MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Meta-Learning for Physically-Constrained Neural System Identification

Chakrabarty, Ankush; Wichern, Gordon; Deshpande, Vedang M.; Vinod, Abraham P.; Berntorp, Karl; Laughman, Christopher R.

TR2025-159 November 05, 2025

Abstract

We present a gradient-based meta-learning framework for rapid adaptation of neural state-space models (NSSMs) for black-box system identification. When applicable, we also incorporate domain-specific physical constraints to improve the accuracy of the NSSM. The major benefit of our approach is that instead of relying solely on data from a single query system, our framework utilizes data from a diverse set of source systems, enabling learning from limited contextualizing data from a query system, as well as with few online training iterations. Through benchmark examples, we demonstrate the potential of our approach, study the effect of fine-tuning subnetworks rather than full fine-tuning, and report real-world case studies to illustrate the practical application and generalizability of the approach to practical problems with physical- constraints. Specifically, we show that the meta-learned models result in improved downstream performance in model-based state estimation in indoor localization and energy systems. Keywords: System identification, machine learning, knowledge transfer, state-space models, bilevel optimization, estimation, emerging applications

Neurocomputing 2025

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Meta-Learning for Physically-Constrained Neural System Identification

Ankush Chakrabarty[a,*], Gordon Wichern[a], Vedang M. Deshpande[a], Abraham P. Vinod[a], Karl Berntorp[a], Christopher R. Laughman[a]

[a] *Mitsubishi Electric Research Laboratories, 201 Broadway, 8th Floor, Cambridge, MA 02139, USA*

Abstract

We present a gradient-based meta-learning framework for rapid adaptation of neural state-space models (NSSMs) for black-box system identification. When applicable, we also incorporate domain-specific physical constraints to improve the accuracy of the NSSM. The major benefit of our approach is that instead of relying solely on data from a single query system, our framework utilizes data from a diverse set of source systems, enabling learning from limited contextualizing data from a query system, as well as with few online training iterations. Through benchmark examples, we demonstrate the potential of our approach, study the effect of fine-tuning subnetworks rather than full fine-tuning, and report real-world case studies to illustrate the practical application and generalizability of the approach to practical problems with physical-constraints. Specifically, we show that the meta-learned models result in improved downstream performance in model-based state estimation in indoor localization and energy systems.

Keywords: System identification, machine learning, knowledge transfer, state-space models, bilevel optimization, estimation, emerging applications

1. Introduction

In order to design control systems in a model-based manner, one often has to resort to a system identification step. In particular, data-driven system identification has proven to be a useful modeling technique for systems for which structural knowledge is relatively unknown. One class of such models incorporates a classical state-space description as a core component, which has been simple to integrate with state-space methods for control and estimation, such as model predictive control or Kalman filtering. To widen the class of systems that can be modeled with state-space descriptions, recent advances have been made in the direction of neural state-space models (SSMs), where the state-space description is made in some latent space that is learned by neural networks. The neural networks may comprise shallow recurrent layers with a linear time-invariant state-space model in the latent space [1] or multi-layer perceptrons followed

*Corresponding author. Email: achakrabarty@ieee.org.

by linear-parameter-varying structure [2]. More recently, SSMs have been incorporated within the neural architecture explicitly without involving post-hoc adaptation operations [3, 4] for wider classes of nonlinear systems; see [5] for a recent survey. Unmodeled dynamics can be represented using neural SSMs after constructing a physics-informed prior model [6, 7, 8], regularization [9], and additional control-oriented structure can be embedded during training [10]. Another interesting direction of research has led to the development of autoencoder-based SSMs, where the neural architecture comprises an encoder that transforms the ambient state-space to a (usually high-dimensional) latent space, a decoder that inverse-transforms a latent state to the corresponding ambient state, and a linear SSM in the latent space that satisfactorily approximates the system's underlying dynamics [11, 12, 13]. An argument for the effectiveness of autoencoder-based approaches is based on Koopman operator theory [14], which posits that a nonlinear system (under some mild assumptions) can be lifted to an infinite-dimensional latent space where the state-transition is linear; an autoencoder allows a finite-dimensional, therefore tractable, approximation of the Koopman transformations [15].

Almost without exception, NSSMs have been constructed using data from a single *query* system under consideration. In practice though, modeling experts often have access to data for a range of similar (not necessarily identical) *source* systems. These source systems usually contain information that could speed up NSSM training for a query system, as long as the query system is similar to some of the source systems. Herein, we propose the use of gradient-based meta-learning to leverage data from similar systems to rapidly adapt a NSSM to a new query system with (i) few data points from the query system, and (ii) with very few online training iterations. To the best of our knowledge, we are the first to propose gradient-based meta-learning for physically-constrained system identification of black-box dynamics.

In our study, we employed MAML due to its versatility in adapting to new tasks with minimal data. Alternative meta-learning paradigms, such as metric-based and black-box approaches, offer distinct advantages and limitations. Metric-based methods, like prototypical networks [16] and matching networks [17] learn similarity metrics for classification tasks, excelling in few-shot learning scenarios but potentially struggling with tasks exhibiting significant heterogeneity. Black-box approaches, including memory-augmented networks [18] and meta-networks [19] utilize models with internal memory mechanisms to capture task-specific information, offering flexibility but often requiring substantial data and computational resources, and may lack interpretability. Our choice of MAML aligns with the need for a model-agnostic framework capable of rapid adaptation across varied tasks, as also discussed in recent surveys on meta-learning techniques [20].

Based on [20], we emphasize that meta-learning and multi-task learning (MTL) pursue fundamentally different goals. In MTL, a single model is jointly optimized across a fixed set of tasks: some layers may be task-specific, but no further adaptation occurs at inference time. By contrast, meta-learning explicitly trains a learner to produce rapid, on-the-fly adaptations for entirely new tasks, typically via a bi-level optimization that embeds an adaptation mechanism in its meta-objective. Similarly, although transfer learning also reuses

pre-trained weights, its pre-training phase is agnostic to any specific adaptation strategy: it merely provides a starting point for subsequent fine-tuning. During inference, a transfer-learned model must undergo a (often extensive) series of gradient steps—governed by a predetermined iteration budget—on a moderately sized target dataset. Meta-learning, in contrast, optimizes for few-shot adaptation, minimizing the number of required updates and tailoring the initialization itself to facilitate rapid learning on unseen tasks.

Prior art that bears some resemblance to this idea include: using data from linear systems within a prescribed ball to reduce model estimation error [21] and in-context learning with attention-based transformer networks that can represent SSMs for classes of dynamical systems, rather than single instances of that system class [22, 23]. In [21], the authors handle the case when the underlying class of systems exhibit linear dynamics. In this work, we consider the broader class of parameterized nonlinear systems and provide theoretical conditions that enable a definition of data-based similarity. The key differences with the line of work using transformer-based architectures is more nuanced. First, as argued in [24], many dynamical systems can be identified with shallow or slightly deep networks, and rarely require massive very large sizes, such as foundation models and transformer networks that are often extremely deep and contain millions (sometimes, billions) of parameters. Such massive transformer networks are beneficial for generalization and large-scale learning, but will be difficult to deploy in control or estimation applications requiring online adaptation, fast Jacobian calculations for linearization, or implementation on-chip. Therefore, it is necessary to devise fast adaptation mechanisms for small-sized networks, which is the objective of this work. Second, gradient-based meta-learning does not require massive pre-training datasets, making it more amenable to applications where a large number of similar systems' data is not available. Conversely, we will show empirical evidence that our proposed methodology can train from small training sets and adapt with very few online iterations. Outside of system identification, meta-learning has been proposed for adaptive control [25], receding horizon control [26, 27, 28], and optimization [29, 30].

The primary *contribution* of this work is to rapidly adapt neural SSMs for an unknown nonlinear system, using data obtained from other systems that exhibit similar dynamics to the query system. In other words, we design a framework to train a neural SSM on a variety of similar source systems and adapt online, with a few data points and a few training iterations, to make accurate predictions for a query system. This meta-learned neural SSM is shown via benchmark and practical examples to result in higher predictive accuracy than a neural SSM trained solely using query system data, or even a neural SSM trained on the entire source plus query data.

In particular, we rely on model agnostic meta-learning (MAML) [31], which involves solving a bilevel optimization problem during the meta-training procedure. The outer loop of the bilevel optimization intends to extricate task-independent features that are useful across all the similar systems, while the inner loop is designed to rapidly adapt to an unknown model with limited data. To combat numerical issues [32], a recent variant of MAML, referred to as almost-no-inner-loop (ANIL), separates the network into a base-

learner and a meta-learner, and dispenses with (or significantly reduces) inner-loop updates, to improve meta-training performance [33]. Additional improvements have been made to the base MAML algorithm to reduce memory- and sample-complexity such as implicit MAML [34], Reptile [35], and a perturbation-based first-order MAML [36], but we focus on the MAML and ANIL algorithms as the proposed approach can easily be adapted to these variants. Additional contributions include: incorporating physical constraints on the unconstrained neural SSM for two real-world case-studies, benchmarking the proposed approach against existing neural system identification methods, and investigating the effectiveness of full MAML adaptation versus ANIL adaptation of a subset of neural SSM layers.

Organization. In Section 2, we introduce the NSSM and formally state the system identification problem when similar systems' data is available. In Section 3, we review meta-learning approaches and discuss their application to the system identification setup of Section 2. Incorporating physics constraints into the NSSM is described in Section 4. In Section 5, we use two numerical examples, one for benchmarking against previous neural system identification methods, and one for demonstrating the effectiveness of meta-learning. In Section 6, we present two real-world case studies to demonstrate the practical application of meta-learning for physically constrained neural system identification. The Supplementary material contains additional explanatory figures and schematic diagrams, along with pseudocode for implementation.

2. Neural System Identification

2.1. Motivation

We assume that the query dynamical system under consideration has the form

$$x_{t+1} = f(x_t, u_t|\theta^\star), \quad y_t = h(x_t, u_t|\theta^\star), \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathbb{R}^{n_u}$ is a (usually, persistently exciting) control input, $y_t \in \mathbb{R}^{n_y}$ is the output, and $\theta^\star \in \mathbb{R}^{n_\theta}$ is the parameter vector. We do not assume that the modeling functions f and h are known, only that they satisfy basic smoothness and observability conditions, so that the system is well-posed, and solutions to the difference equations exist and are unique. We assume that the unknown parameter vector $\theta^\star$ belongs to a compact set $\Theta \subset \mathbb{R}^{n_\theta}$, and that the set Θ is known at design time. This is not a restrictive assumption, as most physical systems have models with parameters that can be, for instance, bounded within a certain range.

We assume the availability of a dataset $\mathfrak{D}_{\text{qry}}$ from the query system (1), with limited dataset size. In the classical system identification setting, this dataset contains input-output $\{u_t, y_t\}_{k=0}^{T^\star}$ data, with small $T^\star \in \mathbb{N}$. Alternatively, one may have access to a specific state representation predefined in a computer simulation environment: for example, in a digital twin of the query system under consideration. In such cases, the state representation x_t may be available, and the dataset $\mathfrak{D}_{\text{qry}}$ may contain state-input $\{x_t, u_t, y_t\}_{k=0}^{T^\star}$

data; in this case, observability conditions must be met to ensure that state reconstruction from outputs is feasible. Although the modeling functions f and h may be accessible in this setting, advanced simulation software often comprise modules of such high model complexity that attempting model-based design with these modules is often impractical, and sometimes intractable: for instance, if the model components cannot be represented using closed-form expressions.

Our objective is to leverage the limited dataset $\mathfrak{D}_{\text{qry}}$ to learn a neural predictive model that predicts the evolution of the system (1) accurately using a history of measured variables, while enforcing domain-specific or 'physics' constraints on the system. The physics constraints are typically derived from first principles, and are used to ensure that the learned model is physically consistent with the query system. We do not claim to be able to handle general nonlinear constraints, and limit ourselves to specific classes of constraints that arise in systems that we present later in Section 4. These constraints can be explicitly modeled using specific layers and activations in a neural network architectures.

2.2. Neural State-Space Modeling

To fulfill our objective, we propose using a neural state-space model (NSSM) architecture that can be represented in a general form; see Figure 1. Inputs to the NSSM are passed through encoding layers $\mathcal{E}(\cdot)$ to transform the inputs into a latent state ψ_t, which is subsequently passed through a state-transition operator $\mathcal{A}_\psi(\cdot)$ to obtain the updated latent state ψ_{t+1}. This updated latent state is decoded by the layers $\mathcal{D}_x(\cdot)$ and $\mathcal{D}_y(\cdot)$ to obtain intermediate variables $\tilde{x}_{t+1}$ and $\tilde{y}_{t+1}$, from which the predicted output $\hat{y}_{t+1}$ and updated state $\hat{x}_{t+1}$ is computed by enforcing the physical constraints using the constraint operator $\mathcal{P}(\cdot)$. The following cases illustrate how the NSSM architecture changes to reflect the composition of $\mathfrak{D}_{\text{qry}}$.

Case I, $\mathfrak{D}_{qry} = \{u, y\}$

When $\mathfrak{D}_{\text{qry}}$ contains input-output data $\{u_t, y_t\}_{t=0}^{T^\star}$, the appropriate architecture is shown in Figure 1[B]. In this case, the encoding layers $\mathcal{E}(\cdot)$ transform a history of inputs $U_{t-H:t}$ and outputs $Y_{t-H:t}$ into a latent state ψ_t, where H is a user-defined window length of past data. The latent ψ_t is updated using the state-transition operator $\mathcal{A}_\psi(\cdot)$, and the updated latent state ψ_{t+1} is decoded by the layers $\mathcal{D}_y(\cdot)$ to obtain an intermediate variable $\tilde{y}_{t+1}$. The predicted output $\hat{y}_{t+1}$ is computed by enforcing the physical constraints using the constraint-enforcing operator $\mathcal{P}(\cdot)$. The main operations are summarized by the equations:

$$\psi_t = \mathcal{E}(U_{t-H:t}, Y_{t-H:t}), \quad \psi_{t+1} = \mathcal{A}_\psi(\psi_t), \tag{2a}$$

$$\tilde{y}_{t+1} = \mathcal{D}_y(\psi_{t+1}), \quad \hat{y}_{t+1} = \mathcal{P}(\tilde{y}_{t+1}). \tag{2b}$$

To clarify, the training of the NSSM involves learning neural weights for the operators $\mathcal{A}_\psi$, $\mathcal{D}_y$, $\mathcal{E}$, and $\mathcal{P}$ or a combination thereof. For an input history $Y_{t-H:t}$ and $U_{t-H:t}$, the latent encoding ψ_t is computed using (2a), after which, for a user-defined prediction horizon of H_p, we recursively compute $\psi_{t+1}, \cdots, \psi_{t+H_p}$

with the state-transition operator $\mathcal{A}_\psi$. Subsequently, we can compute $\hat{Y}_{t:t+H_p-1} = \{\hat{y}_t, \hat{y}_{t+1}, \cdots, \hat{y}_{t+H_p-1}\}$ using (2b). As the classical NSSM training is done offline, the output y_t is available, with which one can construct $Y_{t:t+H_p-1}$. Then, the multi-step predictions can be evaluated via a mean-squared-error (MSE) loss function

$$\mathsf{L}_{\mathrm{SSM}}^{uy} = \mathsf{MSE}_{t:t+H_p-1}\big(y_{t+1}, \hat{y}_{t+1}\big) \tag{3}$$

that can be minimized using batched data and stochastic gradient descent methods.

Case II, $\mathfrak{D}_{qry} = \{x, u, y\}$

When $\mathfrak{D}_{\mathrm{qry}}$ contains state-input-output data $\{x_t, u_t, y_t\}_{t=0}^{T^\star}$, the NSSM architecture is modified as shown in Figure 1[C]. In this case, the encoding layers $\mathcal{E}(\cdot)$ transform the state x_t and input u_t into a latent state ψ_t, which is passed through the state-transition operator $\mathcal{A}_\psi$ to obtain the updated latent state ψ_{t+1}, from which decoders $\mathcal{D}_x$ and $\mathcal{D}_y$ compute intermediate variables $\tilde{x}_{t+1}$ and $\tilde{y}_{t+1}$, respectively. The predicted state $\hat{x}_{t+1}$ and output $\hat{y}_{t+1}$ are computed by enforcing the physical constraints using the constraint-enforcing operator $\mathcal{P}$. The main operations are summarized by the equations:

$$\psi_t = \mathcal{E}(x_t, u_t), \tag{4a}$$
$$\psi_{t+1} = \mathcal{A}_\psi(\psi_t), \tag{4b}$$
$$\tilde{x}_{t+1} = \mathcal{D}_x(\psi_{t+1}), \tag{4c}$$
$$\tilde{y}_{t+1} = \mathcal{D}_y(\psi_{t+1}), \tag{4d}$$
$$\hat{x}_{t+1}, \hat{y}_{t+1} = \mathcal{P}(\tilde{x}_{t+1}, \tilde{y}_{t+1}). \tag{4e}$$

Training the NSSM involves computing weights of $\mathcal{E}$, $\mathcal{A}_\psi$, $\mathcal{D}_x$, and $\mathcal{D}_y$. Since the network (4) is reminiscent of neural Koopman operators [37], we use the same loss functions as proposed in that literature. Namely, we minimize a loss function

$$\mathsf{L}_{\mathrm{SSM}}^{uxy} = \mathsf{L}_{\mathrm{recon}} + \mathsf{L}_{\mathrm{pred},x} + \mathsf{L}_{\mathrm{pred},y}. \tag{5}$$

The components of this loss function are: $\mathsf{L}_{\mathrm{recon}} = \mathsf{MSE}_{t-H:t}\big(x_t, \mathcal{P}\circ\mathcal{D}_x\circ\mathcal{E}(x_t, u_t)\big)$, $\mathsf{L}_{\mathrm{pred},x} = \mathsf{MSE}_{t:t+H_p-1}\big(x_{t+1}, \hat{x}_{t+1}\big)$, $\mathsf{L}_{\mathrm{pred},y} = \mathsf{MSE}_{t:t+H_p-1}(y_{t+1}, \hat{y}_{t+1})$, where the reconstruction loss ensures consistency between the lifting encoder and the de-lifting decoder, and the prediction losses ensure good predictive performance. Note that $\mathsf{L}_{\mathrm{recon}}$ has an abuse of notation, as $\mathcal{P} \circ \mathcal{D}_x$ actually indicates only the state reconstruction component.

Remark 1. In the case of multi-step prediction, we recursively compute $\psi_{k+1:k+N_S}$ for N_S steps starting from ψ_k with the state-transition operator and inputs. The MSE loss will be replaced by a multi-step MSE loss.

2.3. Extended Kalman filtering

We demonstrate the practical application of our proposed approach through real world examples using a meta-learned predictive model for subsequent applications, specifically, model-based estimation. We adopt

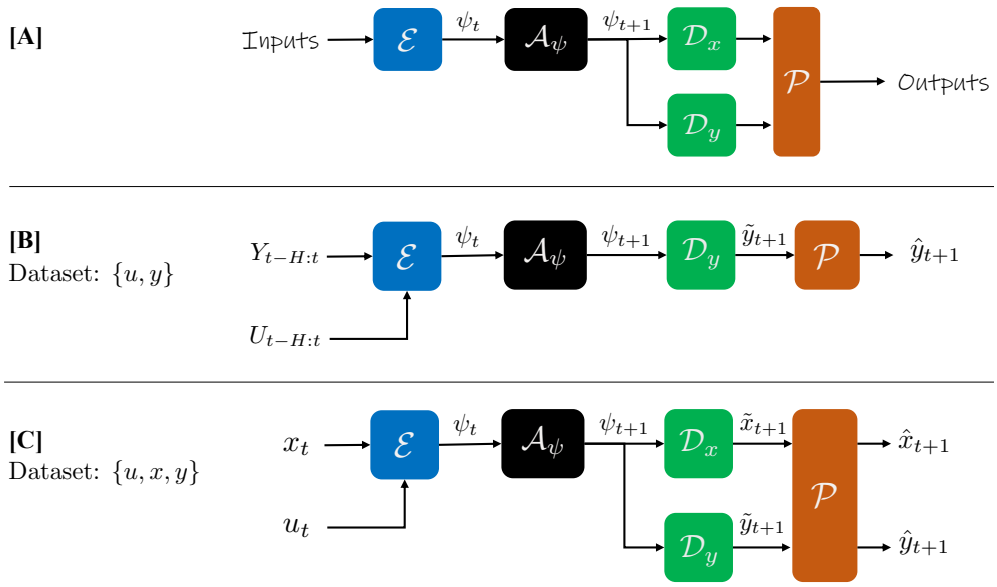

Figure 1: Neural state-space models: general abstraction, and specific models for various dataset compositions.

an extended Kalman filtering (EKF) framework for state estimation because of its applicability to a wider class of nonlinear systems [38].

In this section, we will briefly review an EKF for a general dynamic system expressed in the following form

$$x_{t+1} = \hat{f}(x_t, u_t) + w_t, \tag{6a}$$

$$y_t = \hat{h}(x_t, u_t) + \eta_t, \tag{6b}$$

which is easier to integrate with model-based filtering based on (1). Note that $\hat{f}(\cdot)$ and $\hat{h}(\cdot)$ here represent mathematical prediction models of the query system (1). The deviations of the predictive models (6) from the real system is captured by the random vectors w_t and η_t which are assumed to be zero-mean Gaussian uncertainties with covariances Q_t^w and Q_t^η, respectively. The EKF estimate of the state vector at time t is represented by a multivariate normal distribution, characterized by the mean-covariance pairs (x_t^-, P_t^-) and (x_t^+, P_t^+). These pairs denote the predicted prior after assimilating measurements up to time $t-1$ and the updated posterior after assimilating measurements up to time t, respectively.

Given a prediction model (6) with an initialization (x_0^-, P_0^-), the EKF implements the following time updates:

$$\boldsymbol{P}_{k+1}^- = F_t P_t^+ F_t^\top + Q_t^w, \tag{7a}$$

$$x_{k+1}^- = f(x_t^+, u_t), \tag{7b}$$

and the measurement updates:

$$K_t = (P_t^- H_t^\top)(H_t P_t^- H_t^\top + Q_t^\eta)^{-1}, \tag{7c}$$

$$P_t^+ = (I - K_t H_t) P_t^-, \tag{7d}$$

$$x_t^+ = x_t^- + K_t\big(y_t - h(x_t^-, u_t)\big), \tag{7e}$$

where F_t and H_t are Jacobian matrices defined as

$$F_t \triangleq \frac{\partial}{\partial x} f(x_t^+, u_t), \quad H_t \triangleq \frac{\partial}{\partial x} h(x_t^-, u_t). \tag{8}$$

In this work, we propose the use of meta-learned physics-constrained NSSMs described in Fig. 1 as the prediction models in (6) for implementing the EKF (7). Computing the Jacobian matrices in (8) involves differentiating the neural operators' outputs with respect to their inputs. We use auto-differentiation (AD) capabilities in standard deep learning libraries like `PyTorch` for this purpose.

2.4. Defining similarity of systems

Our work leverages the notion of *similarity* between systems as the motivation for using meta-learning for parameter identification.

Definition 1. Consider a class of systems of the form (1) with $f \in \mathscr{F}, h \in \mathscr{H}$. Then, a control input sequence $u_{0:T}$ driving two systems parameterized by θ and $\hat{\theta}$ and starting initial conditions x_0 and $\hat{x}_0$, respectively, results in outputs $y_{0:T}$ and $\hat{y}_{0:T}$ that satisfy,

$$\sup_{0 \le t \le T} \|y_t - \hat{y}_t\| \le E_{\mathscr{F},\mathscr{H}}(\|x_0 - \hat{x}_0\|, \|\theta_0 - \hat{\theta}_0\|), \tag{9}$$

for some function $E_{\mathscr{F},\mathscr{H}} : \mathbb{R}_{\ge 0} \times \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ such that $E_{\mathscr{F},\mathscr{H}}$ is increasing in both arguments and $E_{\mathscr{F},\mathscr{H}}(0,0) = 0$.

Definition 1 formalizes the notion that systems with close parameters and near-identical initial conditions produce similar behaviors, and the function $E_{\mathscr{F},\mathscr{H}}$ reveal how model sensitivity and contraction properties contribute to output trajectories deviating from one another. Definition 1 is motivated from a rich body of literature in system identification on distinguishability [39, 40, 41]. While a theoretical treatment of the notion of similiarity is beyond the scope of the paper, we show that such a definition is well-defined for a broad class of systems in the following theorem, and empirically holds for other systems as well in our numerical experiments.

Theorem 1. Let the following assumptions hold:
1) (Lipschitz f, h) There exist positive scalars $\mathfrak{L}_\theta^f, \mathfrak{L}_x^h, \mathfrak{L}_\theta^h > 0$ such that,

$$\|f(x,u,\theta) - f(x,u,\hat{\theta})\| \le \mathfrak{L}_\theta^f \|\theta - \hat{\theta}\|, \qquad \forall x \in \mathbb{R}^{n_x},\ \forall u \in \mathbb{R}^{n_u},\ \forall \theta, \hat{\theta} \in \Theta,$$

$$\|h(x,u,\theta) - h(\hat{x},u,\hat{\theta})\| \le \mathfrak{L}_x^h \|x - \hat{x}\| + \mathfrak{L}_\theta^h \|\theta - \hat{\theta}\|, \qquad \forall x, \hat{x} \in \mathbb{R}^{n_x},\ \forall u \in \mathbb{R}^{n_u},\ \forall \theta, \hat{\theta} \in \Theta.$$

2) (Contractive f) There exists a constant $\lambda^f \in (0,1)$ such that for all $x, \hat{x} \in \mathbb{R}^{n_x}$,

$$\|f(x,u,\theta) - f(\hat{x},u,\theta)\| \leq \lambda^f \|x - \hat{x}\|.$$

Let $\theta, \hat{\theta} \in \Theta$, and let $x_{0:T}$, $\hat{x}_{0:T}$ be the state trajectories and $y_{0:T}$, $\hat{y}_{0:T}$ be the output trajectories generated by the system (1) under parameters θ and $\hat{\theta}$, respectively, starting from initial conditions x_0 and $\hat{x}_0$, and with the same control inputs $u_{0:T}$. Then,

$$\sup_{0\leq t\leq T} \|y_t - \hat{y}_t\| \leq E_{\mathscr{F},\mathscr{H}}(\|x_0 - \hat{x}_0\|, \|\theta_0 - \hat{\theta}_0\|) \triangleq \mathfrak{L}_x^h \|x_0 - \hat{x}_0\| + \left(\frac{\mathfrak{L}_x^h \mathfrak{L}_\theta^f}{1-\lambda^f} + \mathfrak{L}_\theta^h\right)\|\theta - \hat{\theta}\|.$$

Proof. Let $\delta_\theta = \|\theta - \hat{\theta}\|$ and define the state error recursively as $\delta_t = \|x_t - \hat{x}_t\|$. Using triangle inequality and (1),

$$\|x_{t+1} - \hat{x}_{t+1}\| = \|f(x_t,u_t,\theta) - f(\hat{x}_t,u_t,\hat{\theta})\| \leq \|f(x_t,u_t,\theta) - f(x_t,u_t,\hat{\theta})\| + \|f(x_t,u_t,\hat{\theta}) - f(\hat{x}_t,u_t,\hat{\theta})\|.$$

By Lipschitz and contractivity assumptions on f, we get

$$\delta_{t+1} \leq \mathfrak{L}_\theta^f \delta_\theta + \lambda^f \delta_t.$$

This linear recursion with initial condition $\delta_0 = \|x_0 - \hat{x}_0\|$ yields the following upper bound since $\lambda^f < 1$,

$$\delta_t \leq \|x_0 - \hat{x}_0\| + \frac{\mathfrak{L}_\theta^f}{1-\lambda^f}\delta_\theta.$$

Finally, by Lipschitz assumption on h, we have for all $t \geq 0$

$$\|y_t - \hat{y}_t\| \leq \mathfrak{L}_x^h \delta_t + \mathfrak{L}_\theta^h \delta_\theta \leq \mathfrak{L}_x^h \|x_0 - \hat{x}_0\| + \left(\frac{\mathfrak{L}_x^h \mathfrak{L}_\theta^f}{1-\lambda^f} + \mathfrak{L}_\theta^h\right)\delta_\theta.$$

The proof is complete with the fact the above upper bound holds uniformly in t. □

2.5. Problem Statement

In practice, we do not always have access to a large amount of data from the query system. Therefore, training NSSMs with limited data results in overfitting and poor generalization, i.e., inaccurate query model predictions. It is not uncommon for data from systems similar to the query system to have been gathered in the past. For instance, previously deployed engineering systems similar to the target application may have been monitored, and data generated from these systems can supplement the limited query dataset $\mathfrak{D}_{\text{qry}}$ to improve generalization. We refer to this additional dataset as the source dataset $\mathfrak{D}_{\text{src}}$. The source dataset $\mathfrak{D}_{\text{src}}$ contains (u,y) or (x,u,y) data from a system that is similar to the query system, but not identical. The source dataset $\mathfrak{D}_{\text{src}}$ is assumed to be available at design time, and to have the same signal composition as the query dataset $\mathfrak{D}_{\text{qry}}$.

More formally, to reduce the data required from the query $\theta^\star$-system, we leverage simulation models to obtain data from $N_{\text{src}} \in \mathbb{N}$ dynamical systems that are similar to the query system. These *source systems* adhere to the form (1) and are parameterized by $\theta^\ell \in \Theta \setminus \theta^\star$ for $\ell = 1, 2, \ldots, N_{\text{src}}$. By collecting data from each of the N_{src} source systems, for instance during field experiments or via digital-twin simulations, we have access to a source dataset, which can be written as

$$\begin{aligned} \mathfrak{D}_{\text{src}} &\triangleq \left\{u_{0:T_\ell}, y_{0:T_\ell} | \theta^\ell\right\}_{\ell=0}^{N_{\text{src}}} \\ \text{or } \mathfrak{D}_{\text{src}} &\triangleq \left\{x_{0:T_\ell}, u_{0:T_\ell}, y_{0:T_\ell} | \theta^\ell\right\}_{\ell=0}^{N_{\text{src}}}, \end{aligned} \tag{10}$$

obtained over a time-span from zero to $T_\ell \in \mathbb{N}$ for some (not necessarily identical) initial condition and excitation inputs $U_{0:T_\ell}$. Often, the query system state data is collected with expensive instrumentation over a small time span (not necessarily real-time), so $T^\star \ll \inf_\ell T_\ell$.

By leveraging *meta-learning*, we can utilize the source data $\mathfrak{D}_{\text{src}}$ to learn NSSMs that can represent models with different $\theta \in \Theta$ parameters and rapidly adapt (i.e., with few online training updates) to the query system dynamics with unknown parameters $\theta^\star$ despite limited data. This adaptive identification approach also has the added practical benefit that no explicit estimation of the parameter $\theta^\star$ is required, which would likely require more data. Our objective is to design a rapid adaptation mechanism using meta-learning that can yield an accurate NSSM of the query system by learning from similar source systems. The adapted model can then be employed for downstream model-based control or model-based estimation tasks, as we will show in Section 6.

3. Model-Agnostic Meta-Learning (MAML) for Identification

MAML [31] is a highly recognized and widely utilized meta-learning algorithm, designed for rapid online adaptation. Its primary objective is to learn a set of network weights that can be rapidly fine-tuned during inference using a minimal amount of adaptation data. MAML employs a nested training approach: the inner loop fine-tunes a shared set of initial model parameters with a small amount of task-specific adaptation data, while the outer loop updates these initial parameters across a mini-batch of diverse tasks. This method encourages the learning of model parameters that can rapidly adapt to new tasks. Supplementary Fig. S1 provides a detailed overview of our proposed method for applying MAML to system identification.

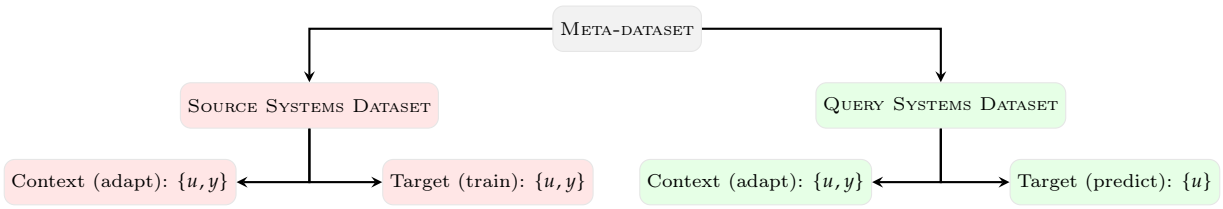

Figure 2: Partitioning of a meta-dataset into source and query datasets, each further divided into context and target sets.

One of the most challenging aspects of implementing meta-learning algorithms like MAML is data partitioning; see Figure 2 for clarification regarding nomenclature (support/query, context/target). Given a dataset of source systems $\mathfrak{D}_{\text{src}}$ as defined in (10), we can divide the data for the ℓ-th system into a *context* dataset $\mathcal{C}^\ell$ for inner-loop updates and a *target* dataset $\mathcal{T}^\ell$ to evaluate the loss function for the model parameters adapted in the inner loop. If ω denotes the set of trainable NSSM parameters[1], then the inner-loop MAML update is

$$\omega_m^\ell = \omega_{m-1}^\ell - \beta_{\text{in}} \nabla_{\omega_{m-1}^\ell} \mathsf{L}_{\text{SSM}}\left(\mathcal{C}^\ell; \omega_{m-1}^\ell\right) \tag{11}$$

where m denotes the iteration of inner-loop updates, β_{in} the inner-loop learning rate, and $\mathsf{L}_{\text{SSM}}\left(\mathcal{C}^\ell; \omega_{m-1}^\ell\right)$ is the loss function from (3) or (5), evaluated on the context set with the neural weights computed after $m-1$ inner-loop updates.

The inner-loop updates are performed separately for each trajectory in the batch using their respective context sets. In contrast, the outer-loop utilizes the target sets from all trajectories in the batch. The outer-loop optimization step is generally expressed as follows:

$$\omega = \omega - \beta_{\text{out}} \nabla_\omega \sum_{b=1}^{B} \mathsf{L}_{\text{SSM}}\left(\mathcal{T}^b; \omega_m^b\right) \tag{12}$$

where B is the number of tasks (i.e., trajectories) in a training mini-batch, β_{out} is the outer-loop learning rate, and $\mathsf{L}_{\text{SSM}}\left(\mathcal{T}^b; \omega_m^b\right)$ is the loss function on the target data $\mathcal{T}^b$ after m inner-loop iterations (11). Updating the model parameters in the outer-loop as described in (12) across B tasks yields a parameter set ω that can be quickly adapted at inference time. During inference, we perform only m inner-loop updates (11) with target system data $\mathcal{C}^\star = \mathfrak{D}_{\text{qry}}$, and evaluate NSSM performance with the updated parameters $\omega_m^\star$.

When dividing trajectory data into context sets for inner-loop fine-tuning and target sets for evaluating the fine-tuning, different strategies are employed for training and inference. During inference, the initial points of the observed trajectory are used as the context set, with subsequent points forming the target set. This approach simulates real-world scenarios where the model is first fine-tuned before being utilized. During training, consecutive points from any part of the trajectory are randomly selected as the context set, while other consecutive points (not necessarily following the context set) are used as the target set. This approach ensures the model does not always rely on initial conditions; for instance, when the target data is collected at steady-state conditions.

The nested training scheme in MAML can lead to a very challenging optimization problem [32]. Additionally, it may not be necessary to update all model parameters in the inner-loop [33]. To address this, the almost no inner-loop (ANIL) algorithm was introduced in [33]. The key idea behind ANIL is that the outer-loop update helps extract low-level features reusable across tasks, while the inner-loop facilitates rapid

[1]We must ensure that meta-trained parameters are genuinely trainable. For instance, physics constraint layer parameters $\mathcal{P}$ might need to remain fixed, as demonstrated in our case studies.

learning of a task-specific final layer. As such, ANIL updates only the parameters in the network's last layer during the inner-loop (11), while all layers' parameters are updated in the outer-loop (12).

In the context of few-shot image classification, a commonly studied problem in meta-learning literature, the last layer is known as the "classifier" and the earlier layers, which are not updated in the inner loop, are called the "feature extractor". However, for the NSSM architecture proposed in this work, it is unclear whether the encoder $\mathcal{E}$ or the state-space model parameters $\mathcal{A}_\psi$ should be shared across tasks or fine-tuned in the inner loop. In this paper, we study various options: which subnetwork should behave as the task-independent learner, and which should be the task-dependent adapter.

Formally, if $\omega = (\omega_1, ..., \omega_L)$, where ω_l denotes the network parameters for layer l and L is the total number of layers in the network, we then define $\omega_{\text{in}} \subseteq \omega$ as the subset of parameters updated in the inner-loop. Consequently, the inner-loop update from (11) is written as follows:

$$(\omega_l)_m^i = \begin{cases} \omega_l, & \omega_l \notin \omega_{\text{in}} \\ (\omega_l)_{m-1}^i - \beta_{\text{in}} \nabla \mathsf{L}_{\text{SSM}}\left(C_{Y^i}; (\omega_l)_{m-1}^i\right), & \omega_l \in \omega_{\text{in}} \end{cases} \tag{13}$$

Note that the outer-loop of ANIL is the same as in MAML, and both are equivalent when $\omega_{\text{in}} = \omega$. A summary of the meta-training procedure using source systems' data through ANIL and MAML is provided in Supplementary Algorithm S1, and Supplementary Algorithm S2 explains how to update the weights for the target system.

In Supplementary Section 3, we discuss some theoretical properties of the MAML algorithm, based on the results reported in [42].

4. Physically-Constrained NSSMs

Herein, we explicitly integrate physics constraints into the NSSM architecture to ensure that the predicted system trajectories are physically realizable. We consider two classes of constraints that may stem from the underlying physical laws governing the system's behavior. First, we propose an approach to constrain model predictions within a polytope. Next, we introduce an approach to guarantee that the model predictions analytically satisfy the curl-free property from vector calculus. As demonstrated in Section 6 through real-world applications, incorporating physics constraints into the NSSM architecture not only enhances model prediction accuracy but also improves downstream estimation performance.

4.1. Polytopic constraints

In certain applications, we know that the output of the NSSM $\hat{x}_t$ must satisfy constraints that can be encoded as $\hat{x}_t \in \mathcal{G}$, where $\mathcal{G}$ is a polytope

$$\mathcal{G} := \{x \in \mathbb{X} : Gx \leq 0\}, \tag{14}$$

for some known matrix G. The polytope $\mathcal{G}$ admits an equivalent rays-based description using the Minkowski-Weyl Theorem, [43][2], i.e., for some finite set of rays $\mathcal{R} := \begin{bmatrix}\mathcal{R}_1 & \cdots & \mathcal{R}_r\end{bmatrix} \in \mathbb{R}^{n_x \times r}$ we can write $\mathcal{G} = \mathsf{cone}(\mathcal{R})$, where $\mathsf{cone}(\mathcal{R}) := \sum_{j=1}^{r} \mu_j \mathcal{R}_j$ for some $\mu_j \geq 0$. In order to embed the physics constraint into the NSSM, we propose the following structure for the constraint layer $\mathcal{P}$. For the clarity of discussion, we decompose $\mathcal{P}$ into two parts $\mathcal{P}_x$ and $\mathcal{P}_y$ which act separately on the outputs of decoders $\mathcal{D}_x$ and $\mathcal{D}_y$ respectively. While we set $\mathcal{P}_y$ to be an identity matrix, we enforce the following structure on $\mathcal{P}_x$.

$$\mathcal{P}_x := \mathcal{R}\mu, \text{ where } \mu := \mathsf{ReLU} \circ \mathsf{FC}_r \circ \mathcal{D}_x(\psi), \tag{15}$$

where FC_r represents a fully connected layer with an output dimension of r, and $\mathsf{ReLU}(\cdot)$ denotes the rectified linear unit activation function. The decoder output $\mathcal{D}_x(\psi)$ is passed through a ReLU-activated fully connected layer, ensuring the vector is element-wise non-negative and meets the requirement of $\mu \geq 0$ in the conic constraint. This fully connected layer also maps $\mathcal{D}_x(\psi)$ to a vector μ that is dimensionally compatible with the ray matrix $\mathcal{R}$. Consequently, the product of $\mathcal{R}$ and μ results in a feasible predicted state $\bar{x}$. A general NSSM (4) corresponding to Fig. 1[C] with the constraint structure (15) was learned by minimizing the loss (5) using Supplementary Algorithm S1, and the weights were fine-tuned using Supplementary Algorithm S2 for the query systems.

Alternatively, consider the case where the constrained NSSM (4) is used for state estimation. For the clarity of discussion, we explicitly write the NSSM-based model equations similar to (6), as follows

$$x_{k+1} = \mathcal{P}_x \circ \mathcal{D}_x\big(\mathcal{A} \circ \mathcal{E}(x_t, u_t)\big) + w_t, \tag{16a}$$

$$y_t = \mathcal{D}_y \circ \mathcal{E}(x_t, u_t) + \eta_t, \tag{16b}$$

where w_t and η_t are assumed to be zero-mean Gaussian uncertainties with covariances Q_t^w and Q_t^η, respectively. The EKF (7a) can be readily implemented using (16) as the prediction models. The Jacobians F_t and H_t used in EKF are computed as follows

$$F_t \triangleq \frac{\partial}{\partial x}\mathcal{P}_x \circ \mathcal{D}_x \circ \mathcal{A} \circ \mathcal{E}(x_t^+, u_t), \tag{17a}$$

$$H_t \triangleq \frac{\partial}{\partial x}\mathcal{D}_y \circ \mathcal{E}(x_t^-, u_t). \tag{17b}$$

Although the NSSM (16) is designed to meet state constraints (14), the filter updates using (7a) do not explicitly satisfy these constraints, potentially leading to physics-inconsistent state estimates. Therefore, it is essential to incorporate state constraints during the filtering process [44, 45].

[2]According to Minkowski-Weyl Theorem, the feasible polytope of an inhomogeneous linear inequality $Gx \leq g$ can be equivalently described by a conic combination of rays and a convex combination of vertices. When $g = 0$, only the conic combination remains.

Various methods, such as density truncation [45], estimate projection [44], and pseudo or perfect measurements [46], can enforce constraints in state estimation algorithms [47]. We leverage the constraint-satisfying nature of NSSMs to enforce the same constraints on state vectors estimated by the EKF. Specifically, we propose using an auto-encoder pair $(\mathcal{E}, \mathcal{P}_x \circ \mathcal{D}_x)$ to project the filter estimate obtained in (7e) onto the feasible set. The constraint layer $\mathcal{P}_x$ ensures the decoded state vector meets the gradient constraint. Thus, a constrained state estimate is obtained by:

$$x_t^+ \leftarrow \mathcal{P}_x \circ \mathcal{D}_x \circ \mathcal{E}(x_t^+, u_t). \tag{18}$$

This constraint-projection operation is computationally efficient, involving only forward evaluation of neural networks, unlike other approaches that require solving a constrained quadratic program (QP) [44, 45]. We show in Section 6.1 that polytopic constraints are critical to incorporate physics while modeling vapor compression cycles via NSSMs to ensure unidirectional refrigerant flow.

4.2. Vector-field constraints

We now consider a scenario where the output is a vector field that must meet vector field constraints, specifically the curl-free constraint. This type of constraint can arise from physical conservation laws that govern the system, as seen in the localization application discussed in Section 6.2. In this case, the output models a magnetic field that is curl-free due to the absence of internal currents [48]. In such mechanical engineering applications, it is common for the state x to contain position information in terms of spatial coordinates. That is, some components of the state vector are known to be $[p_X, p_Y]$ or $[p_X, p_Y, p_Z]$, which denote spatial coordinates in a 2- or 3-dimensional Cartesian coordinate system.

Therefore, the curl-free constraint requires that $\vec{\nabla} \times h = 0$, where $\vec{\nabla}$ is the vector differential operator, $\vec{\nabla} = \begin{bmatrix} \frac{\partial}{\partial p_X} & \frac{\partial}{\partial p_Y} & \frac{\partial}{\partial p_Z} \end{bmatrix}^\top$ for 3-dimensional coordinates and $\vec{\nabla} = \begin{bmatrix} \frac{\partial}{\partial p_X} & \frac{\partial}{\partial p_Y} \end{bmatrix}^\top$ for 2-dimensions.

We again consider the base learner shown in Fig. 1[C]. The key operations in this physics-constrained NSSM are as described in (4). We select the physics-constraint layer $\mathcal{P}$ to be a vector gradient operator acting on the output channels, namely

$$\hat{x}_{t+1}, \hat{y}_{t+1} = \tilde{x}_{t+1}, \vec{\nabla}\, \tilde{y}_{t+1}.$$

In other words, if we decompose $\mathcal{P}$ into two parts $\mathcal{P}_x$ and $\mathcal{P}_y$ which act separately on the outputs of decoders $\mathcal{D}_x$ and $\mathcal{D}_y$ respectively, then $\mathcal{P}_x$ corresponds to an identity transformation and $\mathcal{P}_y$ involves computing a gradient (i.e., $\mathcal{P}_y = \vec{\nabla}$). Next, we discuss how the output path of the autoencoder implicitly enforces the curl-free constraint on the measurement function h.

The integration of physics-informed vector-field constraints into deep neural networks is broadly discussed in [49]. We adopt their approach to embed specific curl-free constraints within our NSSM, incorporating additional modifications to enhance training performance. Naturally, this constraint must be applied to

the neural network mapping from the state components x to the output y. For this purpose, we utilize the following result [50, pp. 444].

Theorem 2. Let h be a vector field of class C^1 whose domain is a simply connected region $\mathcal{R} \subset \mathbb{R}^3$. Then $h = \vec{\nabla}\varphi$ for some scalar-valued function φ of class C^2 on $\mathcal{R}$ if and only if $\vec{\nabla} \times h = 0$ at all points of $\mathcal{R}$.

This result implies the following: to model a curl-free vector field h using a neural network, we can learn a scalar function $\varphi(x)$, whose gradient $\vec{\nabla}\varphi$ will produce the desired curl-free vector field. Modern deep-learning toolkits, equipped with robust automatic differentiation (AD) modules, make it straightforward to compute y from φ, provided the state components x corresponding to the spatial coordinates $[p_X, p_Y, p_Z] \subset x$ are known. In this case, computing $\vec{\nabla}\varphi$ simply involves performing automatic differentiation with respect to these components. This insight has motivated the design of the output path ($\mathcal{P}_y = \vec{\nabla}$) in our proposed physics-constrained NSSM. Therefore, the estimate of the output function h that satisfies the curl-free constraint, is given by

$$\bar{h}(x, u) = \vec{\nabla}\,\mathcal{D}_y \circ \mathcal{E}(x, u). \tag{19}$$

To represent a non-constant h, the neural network's output must exist and be non-constant. This necessitates the use of smooth activation functions with non-constant derivatives. As seen in (8), we utilize the Jacobian of the NSSM for model-based state estimation, which requires both first-order and second-order derivatives to be smooth. This motivates the use of the `swish` activation function [51],

$$\varsigma(z) = \frac{z}{1 + \exp(-\beta z)}, \tag{20}$$

which is smooth, non-monotonic, and bounded only from below: see Supplementary Fig. S2. Typically, β is a trainable parameter, but for simplicity, we set $\beta = 1$. The `swish` function avoids output saturation since it is unbounded from above, thereby minimizing near-zero gradients for large values similar to the ReLU function. Also, the `swish` activation exhibits a soft self-gating property that does not impede the gradient flow due to its smoothness and non-monotonicity, thereby stabilizing the training procedure [51].

Next, we show some properties of the `swish` activation functions that are critical to our task. We begin with the following lemma.

Lemma 1. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be polynomials in z and e^z, respectively. Then

$$\frac{d}{dz}\left(\mathcal{P}_1 \otimes \mathcal{P}_2\right) = \sum_{i \in \mathcal{I}} c_i z^{q_i} e^{\tilde{q}_i z} =: \sum_{i \in \mathcal{I}} c_i \tilde{P}(z, e^z), \tag{21}$$

where $\otimes$ is the Kronecker product, $\mathcal{I}$ is some index set, and $c_i, q_i, \tilde{q}_i$ are scalar coefficients.

Proof. Since $\mathcal{P}_1(z) = \sum_i c_{1,i} z^i$ and $\mathcal{P}_2(e^z) = \sum_j c_{2,j} e^{jz}$, for some set of coefficient vectors c_1 and c_2, their tensor product can be written as $\sum_i \sum_j c_{1,i} c_{2,j} z^i e^{jz}$.

Applying the chain rule yields the derivative

$$\frac{d}{dz}\left(\mathcal{P}_1 \otimes \mathcal{P}_2\right) = \sum_i \sum_j c_{1,i} c_{2,j} e^{jz} \left(i z^{i-1} + j z^i\right),$$

whose terms can be rearranged to yield (21). □

Next, we show that the n-th derivative of the `swish` activation has an alternative representation that enables us to infer its smoothness properties.

Theorem 3. Let $\varsigma(z)$ be defined as in (20). For any $n \in \mathbb{N}$, the n-th derivative of σ can be written as

$$\varsigma^{(n)}(z) = \frac{\sum_i c_i^{(n)} \left(\mathcal{P}_1^{(n)}(z) \otimes \mathcal{P}_2^{(n)}(e^z)\right)}{(1+e^z)^{n+1}}, \tag{22}$$

where $\mathcal{P}_1^{(n)}$ and $\mathcal{P}_2^{(n)}$ are polynomials in z and $\exp(z)$, respectively and c_i are scalar coefficients.

Proof. We prove this by induction. For $n = 1$, we calculate

$$\varsigma^{(1)}(z) = \frac{(z+1)e^z + e^{2z}}{(1+e^z)^2}.$$

Therefore, $\mathcal{P}_1^{(1)} = [1, z]^\top$ and $\mathcal{P}_2^{(1)} = [e^z, e^{2z}]^\top$, with $c^{(1)} = [1, 1, 1, 0]$.

Now, we assume that the statement holds for n, and it remains to be shown to hold for $n+1$. To this end, we differentiate $\varsigma^{(n)}$, which yields

$$\begin{aligned}
\varsigma^{(n+1)}(z) &= -(n+1)e^z \frac{\sum_i c_i^{(n)} \left(\mathcal{P}_1^{(n)}(z) \otimes \mathcal{P}_2^{(n)}(e^z)\right)}{(1+e^z)^{n+2}} \\
&\quad + \frac{\sum_i c_i^{(n)} \frac{d}{dz}\left(\mathcal{P}_1^{(n)}(z) \otimes \mathcal{P}_2^{(n)}(e^z)\right)}{(1+e^z)^{n+1}}, \\
&= \frac{-(n+1)e^z \sum_i c_i^{(n)} \left(\mathcal{P}_1^{(n)} \otimes \mathcal{P}_2^{(n)}\right) + (1+e^z) \sum_{\mathcal{I}} \tilde{c}_i \tilde{\mathcal{P}}(z, e^z)}{(1+e^z)^{n+2}},
\end{aligned}$$

where $\tilde{c}_i \tilde{\mathcal{P}}(z, e^z)$ is obtained using Lemma 1. We observe that the numerator has introduced additional terms that contain higher-order terms of e^z and z along with their combinations. Therefore, by introducing new polynomials $\mathcal{P}_1^{(n+1)}(z) = [\mathcal{P}_1^{(n)}(z); z^{n+1}]$ and $\mathcal{P}_2^{(n+1)}(z) = [\mathcal{P}_2^{(n)}(e^z); e^{(n+1)z}]$ and choosing appropriate coefficients, one can rewrite the numerator in the previous expression as

$$\frac{\sum_i c_i^{(n+1)} \left(\mathcal{P}_1^{(n+1)}(z) \otimes \mathcal{P}_2^{(n+1)}(e^z)\right)}{(1+e^z)^{n+2}},$$

which affirms the $n+1$-case, and concludes the proof. □

From Theorem 3, we conclude that the n-th derivative of $\varsigma(\cdot)$ is composed of differentiable functions, which indicates that its derivatives exist, and are themselves smooth, ensuring the gradient operations to obtain y from φ are valid. Consequently, the output path of the NSSM is n-times differentiable, as it is composed of smooth functions passed through affine operators (defined by the weights and biases of the NSSM layers). This is useful not only for expressing more complex vector field constraints, but also imperative in the EKF context because one has to evaluate Jacobians of the NSSM as seen in (8), which involves twice-differentiating the output path.

Additionally, Theorem 3 implies that, as long as some $c_i^{(n)}$'s are non-zero, the derivatives of the swish function are non-constant. This is also particularly useful since constant higher-order derivatives would not be useful for expressing non-constant vector fields. We hypothesize that as long as the trained weight matrices of $\mathcal{E}$ and $\mathcal{D}_y$ are non-zero, and not pathological in terms of dropping rank, one can expect that the output path $\vec{\nabla}\mathcal{D}_y \circ \mathcal{E}$ of the proposed neural architecture is non-constant if the NSSM uses swish activation functions. In fact, we can explicitly check that for $n \leq 2$, the swish function and its derivatives are all smooth and non-constant; see Supplementary Fig. S2. Training the NSSM involves minimizing the differentiable training loss as in (5).

Remark 2. Alternatives to the `swish` activation function include the `tanh`, `SELU`, and `mish` functions, all of which are smooth, and have non-constant first- and second-derivatives. Unlike `tanh`, `swish` is unbounded from above, which prevents vanishing gradients and therefore improves gradient flow during backpropagation through deeper network architectures. Conversely, ReLU or LeakyReLU functions cannot be used as their 2nd-derivatives are (where they exist) zero.

Remark 3. While we have presented the curl constraint here due to its explicit use in the experimental section, the paper [49, §4] also provides a way to incorporate divergence-free constraints, i.e. $\vec{\nabla}\cdot h = 0$ by using a different gradient-based operator.

Remark 4. ReLU-based projection onto a convex polytope defines a piecewise analytic under analytic partition (PAP) function, which is almost-everywhere differentiable and whose autograd-computed "intensional" derivatives coincide with true gradients almost everywhere [52]. The PAP class is closed under composition, so inserting such projections within deeper architectures does not introduce new non-differentiable structures. Finally, optimization theory for PAP functions guarantees that standard (sub)gradient methods avoid pathological non-differentiable points almost surely, implying no intrinsic convergence issues due to the hard projection layer.

5. Benchmarking and Ablations

5.1. Example 1. The Bouc-Wen Hysteretic System Benchmark

5.1.1. Setup and Illustration of Performance

Our first example is the Bouc-Wen oscillator which exhibits hysteretic dynamics; a well-known system identification benchmark. Code and data are available on GitHub at: `github.com/merlresearch/MetaLIC`. The Bouc-Wen model is described by discretizing the continuous dynamics

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = \frac{1}{m_L}(u - k_L x_1 - c_L x_2 - x_3), \tag{23a}$$

$$\dot{x}_3 = \alpha x_2 - \beta\left(\gamma|x_2||x_3|^{\nu-1}x_3 + \delta x_2|x_3|^{\nu}\right), \tag{23b}$$

$$y = x_1 + w_y, \tag{23c}$$

where u is an input excitation acting on the oscillator, z is the hysteretic force, and y is the output displacement that can be measured in real-time [53]. The noise w_y is Gaussian white noise with a bandwidth of 375 Hz, and a standard deviation of 8×10^{-3} mm. The discretization sampling frequency is 750 Hz. The set of parameters, whose nominal values are provided in [53], are unknown for our purposes, but assumed to lie within a range $[\Theta_{\min}, \Theta_{\max}]$ given by Table 1.

Table 1: Target parameter $\theta^\star$ and ranges for Example 1.

$\theta :=$	m_L	c_L	k_L	α	β	γ	δ
$\Theta_{\min}$	1	5	2.5×10^4	2.5×10^4	500	0.5	-1.5
$\Theta_{\max}$	3	15	7.5×10^4	7.5×10^4	4500	0.9	-0.5
$\theta^\star :=$	2	10	5.0×10^4	5.0×10^4	1000	0.8	-1.1

For the query system, parameterized by $\theta^\star$, the benchmark dataset contains a context and target set comprising 40960 samples and 8192 samples, respectively. These samples are noisy, and have been obtained using a multi-sine sweep [54]. In order to test the performance of meta-learned system identification on this system, we extend this example to a family of oscillators by sampling $N_{\text{src}} = 40$ sets of parameters uniformly within the admissible range Θ. These source systems are used for meta-training, and the source dataset is collected by simulating each of these systems for $T_\ell = 10^4$ time steps, with a sampling period of 1/750s. The excitation to all source systems is $u(t) = 120\sin(2\pi t)$. We demonstrate in Supplementary Fig. S3 that the source systems exhibit a wide range of hysteretic loops, and therefore, the meta-learning problem is well-posed, as the dynamics are similar, but not identical, across source systems. Furthermore, the system is Lipschitz (locally) in the parameters, but not contractive; we use this to illustrate that even when Theorem 1 does not hold, empirical results may be satisfactory.

The NSSM architecture has the form (2) since we have access to (u, y) data. We set the backward window length of $H = 20$. The encoder is 7 layers deep, activated by exponential linear unit (ELU) functions, with $[256, 128, 128, 64, 64, 32, 32]$ neurons across layers. The latent dimension $n_\psi = 16$, and the state-transition operator $\mathcal{A}_\psi$ is fully connected 5 layers deep with 256 neurons each and ELU activation. The decoder structure is a mirror reflection of the encoder, and $\mathcal{P}$ is the identity operator, because no physical constraints are required to be enforced in the benchmark system. For the MAML algorithm, we set inner- and outer-loop learning rates to be 0.001, the adaptation budget to be $M = 40$ iterations, and run the algorithm for 10K epochs with a 80/20 validation split in the source dataset. With NVIDIA 3090X GPU acceleration, the full meta-training phase takes 16 hr.

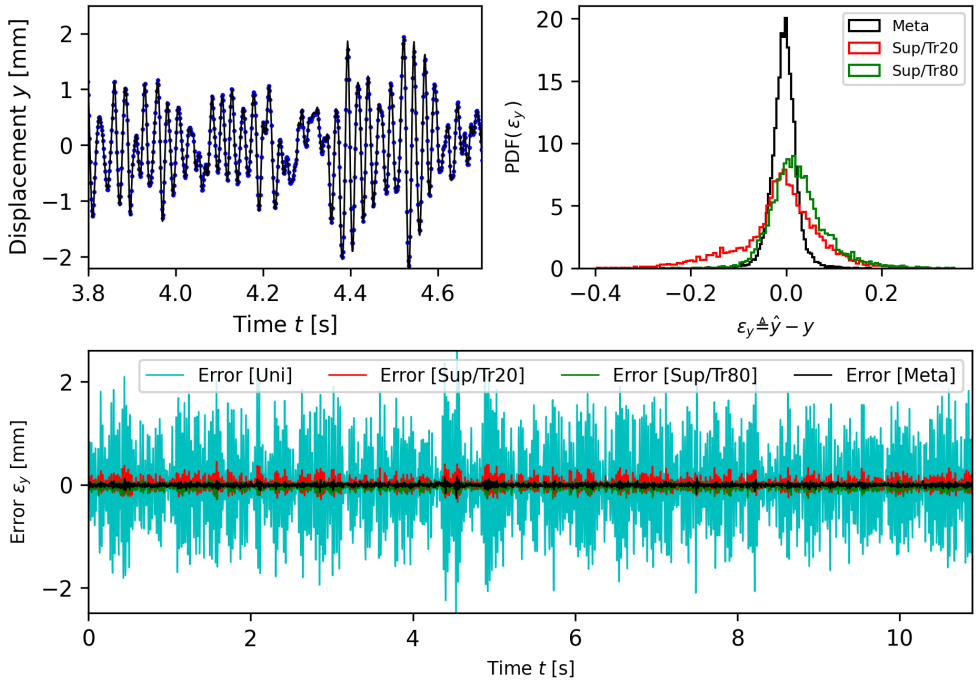

Figure 3: Performance of meta-learning for Example 1.

We present the performance of the meta-learned NSSM in Fig. 3. The top left subplot shows that the final MAML-NSSM predictions (blue circles) are overlapping with the ground truth data (black line) for a subset of time where the Bouc-Wen system transitions between two magnitudes of oscillation, between [-1,1] within 3.8–4.2 s, and between [-2,2] within 4.8–5.2 s. Both phases of oscillatory response are reproduced with small predictive error, as evident by the top-right subplot which shows a distribution of the prediction error for three algorithms including MAML. Clearly, meta-learned NSSM errors are tight around 0 with a 99% confidence interval of $[-0.1, 0.1]$. This is further supported by the lower subplot, which shows the error signal evolution over time (black line) is very close to zero most of the time.

5.1.2. Benchmarking Results

Table 2: Mean performance comparison of meta-learned NSSM with supervised learning.

	META	UNIV	SUP/TR20	SUP/TR80
RMSE [$\times 10^{-5}$ mm]	**2.83**	72.35	8.41	6.50
Fit [%]	**95.47**	29.12	86.76	90.67

As reported in Table 2, we also compare against: (i) a Universal NSSM, which is trained on all the data, both from the source and query systems, along with greedy supervised learned NSSM with just the query data seeing (ii) 80% of the query data as context, called TR80, or (iii) 20% of the target data as context, called TR20. The objective of comparing against TR80 is to demonstrate that MAML-NSSM can outperform even when almost the entire dataset is available for training, because the query dataset itself is of limited size. The TR20 case is to compare performance when the same amount is given for adaptation/learning, to demonstrate the benefit of using commonalities from similar systems versus learning from scratch. We evaluate these models using the goodness-of-fit and root-mean-squared-error[3]; mean metrics are presented over 20 independent runs. Clearly, MAML outperforms all 3 ablations, with the closest competitor being TR80, as expected, because it has access to the most amount of query system data. The universal model performs the worst, as it is unable to capture the dynamics of the target system well, due to the large variability in the source systems.

Furthermore, we compare the performance of various meta-learning algorithms for this benchmark problem, including FO-MAML [31], iMAML [34], and Reptile [35]. As this is a well-explored problem in the nonlinear system identification literature, we also compare against the dynoNet algorithm [55], the polynomial-basis algorithm proposed in [56] specifically for hysteretic systems, and the method in [57] that uses nonlinear finite impulse response (NFIR) models, nonlinear auto regressive with exogenous input models (NARX); and nonlinear orthornormal basis function (NOBF). We only present RMSE comparisons in Table 3 as those are explicitly reported in all the competitor papers.

From the table, we deduce that the expressivity of neural SSMs in general is higher than NFIR and NARX for this system, with the exception of the method of [56], which exhibits excellent performance. This is because the polynomial basis method is tailored specifically to the Bouc-Wen (and generally, hysteretic) systems and, more importantly, their results are reported for the noise-free case $w_y \equiv 0$. Overall, MAML performs well on this problem, even though our modeling paradigm is generalizable to a wide class of

[3]We define these formally as

$$\mathsf{Fit}(\%) = 100\left(1 - \frac{\|y - \hat{y}\|}{\|y - \bar{y}\|}\right), \ \mathsf{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2},$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the total number of samples.

Table 3: Mean RMSE comparison of meta-learning algorithms for the Bouc-Wen system.

Algorithm	RMSE [mm]	Algorithm	RMSE [mm]
MAML [31]	2.53×10^{-5}	FO-MAML [31]	2.74×10^{-5}
iMAML [34]	2.97×10^{-5}	Reptile [35]	2.66×10^{-5}
dynoNet [7]	4.52×10^{-5}	Polynomial [56]	1.87×10^{-5}
NFIR [57]	16.4×10^{-5}	NARX [57]	17.3×10^{-5}

systems. The variants of MAML (iMAML, first-order-MAML and Reptile) perform comparably, although we have found empirically that iMAML hyperparameters are harder to choose than the others, and iMAML performance is quite sensitive to this selection.

5.2. Example 2: Investigating Subnetwork Fine-Tuning with ANIL on Chaotic Oscillators

We use a family of unforced van der Pol oscillators subject to parameter uncertainty as a test problem for validating our meta-learned NSSM. Each oscillator is given by the continuous dynamics

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = \theta x_2(1 - x_1^2) - x_1, \quad y = x, \tag{24}$$

which are discretized by a forward-Euler method with a time-step of 0.01s.

5.2.1. Data collection and implementation details

The parameter θ is randomly sampled from a uniform distribution $\Theta = \mathcal{U}([0.5, 2])$ for generating the source and target systems. Because θ is effectively a damping parameter, these sampled systems induce a broad range of dynamics. The parameter value for the target system was randomly chosen to be $\theta^\star = 1.572$. Both the range of θ and the target $\theta^\star$ are unknown to the learner. We generated trajectories for $N_s = 200$ source systems, each with unique $\theta \sim \Theta$ values. For each of the source systems, the initial state and the final time were randomly sampled from uniform distributions $\mathcal{U}([-1, 1]^2)$ and $T \sim \mathcal{U}([10, 40])$ s, respectively. Initial state of the target system was fixed to be $[1, -0.5]^\top$. The past and predictive window lengths were set to $H = 10$ and $H_p = 5$, respectively.

The encoder of our NSSM consists of an input layer for x, followed by 5 hidden layers with 128 neurons each, and an output layer generating a latent variable of dimension $n_\psi = 128$. The deep neural network was implemented in `PyTorch` [58] using rectified linear units (ReLUs) as the activation function. Hyper-parameters for the training were set as follows: the batch size $B = 32$, the number of inner-loop iterations $m = 10$, the total number of meta-training epochs set to 10^4, and the step-sizes for MAML set to 0.01 and 0.001 for the inner and outer loops, respectively. The MAML was allowed $m = 40$ steps for online adaptation.

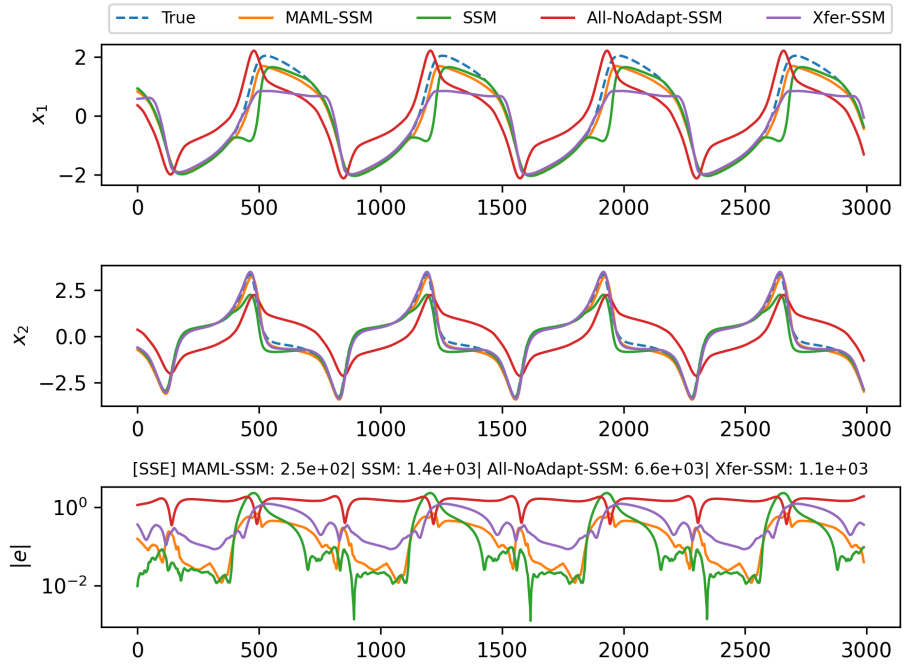

Figure 4: Comparative study of various meta-learning approaches for system identification with neural SSMs.

5.2.2. Meta-learning vs. supervised and transfer-learning

To evaluate the effectiveness of the proposed `MAML-SSM`, we compared it with several baselines. These baselines share the same architecture as `MAML-SSM` but are trained using a single training loop instead of the inner-outer bi-level loop of MAML. The baselines and the reasoning behind their selection are as follows:

- `SSM`: Trained using supervised learning, in a classical one-training-loop, with only the target system data. This baseline is chosen to see if `MAML-SSM` can learn from similar systems or if the data from other systems degrades the predictive quality.
- `All-NoAdapt-SSM`: Trained using supervised learning with both source and target system data, without any online adaptation steps. This baseline investigates if supervised learning is sufficient to create a good predictive model given access to all available data (source and target).
- `Xfer-SSM`: Trained using transfer learning, which involves supervised learning on all source system data, followed by a few online adaptation steps using the target system data. This baseline evaluates if meta-learning can surpass transfer learning in few-shot, data-limited scenarios.

While supervised learners were allowed 10 times more training steps than `MAML-SSM`, the number of adaptation steps were identical for `MAML-SSM` and `Xfer-SSM`.

Figure 4 showcases the performance of `MAML-SSM` and baseline models. During online inference, the

context set is derived from the first 400 time steps, while the NSSM predicts the target set for the next 3000 time steps. The top and middle subplots depict the system's state evolution, with dashed lines indicating the true query system states. The bottom subplot illustrates the sum-squared-error (SSE) evolution between the true and predicted states over the 3000 time steps. The proposed `MAML-SSM` clearly achieves the lowest prediction SSE, outperforming all baselines. The transfer learning model, `Xfer-SSM`, ranks second with twice the SSE of `MAML-SSM`, followed closely by `SSM`. In contrast, `All-NoAdapt-SSM`, which uses all data without adaptation, shows significantly poorer performance consistently over time.

The x_1 subplot suggests that `SSM` likely overfits the context data, resulting in excellent predictive accuracy within the training set but significant decline at out-of-set points. In contrast, `All-NoAdapt-SSM` appears to severely underfit the data, as it aims to find a set of NSSM weights that match the 'average' dynamics of both the source and query systems, rather than adapting specifically to the query system. As anticipated, both meta- and transfer-learning models perform well, with our proposed `MAML-SSM` outperforming `Xfer-SSM`.

Given the limited data from the target system and the few adaptation iterations, the trained learner (prior to adaptation) needs neural weights that can quickly adapt to the target system. This is precisely what MAML is designed for, enabling `MAML-SSM` to swiftly develop an effective predictive model for the given example. Conversely, the transfer learning approach (prior to adaptation) is not specifically trained with adaptation in mind. Thus, the initial weights are likely optimized for generating good predictions for the source systems, and the few-shot adaptation is insufficient for `Xfer-SSM` to achieve the same accuracy as `MAML-SSM`. This is particularly evident in the x_1 plot, where transfer learning fails to capture the positive phase of the oscillatory dynamics.

5.2.3. Subnetwork Fine-Tuning with ANIL

ANIL facilitates partial meta-learning of NSSMs by dividing the NSSM into two components: a base-learner (with parameters $\omega_l \notin \omega_{\text{in}}$) and a meta-learner (with parameters $\omega_l \in \omega_{\text{in}}$). In this setup, only the meta-learner undergoes online adaptation, while the base-learner remains fixed. While we understand that the contribution of the encoder $\mathcal{E}$ is to lift or project the inputs to a latent space which provides benefits in terms of computational tractability or expressivity, and the role of the state-space matrix is to encode the dynamics in the latent space, it is not immediately clear which should be the base-learner and which layers should be adapted. To answer this question, we use the target system with a fixed final time 10s, and allow various subnetworks of the model to behave as the base-learner, and other layers to adapt. The decoders are always allowed to adapt in order to maintain consistency with the adapted encoder, i.e. so they can retain reconstructive ability. The experiment results are shown in Fig. 5, where the columns of the table indicate what percentage of the testing data was used for adaptation, the colorbar represents the color-code for the RMSE (lighter colors indicate better accuracy of the NSSM), and the rows indicate which meta-learning algorithms were considered. Of course, MAML and first-order (FO)-MAML allow for full fine-tuning, i.e. all

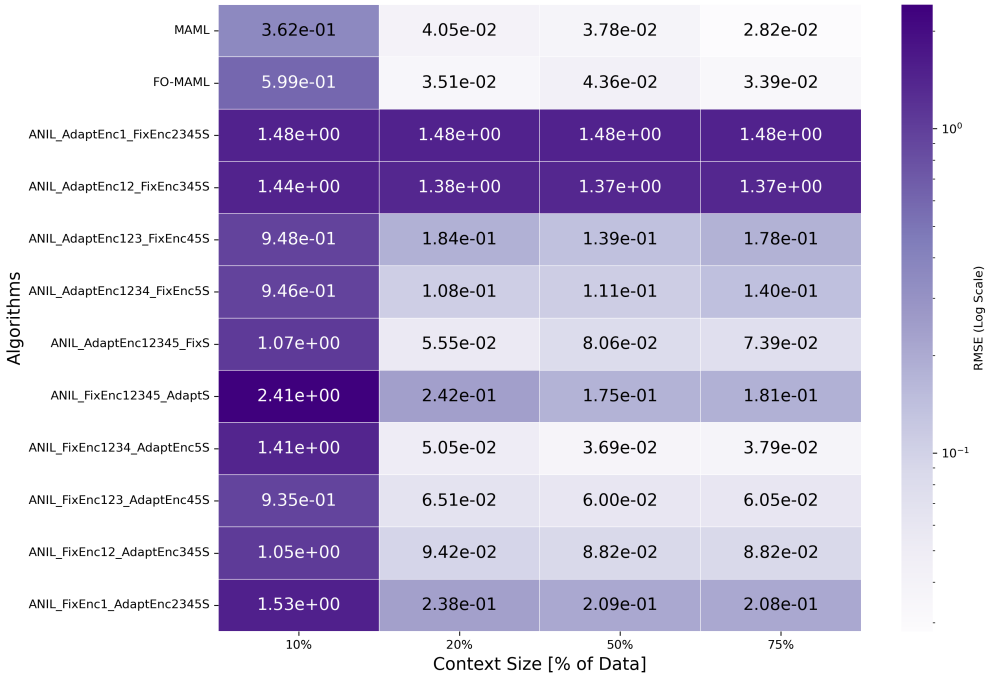

Figure 5: RMSE (averaged over 20 runs) heatmap produced by tuning subnetworks with ANIL.

layers are adapted. For the last 10 rows, we report the results of ANIL using the following nomenclature: `ANIL_FixEnc123_AdaptEnc45S` to indicate that layers `1,2,3` of the encoder are fixed (`1` being nearest to the input channels) and only the `4,5` layers of the encoder along with the state-transition layers `S` $= \mathcal{A}_\psi$ are adapted. Similarly, `ANIL_AdaptEnc12345_FixEncS` indicates that all encoder layers were adapted and the state-transition layers were fixed. Clearly, from Fig. 5, we can see that the best performance is for MAML and FO-MAML, which allows full fine-tuning. This is to be expected, as this allows for the most expressive adaptation. Interestingly, we see that the worst performance is obtained when the encoder is mostly fixed and only the state-transition layers are allowed to be adapted. This indicates that it is most critical to allow the encoder to learn and adapt the lifting/projection function that determines the latent space rather than just the state-transition matrix. Thinking about this in another way, it is not enough for just $\mathcal{A}_\psi$ to be adapted while fixing most of $\mathcal{E}$ if we are to learn across similar systems with some variety in the dynamics.

It is also important to note that even though meta-learning is effective for few-shot learning, there is a practical lower bound in terms of the amount of data that is enough for a given problem: here, 10% target data is too small for meta-inference and consistently gives poor results, whereas anything more than 10% yields decent performance, although the best performance overall is with 50% or more data for most high-performing algorithms. Of course, since we report the mean in the colortable, it is possible that there is slightly higher error when more data is provided, but the discrepancy is always small and most likely due to random seeds. However, one reason why adding more data might be resulting in worsened

performance without full fine-tuning is due to overfitting, e.g. in the cases of `ANIL_AdaptEnc123_Fix45S` and `ANIL_AdaptEnc1234_Fix5S`. The best performance is reported when the first few layers of the encoder is fixed and the final layers are adapted along with adaptation of $\mathcal{A}_\psi$. This can be defended by our knowledge of deep neural network training, wherein the early layers are responsible for extracting basic, general features from the input data, such as common features and and simpler transformations, whereas later layers focus on more complex, system-specific transformations [59].

6. Case Studies with Constrained Physics

We study two real-world systems, detailing the challenges encountered in these applications and how they benefit from the proposed approach, along with implementation details.

6.1. Vapor Compression with Unknown Refrigerant Mass

Fig. 6[A] illustrates a standard vapor compression system, an essential technology for numerous heat transfer applications in both residential and industrial settings. The system includes a compressor that drives the working fluid (e.g., a refrigerant) through the VCS, and two fans that push air through the heat exchangers (condenser and evaporator). The refrigerant absorbs heat from the air passing through the evaporator and releases it to the air passing through the condenser.

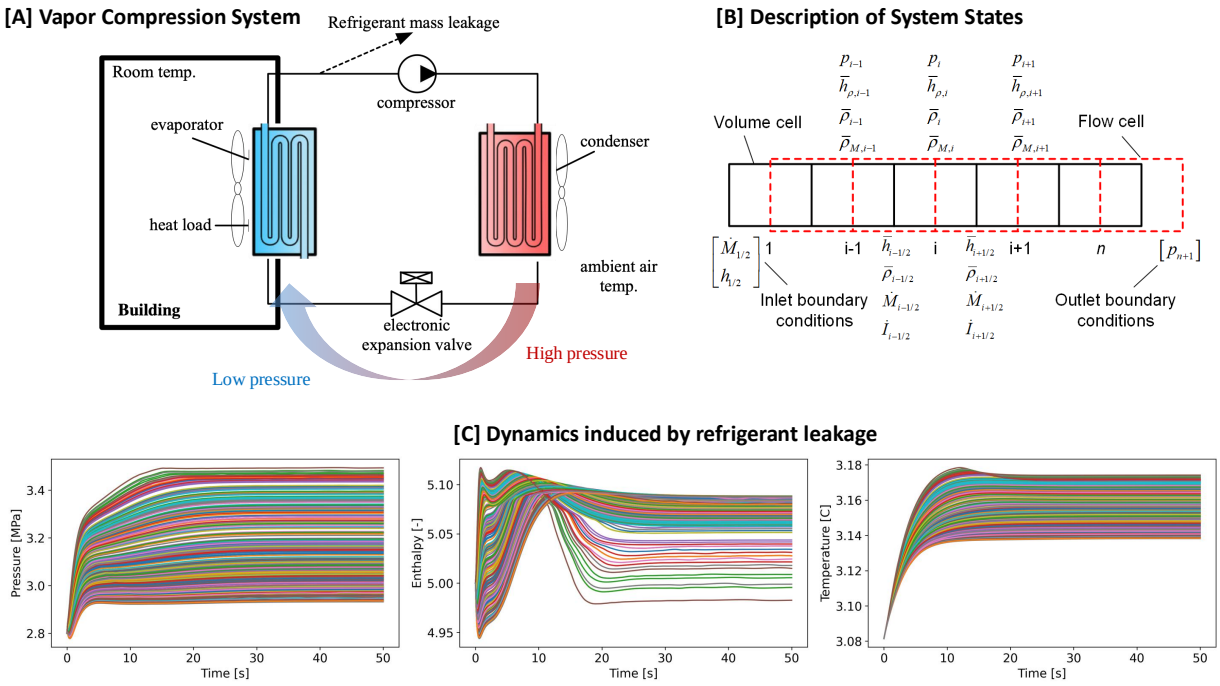

Figure 6: Range of HVAC dynamics induced by variable refrigerant mass.

Many technological solutions including control [60], estimation [61, 46, 44], and performance optimization [62, 63] of VCSs require accurate predictions of its dynamics which is obtained from simulation models and digital twins [64, 65]. State-of-the-art prediction models of VCSs are derived using fundamental physics principles that govern the individual components in VCS and the overall system [66]. Although high prediction fidelity can be achieved by physics-based models of VCSs, their development and deployment face several computational challenges, despite recent efforts to simplify and augment them with efficient data-driven learning modules [67]. Their computational complexity becomes a major bottleneck for efficient deployment in scenarios requiring repeated model linearizations and forward simulations. Consequently, there is a need for efficient surrogate models of high-fidelity physics-based models of VCSs which replicate dynamics of the underlying system.

Vapor compression systems of similar technical specifications exhibit similar dynamic behaviors since the governing physics are identical. However, each system may exhibit a unique dynamic response due to factors such as unit-to-unit manufacturing variability, differences in installation configurations, and differences in refrigerant quantity present in the system. For the purpose of this study, we particularly focus on the dynamics variability induced by the varying refrigerant quantity across different systems which may arise due to installation errors, refrigerant leakage over equipment life cycle, or refrigerant replacement during regular maintenance.

Generally, dataset for a particular VCS is typically limited and expensive to obtain as it may require sophisticated instrumentation. However, an entity of interest (e.g., manufacturer or supplier) may have access to a collectively large operational dataset of different VCSs located across many customer sites. Additionally or alternatively, a training dataset can also be generated from high fidelity physics-based models of VCSs. Thus, surrogate modeling efforts of VCSs may significantly benefit from the proposed meta-learning approach. A large source dataset collected across different systems can be used for learning a general NSSM for VCSs, while the limited context dataset from a query system can be used for fine-tuning the NSSM weights.

6.1.1. Data Generation

The training dataset is generated using a physics-based simulator of the VCS. The model includes two heat exchangers, discretized into four finite volumes each, and algebraic models for the compressor and expansion valve. The heat exchangers consist of components that describe the thermal behavior of the pipe wall, air flow, and one-dimensional refrigerant pipe-flow, forming an index-1 system of differential algebraic equations (DAEs). The refrigerant flow is modeled using conservation laws of mass, momentum, and energy. The model is represented by a set of ordinary differential equations (ODEs) obtained after order reduction of DAEs via Pantelides' algorithm. While compressor frequency and expansion valve position are the control inputs to the model, the state vector consists the temperature $\theta[i]$, pressure $\mathsf{P}[i]$, and specific enthalpy $\mathsf{h}[i]$ for each finite volume $i \in \{1 \cdots 8\}$, making $n_u = 2$ and $n_x = 24$. Only 6 of the 24 states are directly measured

which include pressures and temperatures at select finite volumes of the heat exchangers [44]. Specifically, the measurement vector is given by

$$y = \begin{bmatrix} \mathsf{P}[1] & \theta[2] & \theta[4] & \theta[6] & \mathsf{P}[8] & \theta[8] \end{bmatrix} \in \mathbb{R}^6. \tag{25}$$

The training dataset consisting 250 trajectories was generated using the simulation model such that each trajectory corresponded to a system with a different refrigerant mass. The model initializes all variables to their default values. A set of 250 initial conditions for the training dataset was created by simulating the model to a steady state over a 50-second duration with constant control inputs, as shown in Fig. 6[C]. The compressor frequency and EEV position were set to 50 Hz and 300 counts for all runs. This reduces the impact of default initialization and enables the creation of a meaningful, controlled set of initial conditions to test our proposed methods. The scenario of varying refrigerant mass was emulated by selecting different pipe lengths for HEXs in different trajectories. We note that, identical default initializations of state variables and different pipe lengths amount to variable refrigerant mass across different trajectories. The set of 250 values for pipe length were randomly sampled from the uniform distribution $\mathcal{U}(20, 40)$ m. The variability of the dynamics due to variation in pipe lengths and refrigerant mass is shown in Fig. 6[C] for a representative subset of state variables.

A set of 250 system trajectories was generated by simulating the model over 500 seconds with the initial conditions described above and sampled at 0.1 second intervals. To generate a rich dataset with persistently excited inputs, the expansion valve position and compressor frequency (Hz) were randomly sampled from the uniform distributions $\mathcal{U}(282, 312)$ and $\mathcal{U}(40, 80)$, respectively. The set of 250 trajectories was split 80-20 for source and query dataset, wherein only first the 20 seconds of the trajectories were used as context data for fine-tuning. Furthermore, 200 source system trajectories were split into 60-40 for training and validation.

High-fidelity simulators for VCS applications are based on system physics, inherently satisfying fundamental physical constraints like the pressure gradient constraint (non-increasing pressure in the direction of refrigerant flow). To ensure consistency with physical laws, state estimates from a learned NSSM and EKF must also satisfy these constraints. We explicitly incorporate these constraints into the NSSM and filtering algorithm. Specifically, can write the pressure gradient constraint as a linear inequality $Gx \leq 0$, and utilize the discussion in Section 4.1 to improve the estimates.

6.1.2. Results and Discussion

State estimation results of the EKF (7a) that uses the NSSM (16) for predictions are shown in Fig. 7. We considered two variations of NSSMs for a comparison. An NSSM whose weights were fine-tuned using Supplementary Algorithm S2 is referred to as Meta-NSSM, whereas NSSM learned without the fine-tuning is referred to as No-Meta-NSSM. EKFs using these two variations of NSSMs were tested on query systems. Reference trajectories of one of the query systems and those estimated by EKFs along with the associated

confidence bounds are shown in Fig. 7.

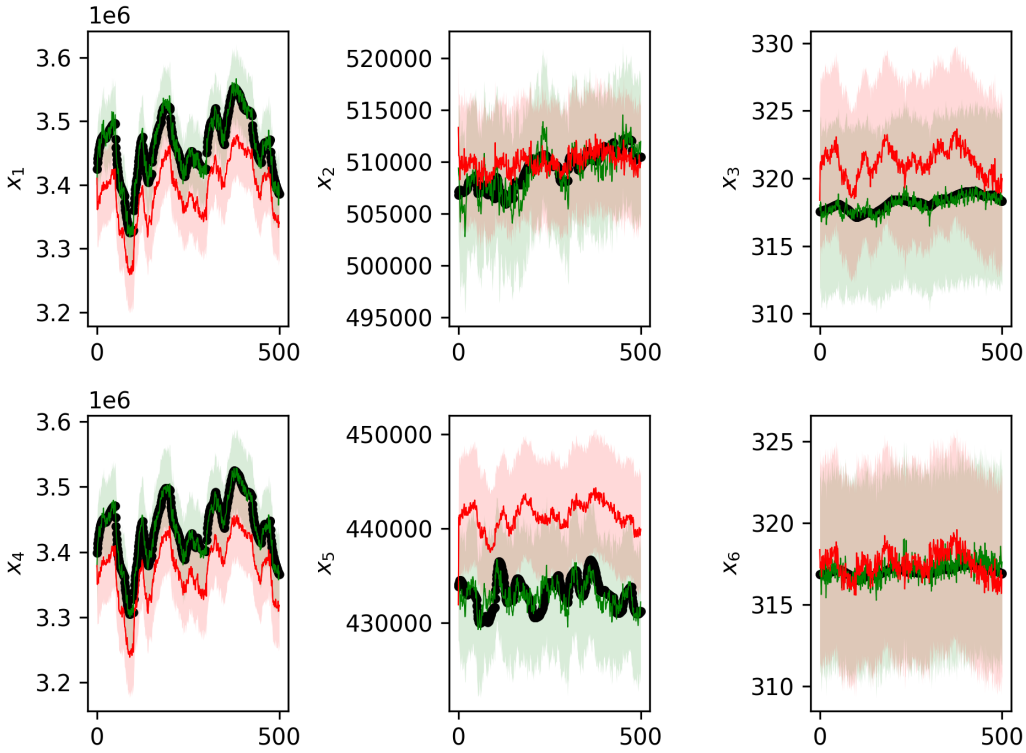

Figure 7: Reference state trajectories (black) and those estimated by EKF using Meta-NSSM (green) and No-Meta-NSSM (red) as the prediction models. Shaded area around the solid lines denote 3σ-bounds.

Meta-NSSM-EKF demonstrated significantly smaller estimation errors than No-Meta-NSSM-EKF as observed in Fig. 7. The superior estimation accuracy of Meta-NSSM-EKF can be primarily attributed to better prediction capabilities of the Meta-NSSM since its weights are fine-tuned or adapted for a particular query system. On the other hand, No-Meta-NSSM-EKF uses a general prediction model learned for a family of VCSs without system-specific adaptations, thus, leading to an overall degraded estimation performance. These findings align with those reported in the physics-informed estimation literature [68].

6.2. Indoor Localization with Unknown Magnetic Fields

An another motivating application for combining physics-informed NSSMs, meta-learning, and state estimation is in magnetic-field-based indoor positioning. Magnetic materials present in buildings cause anomalies in the ambient magnetic field [69, 70], which can be leveraged for localization. However, several challenges hinder high-accuracy positioning using magnetic fields. The primary challenge is that the function mapping the magnetometer position to the magnetic field is unique for each indoor environment, lacks a specific structure, and must adhere to physics-informed constraints (i.e., it must be a curl-free vector field

due to the absence of local currents). As a result, it is necessary to identify the unknown magnetic field, typically using noisy measurements obtained online.

Using costly offline instrumentation, we can gather a very limited amount of state-output data from the actual system, represented as

$$\mathfrak{D}_{\text{tgt}} = \{(x_{0:T^\star}, u_{0:T^\star}, y_{0:T^\star})|\theta^\star\}$$

collected over the time span $0, \ldots, T^\star$. Our objective is to develop an NSSM for the query system with limited data, enforcing physics-informed constraints within the NSSM architecture, and subsequently use this NSSM for online state estimation in the presence of sensor and process noise. Herein, we demonstrate physics-constrained meta-learning for learning the magnetic field with subsequent state estimation for positioning.

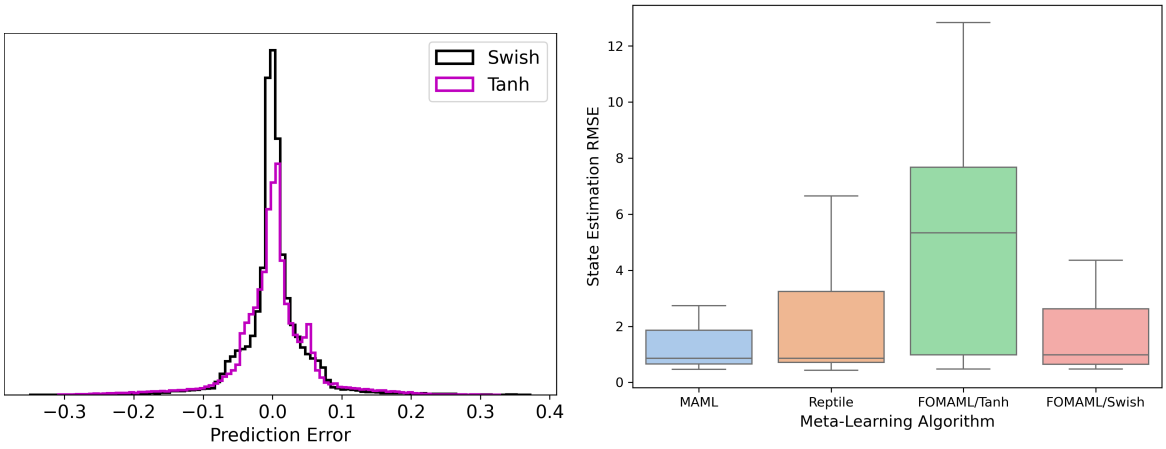

Figure 8: (*Left*) Histograms of predictive error distribution between swish and tanh MAML. (*Right*) Boxplots demonstrating state estimation RMSE across different meta-learning EKF for 20 unknown query systems.

6.2.1. System Description

We consider the scenario from [48], adapted to a recursive setting as in [71]. In particular, we use the magnetic-field model from [48] and have obtained ground truth data for evaluating the proposed approach. Consider a sphere of a radius of r_0m centered at the origin, uniformly magnetized with $\mathcal{M} = [m_X, m_Y, 0]^\top$A/m. The magnetic-field function h_θ, which maps the position $p := [p_X, p_Y]$ to y, is given by

$$h_\theta = \begin{cases} -\mathcal{M}/3 & \text{if } r < r_0 \\ \frac{m_0}{4\pi}\left(\mathcal{M}/r^3 + 3/r^5(\mathcal{M}^\top p)p\right) & \text{if } r \geq r_0 \end{cases} \tag{26}$$

where $m_0 = 4/3\pi r_0^3$ and $r = \|p\|$. Thus, the measurement model is: $y_t = h_\theta(x_t) + \eta_t$, where $\eta_t \sim \mathcal{N}(0, Q_t^\eta)$ is zero-mean Gaussian noise with covariance Q_t^η.

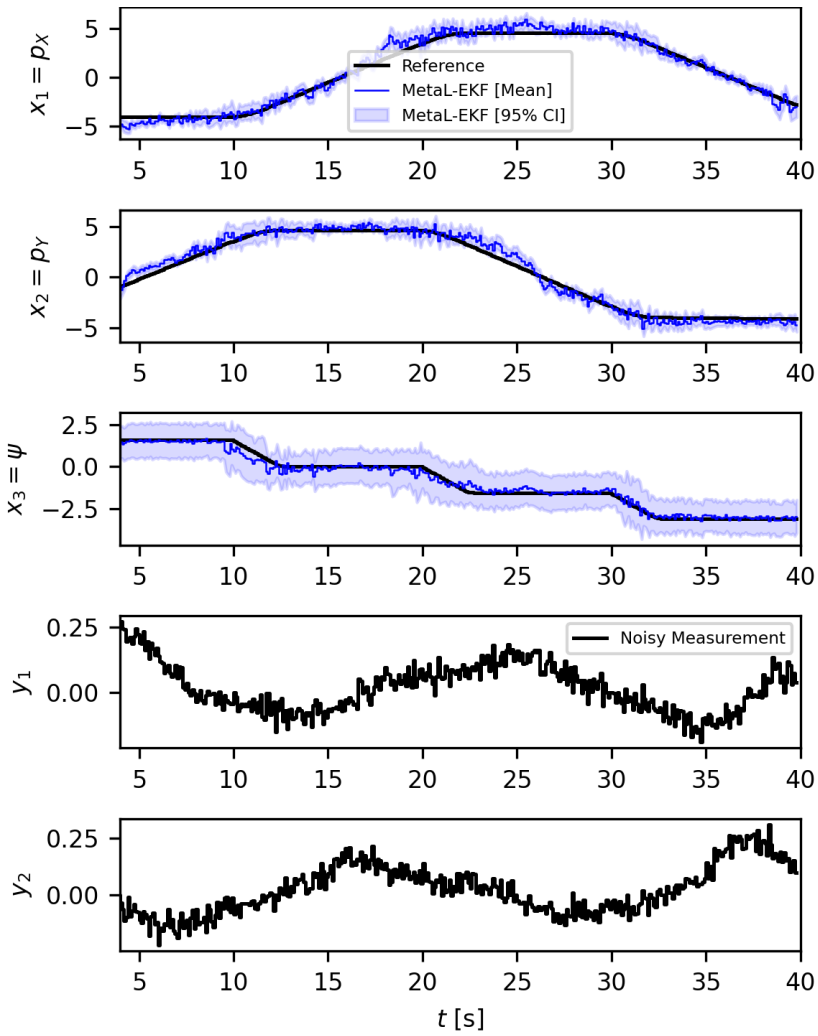

Figure 9: Performance of MetaL-EKF with MAML-FT/Swish for a randomly chosen test vehicle under uncertain magnetic field. Horizontal axis starts at $t = 4$ s to reflect that previous data was used as context for meta-inference.

We consider a mobile wheeled robot with car-like kinematics, described by a kinematic single-track model operating in the XY-plane. This model has three states: the global (planar) position and the heading angle, represented as $x = [p_X, p_Y, \psi]$. The wheel-speed measurements directly provide the velocity v_X. The continuous-time model is

$$\dot{x} = \begin{bmatrix} v_X \cos(\psi + \beta)/\cos(\beta) \\ v^X \sin(\psi + \beta)/\cos(\beta) \\ v^X \tan(\delta)/L \end{bmatrix}, \tag{27}$$

where $L = l_f + l_r$ are the distances from the origin to the front and rear wheel axle, respectively, $\beta = \arctan(l_r \tan(\delta)/L)$ is the kinematic body-slip angle where δ is the steering angle, and the velocity is related to the wheel speeds by $v_X = (\omega_f + \omega_r)R_w/2$; R_w is the wheel radius.

Following temporal discretization, (27) can be expressed as

$$x_{k+1} = f_\theta(x_k, u_t) + w_t, \tag{28}$$

where $u_t = [\delta, \omega_f, \omega_r]$ denotes the control inputs, and $w_t \sim \mathcal{N}(0, Q_t^w)$ is Gaussian noise which accounts for the model mismatch. The steering controller is a simple pure-pursuit controller tracking a reference path $p_r(t)$.

In the context of (1), the unknown parameters θ include the maximum steering angle δ, the wheelbase parameters l_f, l_r, which influence the vehicle dynamics; as well as the the magnetic-field radius r_0, and the magnetization parameters m_X, m_Y, which affect the magnetic vector field.

6.2.2. Implementation Details

To generate a training dataset, the following parameter ranges were considered: $l_f, l_r \in [0.02, 0.25]$, $\delta \in [10, 20]$, $m_X, m_Y \in [-1.5, 1.5]$, and $r_0 \in [1.5, 4.5]$. A dataset containing 80 source and 20 query systems was constructed by randomly sampling parameters from the specified ranges and executing simulations over the duration of 40s that yielded x, u, and y data for each system tracking identical reference paths. The trajectories were sampled at 0.1s intervals. Despite the identical references to be tracked, significant variation within $\mathfrak{D}_{\text{src}}$ and $\mathfrak{D}_{\text{tgt}}$ was observed due to differences in θ (plot not shown due to limited space).

The dimension of the latent space was selected to be $n_\psi = 128$ following a preliminary architecture search. Each NSSM operator was implemented with `swish`-activated, fully connected layers. The encoder and two decoders were structured as $\mathcal{E} : 6 - 256 - 128 - 128 - 64 - 32 - 32 - 128$, $\mathcal{D}_x : 128 - 32 - 64 - 128 - 256 - 3$ (outputting 3 updated states), and $\mathcal{D}_y : 128 - 64 - 64 - 128 - 128 - 256 - 256 - 1$ (outputting a scalar potential). The state transition operator $\mathcal{A}_\psi$ used six fully connected layers with a hidden dimension of 256. While the first few layers of $\mathcal{E}$ were fixed after meta-training, the last two layers of $\mathcal{E}_{xu}$ and all layers of $\mathcal{A}_\psi$, $\mathcal{D}_x$, and $\mathcal{D}_y$ were adapted with the query system data during meta-inference. We utilize the results of Section 4.2 to enforce curl-free constraint on the learned magnetic field. We emphasize that in this case study, $\mathcal{P}_y \equiv \vec{\nabla}$ is not a trainable neural network; it simply represents the vector differential operator that differentiates the output of $\mathcal{D}_y$ with respect to p.

While one may argue that this network is unnecessarily deep and the base-learner could have been constructed with fewer layers, the reason we chose many layers (other than just to improve expressivity across the family of source and query systems) was to investigate whether the `swish` activation truly avoids vanishing gradients in deep networks, and whether some empirical advantages may be obtained compared to other activation functions such as the `tanh` function that have been previously considered.

6.2.3. Results and Discussion

The meta-learned NSSM was trained offline on the source dataset for 30,000 epochs using the Adam optimizer. The learning rates were fixed at $\beta_{\text{out}} = 10^{-4}$ and $\beta_{\text{in}} = 10^{-3}$, and $M = 10$ adaptation iterations

were allowed in accordance with Supplementary Algorithm S1. Training on an NVIDIA 3090X GPU took 22 hours, with weights saved when validation loss decreased.

For meta-inference, 20% of the query system data (the initial $T_c = 8$s of the total $T_\ell = 40$s) was used. The NSSM's performance was then evaluated based on prediction and estimation errors for the remaining 80% of the query system's trajectory, which it had not seen. The CPU time for meta-inference adaptation of the non-fixed layers was 0.58 seconds on average.

Our numerical results are presented in Fig. 8, Fig. 9, and Supplementary Fig. S4. In Fig. S4, we compare the magnetic field estimation error $y - \hat{y}$ between the the true field h_θ, obtained by evaluating h_θ in (26) parameterized with $m_X = 1$, $m_Y = 1$, $r_0 = 3$m and the estimated magnetic field using the meta-learned `swish`-activated (black) and `tanh`-activated (magenta) NSSM, evaluated over a regular 200×200 grid in a 2-D spatial domain. Clearly, for both estimates, the error is small: that is, the learned NSSM closely approximates the magnetic field throughout the domain, with noticeable differences only around a small neighborhood of the region of discontinuity at $r_0 = 3$m. The `tanh`-activated NSSM induces larger approximation errors than the `swish`-activated NSSM: this is cross-checked by computing the root-mean-squared error (RMSE) of approximation on the 100×100 grid: for the `swish` activated NSSM, the RMSE is 6.72×10^{-4} compared to 8.03×10^{-4} for `tanh`-activation. The distribution of the errors is also shown via histograms in the left subplot of Fig. 8, where we see that the `swish`-based NSSM has a tighter variance around zero, with many more time-points with estimation errors near-zero.

We used the meta-learned NSSM as a predictive model for implementing an EKF of the form (7), referred to as the *MetaL-EKF*. In the right subplot of Fig. 8, we compare the state estimation accuracy of MetaL-EKF implemented with different meta-learned NSSMs. Boxplots of the cumulative state estimation error norm $\sum \|x - \hat{x}\|$ are reported for 20 unknown query systems. In particular, we consider four variants of meta-learning, two of which are the most popular meta-learning algorithms. The first boxplot (blue) is MAML approach with the proposed `swish` activation, the second is Reptile (orange) with `swish` activation, the third (green) is FO-MAML with `tanh` activation, and the final (pink) is the proposed FO-MAML approach with `swish` activation. It is immediately clear that the `swish` activation consistently improves performance. The full fine-tuned MAML outperforms the rest, although the Reptile and FO-MAML algorithms are comparable in average performance, with FO-MAML showing better robustness (tighter confidence) across query systems.

The states and outputs for a particular query system (chosen randomly) is shown in Fig. 9, along with their estimates and 95% confidence intervals, with 10% data used for adaptation (first 4 seconds). MetaL-EKF demonstrates good estimation accuracy for the most part of the unseen trajectory. Relatively larger deviations of estimated state trajectories from the true trajectory and larger state error covariances at some points can be attributed to sharp turns in the vehicle trajectory. Despite these infrequent deviations, MetaL-EKF quickly decreases the gap between the estimated and true trajectories as the vehicle moves past the turning maneuvers, which can be attributed to the predictive capability of the meta-learned MAML+FT/Swish

model.

7. Conclusions and Future Work

In this work, we proposed a gradient-based meta-learning framework for neural state-space model (NSSM) identification, with a focus on rapid adaptation using limited data from a query system by leveraging data from similar source systems, where similarity is defined using deviation of output trajectories with rigorous assumptions on the class of systems considered. The proposed approach integrates physical constraints directly into the NSSM architecture, improving physical consistency and predictive performance. We demonstrated the efficacy of our method through both benchmark examples and real-world case studies in indoor localization and vapor compression systems, and highlighted the benefits of subnetwork fine-tuning strategies over full model adaptation.

Our findings suggest that meta-learned NSSMs offer strong potential for black-box system identification in practical settings where fast adaptation with limited data is critical. Furthermore, we provided empirical evidence supporting the use of task-specific fine-tuning mechanisms and detailed the advantages of MAML and ANIL algorithms under different architectural configurations and data availability scenarios. We also demonstrate empirically that surgical fine-tuning for system identification involves fixing part of the encoder head, and allowing the rest of the encoder, along with the transition operator weights, to adapt to new systems.

Future work includes expanding the class of physical constraints considered, integrating hybrid modeling approaches that blend first-principles and data-driven components, and investigating theoretical bounds for generalization in meta-learning when the data is generated by a dynamical system and not independently sampled.

References

[1] J. M. Zamarreño, P. Vega, State-space neural network. Properties and application, Neural networks 11 (6) (1998) 1099–1112.

[2] Y. Bao, J. M. Velni, A. Basina, et al., Identification of state-space linear parameter-varying models using artificial neural networks, IFAC-PapersOnLine 53 (2) (2020) 5286–5291.

[3] M. Forgione, M. Mejari, D. Piga, Learning neural state-space models: do we need a state estimator?, arXiv preprint arXiv:2206.12928 (2022).

[4] G. Beintema, R. Toth, M. Schoukens, Nonlinear state-space identification using deep encoder networks, in: Learning for Dynamics and Control, PMLR, 2021, pp. 241–250.

[5] C. M. Legaard, T. Schranz, G. Schweiger, et al., Constructing neural network-based models for simulating dynamical systems, arXiv preprint arXiv:2111.01495 (2021).

[6] M. Forgione, D. Piga, Model structures and fitting criteria for system identification with neural networks, in: 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), IEEE, 2020, pp. 1–6.

[7] M. Forgione, D. Piga, dynoNet: A neural network architecture for learning dynamical systems, International Journal of Adaptive Control and Signal Processing 35 (4) (2021) 612–626.

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

[9] Y. Liu, R. Tóth, M. Schoukens, Physics-guided state-space model augmentation using weighted regularized neural networks, IFAC-PapersOnLine 58 (15) (2024) 295–300.

[10] E. Skomski, J. Drgoňa, A. Tuor, Automating discovery of physics-informed neural state space models via learning and evolution, in: Learning for Dynamics and Control, PMLR, 2021, pp. 980–991.

[11] D. Masti, A. Bemporad, Learning nonlinear state–space models using autoencoders, Automatica 129 (2021) 109666.

[12] L. C. Iacob, G. I. Beintema, M. Schoukens, et al., Deep identification of nonlinear systems in Koopman form, in: 2021 60th IEEE Conference on Decision and Control (CDC), IEEE, 2021, pp. 2288–2293.

[13] T. Bertalan, F. Dietrich, I. Mezić, et al., On learning Hamiltonian systems from data, Chaos: An Interdisciplinary Journal of Nonlinear Science 29 (12) (2019) 121107.

[14] B. O. Koopman, J. van Neumann, Dynamical systems of continuous spectra, Proceedings of the National Academy of Sciences 18 (3) (1932) 255–263.

[15] B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature communications 9 (1) (2018) 1–10.

[16] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.

[17] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, in: Advances in Neural Information Processing Systems, 2016, pp. 3630–3638.

[18] T. Munkhdalai, H. Yu, Meta networks, in: International conference on machine learning, PMLR, 2017, pp. 2554–2563.

[19] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, T. Lillicrap, Meta-learning with memory-augmented neural networks, in: Proceedings of The 33rd International Conference on Machine Learning, 2016, pp. 1842–1850.

[20] A. Vettoruzzo, M.-R. Bouguelia, J. Vanschoren, T. Rögnvaldsson, K. Santosh, Advances and challenges in meta-learning: A technical review, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (7) (2024) 4763–4779.

[21] L. Xin, L. Ye, G. Chiu, et al., Identifying the dynamics of a system by leveraging data from similar systems, in: 2022 American Control Conference (ACC), 2022, pp. 818–824.

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

[23] D. Piga, F. Pura, M. Forgione, On the adaptation of in-context learners for system identification, IFAC-PapersOnLine 58 (15) (2024) 277–282.

[24] A. Bemporad, Linear and nonlinear system identification under ℓ_1-and group-lasso regularization via L-BFGS-B, arXiv:2403.03827 (2024).

[25] S. M. Richards, N. Azizan, J. E. Slotine, et al., Adaptive-control-oriented meta-learning for nonlinear systems, CoRR abs/2103.04490 (2021). arXiv:2103.04490.

[26] E. Arcari, A. Carron, M. N. Zeilinger, Meta learning MPC using finite-dimensional Gaussian process approximations, arXiv preprint arXiv:2008.05984 (2020).

[27] D. Muthirayan, P. P. Khargonekar, Meta-learning guarantees for online receding horizon learning control, arXiv preprint arXiv:2010.11327 (2020).

[28] J. Yan, A. Chakrabarty, A. Rupenyan, et al., 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 (CASE), 2024, pp. 1910–1915.

[29] S. Zhan, G. Wichern, C. Laughman, et al., Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization, Energy and Buildings 270 (2022) 112278.

[30] A. Chakrabarty, Optimizing closed-loop performance with data from similar systems: A Bayesian meta-learning approach, in: Proc. IEEE Conf. Dec. Control (CDC), IEEE, 2022, p. To appear.

[31] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International conference on machine learning, PMLR, 2017, pp. 1126–1135.

[32] A. Antoniou, H. Edwards, A. Storkey, How to train your MAML, in: Proc. of the Int. Conf. on Learning Representations (ICLR), 2019.

[33] A. Raghu, M. Raghu, S. Bengio, et al., Rapid learning or feature reuse? Towards understanding the effectiveness of MAML, in: Proc. of the Int. Conf. on Learning Representations (ICLR), 2019.

[34] A. Rajeswaran, C. Finn, S. M. Kakade, et al., Meta-learning with implicit gradients, Advances in neural information processing systems 32 (2019).

[35] A. Nichol, J. Schulman, Reptile: a scalable metalearning algorithm, arXiv preprint arXiv:1803.02999 2 (3) (2018) 4.

[36] E. M. Chayti, M. Jaggi, A new first-order meta-learning algorithm with convergence guarantees, arXiv preprint arXiv:2409.03682 (2024).

[37] A. Mauroy, Y. Susuki, I. Mezić, Koopman Operator In Systems And Control, Springer, 2020.

[38] J. L. Crassidis, J. L. Junkins, Optimal Estimation of Dynamic Systems, Chapman and Hall/CRC, 2004.

[39] M. Grewal, Identifiability of linear and nonlinear dynamical systems, IEEE Transactions on automatic control 21 (6) (2003) 833–837.

[40] E. Walter, L. Pronzato, On the identifiability and distinguishability of nonlinear parametric models, Mathematics and computers in simulation 42 (2-3) (1996) 125–134.

[41] D. Silvestre, P. Rosa, C. Silvestre, Distinguishability of discrete-time linear systems, International Journal of Robust and Nonlinear Control 31 (5) (2021) 1452–1478.

[42] Q. Chen, C. Shui, M. Marchand, Generalization bounds for meta-learning: An information-theoretic analysis, Advances in Neural Information Processing Systems 34 (2021) 25878–25890.

[43] K. Fukuda, A. Prodon, Double description method revisited, in: Franco-Japanese and Franco-Chinese conference on combinatorics and computer science, Springer, 1995, pp. 91–111.

[44] V. M. Deshpande, C. R. Laughman, Y. Ma, et al., Constrained smoothers for state estimation of vapor compression cycles, in: Proc. American Control Conference, 2022, pp. 2333–2340.

[45] D. Simon, D. L. Simon, Constrained Kalman filtering via density function truncation for turbofan engine health estimation, International Journal of Systems Science 41 (2) (2010) 159–171.

[46] V. M. Deshpande, C. R. Laughman, Multi-pass extended kalman smoother with partially-known constraints for estimation of vapor compression cycles, in: 22nd IFAC World Congress, 2023.

[47] N. Amor, G. Rasool, N. C. Bouaynaya, Constrained state estimation – a review, arXiv:1807.03463 (2022).

[48] N. Wahlström, M. Kok, T. B. Schön, et al., Modeling magnetic fields using Gaussian processes, in: Int. Conf. Acoustics, Speech, and Signal Process., 2013.

[49] J. Hendriks, C. Jidling, A. Wills, et al., Linearly constrained neural networks, arXiv preprint arXiv:2002.01600 (2020).

[50] S. J. Colley, Vector Calculus, 4th Edition, Pearson, Upper Saddle River, NJ, 2011.

[51] P. Ramachandran, B. Zoph, Q. V. Le, Swish: A self-gated activation function, arXiv preprint arXiv:1710.05941 (2017).

[52] W. Lee, H. Yu, X. Rival, H. Yang, On correctness of automatic differentiation for non-differentiable functions, Advances in neural information processing systems 33 (2020) 6719–6730.

[53] J.-P. Noel, M. Schoukens, Hysteretic benchmark with a dynamic nonlinearity, in: Workshop on nonlinear system identification benchmarks, 2016, pp. 7–14.

[54] M. Schoukens, R. Tóth, On the initialization of nonlinear lfr model identification with the best linear approximation, IFAC-PapersOnLine 53 (2) (2020) 310–315.

[55] M. Forgione, D. Piga, *dynoNet*: A neural network architecture for learning dynamical systems, International Journal of Adaptive Control and Signal Processing 35 (4) (2021) 612–626.

[56] A. F. Esfahani, P. Dreesen, K. Tiels, et al., Polynomial state-space model decoupling for the identification of hysteretic systems, IFAC-PapersOnLine 50 (1) (2017) 458–463.

[57] J. Belz, T. Münker, T. O. Heinz, et al., Automatic modeling with local model networks for benchmark processes, IFAC-PapersOnLine 50 (1) (2017) 470–475.

[58] A. Paszke, S. Gross, F. Massa, et al., PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[59] J. Yosinski, J. Clune, Y. Bengio, et al., How transferable are features in deep neural networks?, Advances in neural information processing systems 27 (2014).

[60] S. Bortoff, P. Schwerdtner, C. Danielson, et al., H_∞ loop-shaped model predictive control with heat pump application, in: 18th European Control Conference, 2019, pp. 2386–2393.

[61] A. Chakrabarty, E. Maddalena, H. Qiao, et al., Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics, Energy and Buildings 253 (2021) 111460.

[62] M. Khosravi, A. Eichler, N. Schmid, et al., Controller tuning by Bayesian optimization an application to a heat pump, in: Proc. 18th European Control Conference, IEEE, 2019, pp. 1467–1472.

[63] A. Chakrabarty, C. Danielson, S. A. Bortoff, et al., Accelerating self-optimization control of refrigerant cycles with bayesian optimization and adaptive moment estimation, Applied Thermal Engineering 197 (2021) 117335.

[64] C. Vering, S. Borges, D. Coakley, et al., Digital twin design with on-line calibration for HVAC systems in buildings, in: Proceedings of Building Simulation 2021: 17th Conference of IBPSA, Vol. 17 of Building Simulation, IBPSA, Bruges, Belgium, 2021, pp. 2938–2945.

[65] C. R. Laughman, V. M. Deshpande, H. Qiao, et al., Digital Twins of Vapor Compression Cycles: Challenges and Opportunities, in: Proc. Int. Congress of Refrigeration (ICR), 2023.

[66] H. Qiao, V. Aute, R. Radermacher, Transient modeling of a flash tank vapor injection heat pump system–Part I: Model development, International Journal of Refrigeration 49 (2015) 169–182.

[67] R. Chinchilla, V. M. Deshpande, A. Chakrabarty, et al., Learning residual dynamics via physics-augmented neural networks: Application to vapor compression cycles, in: 2023 American Control Conference (ACC), 2023, pp. 4069–4076.

[68] V. M. Deshpande, A. Chakrabarty, A. P. Vinod, et al., Physics-constrained deep autoencoded kalman filters for estimating vapor compression system states, IEEE Control Systems Letters 7 (2023) 3483–3488.

[69] B. Li, T. Gallagher, A. G. Dempster, et al., How feasible is the use of magnetic field alone for indoor positioning?, in: Int. Conf. Indoor Positioning and Indoor Navigation, 2012.

[70] M. Angermann, M. Frassl, M. Doniec, et al., Characterization of the indoor magnetic field for applications in localization and mapping, in: Int. Conf. Indoor Positioning and Indoor Navigation, 2012.

[71] K. Berntorp, M. Menner, Constrained gaussian-process state-space models for online magnetic-field estimation, in: Proc. 26th Int. Conf. on Information Fusion (FUSION), IEEE, 2023, pp. 1–7.