and the system class, the first corresponds to a more practical situation in which the user may not know the exact class of dynamical systems but does have control over the excitation signals.

To evaluate all the cases, 320 distinct CSTR realizations were generated by randomly perturbing selected physical parameters. The results, summarized in Table 3, show that the proposed lightweight fine-tuning procedure markedly improves the generalization capability of the pre-trained meta model. In the zero-shot setting, the WH-trained meta model is applied directly to the CSTR without any adaptation, resulting in a relatively large RMSE of 1.249. In Scenario 1, we keep the WH training class fixed and adapt only the excitation class from multisine to RBS, which yields a fivefold reduction in RMSE to 0.242 compared to the zero-shot setting. In Scenario 2, where both the input and the system class are adapted via CSTR simulations, the RMSE further decreases to 0.115, which is close to the noise floor $\sigma_{\text{noise}} = 0.1$. It is worth

Table 3: CSTR case study: RMSE without and with fine-tuning where only the input class changes (Scenario 1) and where both input and system class change (Scenario 2) during fine-tuning.

Setting	Zero-shot	Scenario 1	Scenario 2
RMSE	1.249	0.242	0.115

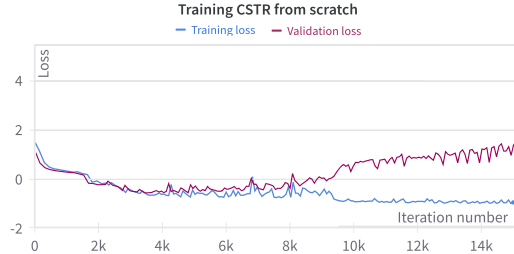


Figure 5: Training and validation curves for the meta model trained from scratch on the CSTR system class. The plot shows the evolution of the negative log-likelihood loss over training iterations for both the training (blue) and validation (red) datasets. A divergence between the training and validation losses is observed after approximately 4000 iterations, indicating overfitting.

noting, however, that the performance achieved in Scenario 1 is already remarkably close to this lower bound, despite adapting only the excitation input while keeping the meta-training system class unchanged. This indicates that a simple input-matching step can close a substantial portion of the sim-to-real gap, while the remaining gap can be bridged by a few-shot adaptation of the system class. Overall, the two scenarios highlight that, in certain practical situations, adapting the input representation alone is sufficient to recover most of the performance gains achievable through full adaptation, offering a practical and computationally efficient solution when the system dynamics are unknown but the input signal can be designed by the user.

For comparison, an additional analysis is conducted, with a meta model trained from scratch directly on the CSTR system class. To ensure fairness, the training process utilized the same number of system realizations used for fine-tuning previously (namely, 25 batches of 32 systems). As shown in Figure 5, this model overfits the small CSTR training set, and the best validation model achieves a test RMSE of 0.136, slightly worse than the fine-tuned meta model. Moreover, training from scratch requires about 4000 iterations to reach its best validation performance, compared to 1000 iterations for fine-tuning. This comparison highlights a practical advantage of the proposed approach: even when the simulator used for pre-training does not perfectly match the target system, a small amount of real or simulator data in the new domain can be used to adapt the pre-trained meta model more efficiently and effectively than training a new meta model from scratch.

6.3 Synthetic data generation for CED benchmark

In this section, we consider the CED benchmark to demonstrate the capabilities of the pre-trained meta model in generating synthetic data, as discussed in Section 5. This benchmark is known to be challenging because of the small size of the training dataset, which consists of only 400 samples.

Since the deterministic and the maximum-likelihood-based formulations achieved similar results, we only report the ones achieved with the maximum-likelihood approach.³

³Results related to the determinist formulation for synthetic data generation are reported on a simulation

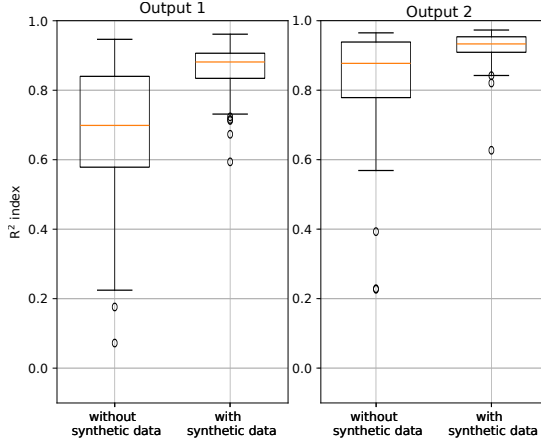


Figure 6: Performance comparison on the CED benchmark when training a dynoNet model with and without synthetic data generated by the pre-trained meta model. Boxplots report the distribution of the R^2 index over 100 Monte Carlo runs for both benchmark outputs. Using only real data (left) leads to higher variance and lower median R^2 , while augmenting the training set (right) with meta model-generated synthetic trajectories significantly improves accuracy.

As a parametric model $\mathbb{G}$ to describe the CED system, a dynoNet neural network [17] is employed. Its architecture comprises LTI layers, which process input sequences through filtering operations, combined with memoryless nonlinear components. Training is performed over 6000 iterations of AdamW. To mitigate overfitting, early stopping is utilized, where the first 300 samples of the training set are used to train the dynoNet, and the remaining 100 samples are used for validation. The same validation dataset is also employed to tune the regularization parameter λ in the loss function (31). The same 300 training samples serve as a context for the pre-trained meta model $\mathcal{M}_\phi$ to generate additional 200 synthetic samples, using a multisine signal as a query input. A Monte Carlo simulation with 100 runs is conducted, where at each run, the initial weights of the dynoNet are randomly initialized, and a new synthetic query input sequence is considered. We remark that the synthetic data was generated by the meta model $\mathcal{M}_\phi$, which was pre-trained on simulated WH models without any further fine-tuning or prior knowledge of the target system.

Results on the test dataset are reported in Figure 6, which shows a boxplot of the R^2 index on the two outputs of the test data over the 100 Monte Carlo runs, with and without using synthetic data. These results highlight the potential of synthetic data to enhance model robustness and performance, with a median R^2 index of 0.70 on the first output (0.88 on the second one) without synthetic data, and a median of 0.88 on the first output (0.93 on the second one) when the synthetic dataset is used. It is also noteworthy to observe the variability of the results across the Monte Carlo runs. Indeed, when no synthetic data is used, variability is solely due to the initial condition of the dynoNet weights. On the other hand, when using synthetic data, variability arises from the initial condition of the dynoNet weights, the selected hyper-parameter λ , and the generated synthetic data. Nevertheless, less variability is observed on the achieved R^2 when synthetic data is used.

7 Conclusions

In this paper, we presented novel developments on in-context learning for system identification, which enables zero-shot prediction of a system of interest without the need to estimate a model for that specific system. This approach eliminates the necessity to implement traditional system

example over the WH systems in the conference paper [30].

identification algorithms, thereby bypassing the need to select hyper-parameters typically chosen by system identification experts, such as model type, model order, training algorithm, etc. Indeed, the meta model is trained offline using a large set of synthetic data and has the capability to directly infer the behavior of the system of interest based on the available context (namely, available training data) and then predict the system’s output for any query input.

New developments based on sequence patching and ensembling also allow us to process long context sequences, circumventing the quadratic complexity of the attention mechanism of Transformers. Moreover, the meta model can generate synthetic data that can be used for data augmentation in classic system identification. This synthetic data allows to leverage information from systems in the class, as well as efficient processing of training data, implicitly learned by the encoder during the pre-training of the meta model.

Numerical results demonstrate how training a meta model with simulated data allows it to generalize well to real data or systems from a different class, potentially with quick adaptation.

Current research activities are focused on two distinct and seemingly opposite directions. On one hand, we aim at scaling to very large-scale meta models, characterized by billions of parameters. These models are expected to describe very broad classes of dynamic systems, pushing the boundaries of what dynamics meta models can comprehend and predict. On the other hand, research is also focusing on creating distilled architectures. These lighter models aim to enhance computational efficiency and accessibility, making advanced meta models techniques more feasible for real-time applications and edge devices with limited processing capabilities.

Limitations The estimated meta model $\mathcal{M}_\phi$ is able to predict, in a zero-shot learning fashion, the output of a given system of interest based on its context input/output sequence and for any possible query input. However, this predictive meta model also comes with limitations. First, meta training requires a substantial offline computational effort on a large synthetic corpus, which is not needed by classical system identification tools when they are applied to a single system with an ad hoc model structure [31]. Second, the meta model is relatively heavy at inference time, making it less suitable for embedded or resource-constrained implementations compared to parsimonious parametric models. Third, while the meta model achieves competitive accuracy in regimes where per-system data are scarce or weakly informative, classical identification methods trained directly on abundant, high-quality data for a specific system can still reach higher asymptotic accuracy. Finally, the meta model provides predictions rather than an immediately interpretable parametric representation, which can be a drawback when frequency-response analysis, sensitivity studies, or control design require a compact and structured model of the system of interest, although recent works on transformer-based model predictive control [10] indicate that such applications are possible but currently more specialized and computationally demanding.

Further developments in the field are expected to draw inspiration from advancements in Natural Language Generation algorithms, exploring the use of alternative architectures beyond Transformers, like Structured State Space and diffusion models. Additionally, there is potential to leverage open models like LLaMA or DeepSeek, applying low-rank adaptations to their weights to tailor them for the specific meta modeling challenges of dynamical systems.

References

- [1] Nonlinear benchmarks. <https://www.nonlinearbenchmark.org/>.
- [2] Andreas Antoniou, Harry Edwards, and Amos J. Storkey. How to train your MAML. In *Proceedings of the 36th International Conference on Machine Learning*, pages 300–309, 2019.

- [3] Gerben Beintema, Roland Toth, and Maarten Schoukens. Nonlinear state-space identification using deep encoder networks. In *Learning for dynamics and control*, pages 241–250. PMLR, 2021.
- [4] Stephen A Billings and SY Fakhouri. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, 18(1):15–26, 1982.
- [5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [6] Stephen Boyd and Leon Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on circuits and systems*, 32(11):1150–1161, 1985.
- [7] Ankush Chakrabarty, Gordon Wicern, Vedang M Deshpande, Abraham P Vinod, Karl Berntorp, and Christopher R Laughman. Meta-learning for physically-constrained neural system identification. *Neurocomputing*, 651:130945, 2025.
- [8] Ankush Chakrabarty, Gordon Wicern, and Christopher R. Laughman. Meta-learning of neural state-space models using data from similar systems. In *World Congress of the International Federation of Automatic Control (IFAC)*, July 2023.
- [9] Max Champneys, Gerben I Beintema, Maarten Schoukens, and Timothy J Rogers. Baseline results for selected nonlinear system identification benchmarks. *IFAC-PapersOnLine*, 58(15):474–479, 2024.
- [10] Xingxiao Chen and Mark Cannon. Successive convex optimization for transformer encoder model predictive control. *arXiv preprint arXiv:2605.14846*, 2026.
- [11] Zhehuai Chen, He Huang, Andrei Andrusenko, Oleksii Hrinchuk, Krishna C Puvvada, Jason Li, Subhankar Ghosh, Jagadeesh Balam, and Boris Ginsburg. Salm: Speech-augmented language model with in-context learning for speech recognition and translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13521–13525, 2024.
- [12] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, 2024.
- [13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *The Eighth International Conference on Learning Representations*, 2020.
- [14] Zhe Du, Haldun Balim, Samet Oymak, and Necmiye Ozay. Can transformers learn optimal filtering for unknown systems? *IEEE Control Systems Letters*, 7:3525–3530, 2023.
- [15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135, 2017.
- [16] Marco Forgione and Dario Piga. Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59:69–81, 2021.

- [17] Marco Forgione and Dario Piga. dynoNet: A neural network architecture for learning dynamical systems. *International Journal of Adaptive Control and Signal Processing*, 35(4):612–626, 2021.
- [18] Marco Forgione, Filippo Pura, and Dario Piga. From system models to class models: An in-context learning paradigm. *IEEE Control Systems Letters*, 7:3513–3518, 2023.
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [20] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.
- [21] R. Kaushik, T. Anne, and J.-B. Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5269–5276. IEEE, 2020.
- [22] Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.
- [23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [24] W.L. Luyben. *Chemical reactor design and control*. Wiley Interscience, 2007.
- [25] Daniele Masti and Alberto Bemporad. Learning nonlinear state-space models using autoencoders. *Automatica*, 129:109666, 2021.
- [26] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.
- [27] Jean Philippe Noël, Maarten Schoukens, and Koen Tiels. Nonlinear system identification benchmarks: Result reporting. <https://www.nonlinearbenchmark.org/results-reporting>, 2025. Visited on Februray 14, 2025.
- [28] Johan Paduart, Lieve Lauwers, Jan Swevers, Kris Smolders, Johan Schoukens, and Rik Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. *Automatica*, 46(4):647–656, 2010.
- [29] Dario Piga, Filippo Pura, and Marco Forgione. On the adaptation of in-context learners for system identification. In *Proceedings of the IFAC Symposium on System Identification*, 2024.
- [30] Dario Piga, Matteo Rufolo, Gabriele Maroni, Manas Mejari, and Marco Forgione. Synthetic data generation for system identification: leveraging knowledge transfer from similar systems. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 6383–6388. IEEE, 2024.
- [31] Gianluigi Pillonetto, Aleksandr Aravkin, Daniel Gedon, Lennart Ljung, Antônio H Ribeiro, and Thomas B Schön. Deep networks for system identification: a survey. *arXiv preprint arXiv:2301.12832*, 2023.
- [32] Filippo Pura, Matteo Rufolo, Marco Forgione, and Dario Piga. Sysid-meta-dist with patching. https://github.com/PuraFilippo/sysid_benchmarks, 2024.

- [33] Spencer M. Richards, Navid Azizan, Jean-Jacques E. Slotine, and Marco Pavone. Adaptive-control-oriented meta-learning for nonlinear systems. *Proceedings of the Robotics: Science and Systems (RSS)*, 2021.
- [34] Matteo Rufolo, Dario Piga, and Marco Forgione. Distributionally robust minimization in meta-learning for system identification. *IEEE Control Systems Letters*, 2025.
- [35] Matteo Rufolo, Dario Piga, Gabriele Maroni, and Marco Forgione. Enhanced transformer architecture for in-context learning of dynamical systems. In *Proceedings of the 2025 European Control Conference*, 2025.
- [36] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.
- [37] Johan Schoukens, Johan Suykens, and Lennart Ljung. Wiener-hammerstein benchmark. In *Proceedings of the 15th IFAC Symposium on System Identification*, 2009.
- [38] Maarten Schoukens, Per Mattsson, Torbjörn Wigren, and J Noël. Cascaded tanks benchmark combining soft and hard nonlinearities. In *Workshop on Nonlinear System Identification Benchmarks*, 04 2016.
- [39] Maarten Schoukens, Rik Pintelon, and Yves Rolain. Identification of wiener-hammerstein systems by a nonparametric separation of the best linear approximation. *Automatica*, 50(2):628–634, 2014.
- [40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [41] Torbjörn Wigren and Maarten Schoukens. Coupled electric drives data set and reference models. 2017.
- [42] Hongpeng Zhou, Chahine Ibrahim, Wei Xing Zheng, and Wei Pan. Sparse bayesian deep learning for dynamic system identification. *Automatica*, 144:110489, 2022.
- [43] Yucheng Zhou, Xiang Li, Qianning Wang, and Jianbing Shen. Visual in-context learning for large vision-language models. In *ACL (Findings)*, 2024.
- [44] Jiaqiang Ye Zhu, Carla Gomez Cano, David Vazquez Bermudez, and Michal Drozdal. InCoRo: In-Context Learning for Robotics Control with Feedback Loops. *arXiv preprint arXiv:2402.05188*, 2024.
- [45] Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. Multilingual machine translation with large language models: Empirical results and analysis. *arXiv preprint arXiv:2304.04675*, 2023.
- [46] Yifan Zuo et al. Accelerating materials discovery with machine learning: A review of applications and challenges. *Nature Reviews Materials*, 2021.

Appendix

A Minimum-variance weights for ensembling

In this appendix section we show that the variance $\sigma_{\mu_k}^2$, for a specific time step k , in (14) is minimized by the weights in (15), i.e.,

$$\omega_k^i = \frac{(\sigma_k^i)^{-2}}{\sum_{j=1}^s (\sigma_k^j)^{-2}}. \quad (32)$$

The minimal ensembling uncertainty $\sigma_{\mu_k}^2$ in (14) is given by the solution of the the optimization problem:

$$\min_{\omega_k^1, \dots, \omega_k^s} \sum_{i=1}^s (\omega_k^i \sigma_k^i)^2. \quad (33)$$

subject to the constraint $\sum_{i=1}^s \omega_k^i = 1$. Using the method of Lagrange multipliers, the augmented Lagrangian is:

$$\mathcal{L}(\omega_k, \gamma) = \sum_{i=1}^s (\omega_k^i \sigma_k^i)^2 + \gamma \left(\sum_{i=1}^s \omega_k^i - 1 \right). \quad (34)$$

Setting the partial derivatives of $\mathcal{L}(\omega_k, \gamma)$ to zero, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \omega_k^i} = 2\omega_k^i (\sigma_k^i)^2 + \gamma = 0 \implies \omega_k^i = \frac{-\gamma}{2(\sigma_k^i)^2}, \quad (35)$$

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^s \omega_k^i - 1 = 0 \implies \gamma = - \sum_{i=1}^s 2(\sigma_k^i)^2. \quad (36)$$

Substituting the expression for γ from (36) into (35), we obtain the optimal weights in (15):

$$\omega_k^i = \frac{(\sigma_k^i)^{-2}}{\sum_{j=1}^s (\sigma_k^j)^{-2}}, \quad i = 1, \dots, s. \quad (37)$$