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

Physics-Constrained Meta-Learning for Online Adaptation and Estimation in Positioning Applications

Chakrabarty, Ankush; Deshpande, Vedang M.; Wichern, Gordon; Berntorp, Karl

TR2024-180 December 18, 2024

Abstract

Deep neural state-space models (NSSMs) using autoencoders are highly effective for system identification. Recent advances in meta-learning allow these models to quickly adapt to specific dynamical systems within families of similar systems. Leveraging advanced automatic differentiation tools, meta-learned NSSMs can serve as predictive models for online state estimation, especially when dealing with systems that have uncertain parameters or unmodeled dynamics. This is particularly relevant in magnetic-field positioning applications, where a magnetometer's motion dynamics may be uncertain, and measurements are taken within an unknown magnetic vector field. In this paper, we present a meta-learning framework that trains 'physics-constrained' NSSMs on a diverse dataset of motion dynamics and magnetic vector fields. These models incorporate physics-informed constraints to learn a curl-free magnetic field. The meta-learned NSSM can rapidly adapt to a new motion model and magnetic field in a few-shot manner (without explicitly estimating the underlying physical parameters) and can be used as a predictive model for state estimation in positioning tasks.

IEEE Conference on Decision and Control (CDC) 2024

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

Physics-Constrained Meta-Learning for Online Adaptation and Estimation in Positioning Applications

Ankush Chakrabarty†, Vedang Deshpande, Gordon Wichern, Karl Berntorp

***Abstract*—Deep neural state-space models (NSSMs) using autoencoders are highly effective for system identification. Recent advances in meta-learning allow these models to quickly adapt to specific dynamical systems within families of similar systems. Leveraging advanced automatic differentiation tools, meta-learned NSSMs can serve as predictive models for online state estimation, especially when dealing with systems that have uncertain parameters or unmodeled dynamics. This is particularly relevant in magnetic-field positioning applications, where a magnetometer's motion dynamics may be uncertain, and measurements are taken within an unknown magnetic vector field. In this paper, we present a meta-learning framework that trains 'physics-constrained' NSSMs on a diverse dataset of motion dynamics and magnetic vector fields. These models incorporate physics-informed constraints to learn a curl-free magnetic field. The meta-learned NSSM can rapidly adapt to a new motion model and magnetic field in a few-shot manner (without explicitly estimating the underlying physical parameters) and can be used as a predictive model for state estimation in positioning tasks.**

I. Introduction

Neural state-space models (NSSMs) have recently been touted for nonlinear system identification. A key feature of NSSMs is their ability to accurately predict the behavior of black-box (or unmodeled) dynamical systems purely from sequential data streams collected from them [1]–[5]. Explicitly incorporating physics-based information, such as physical constraints, within the NSSM architecture can both reduce the amount of data needed for training and yield state estimates that are consistent with domain knowledge; c.f. [6], [7] for a more thorough survey outside of the estimation context. Physics-constrained NSSMs are often trained by supervised learning: model parameters have been trained offline with significant data. As a consequence, they are rarely equipped to adapt to applications that have limited data that is obtained after deployment.

Meta-learning, or (more generally) 'in-context learning', has shown promising results in settings where a single target system may provide small data, but actionable data is available from systems that are *similar* to the target system [8]–[10]. Data for the family of similar systems may be obtained via high-fidelity simulation models for a variety of internal parameters, or via experiments on similar systems deployed in the past. The gradient-based meta-learning methodology comprises a bilevel training procedure [11]. First, a model is *meta-trained* on a dataset consisting of data from a number of similar systems, where the goal is to learn common patterns or common knowledge that can be applied to a new target system. Second, an online *meta-inference* procedure rapidly adapts the meta-trained model for a target system with limited data and in significantly fewer online training iterations. Though not for SSMs, meta-learning has been proposed for optimization [10], [12], adaptive control [13], and receding horizon control [14]–[16].

A motivating application for combining physics-informed NSSMs, meta-learning, and state estimation is in magnetic-field-based indoor positioning [17]. Magnetic materials present in buildings cause anomalies in the ambient magnetic field [18], [19], which can be leveraged for localization. However, to get high-accuracy positioning some challenges need to be overcome. Most critically, the function mapping the magnetometer position to the magnetic field is unique for each indoor environment, does not have specific structure, and needs to enforce physics-informed constraints (i.e., it has to be a curl-free vector field due to the absence of exciting local currents). Consequently, there is a need to identify the unknown magnetic field, generally using noisy measurements obtained online. Herein, we demonstrate the usefulness of physics-constrained meta-learning for learning the magnetic field with subsequent state estimation for positioning.

II. Preliminaries

A. Motivation and Problem Statement

We consider a family of parameterized discrete-time nonlinear systems given by

$$x_{k+1} = f_\theta(x_k, u_k), \quad y_k = h_\theta(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ denotes the state of the system at time instant $k \in \mathbb{N}$ with x_0 as an initial state, $y \in \mathbb{R}^{n_y}$ denotes the measured outputs, f_θ denotes the unmodeled dynamics, h_θ denotes an unmodeled output function, and $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$ denotes a vector of unknown model parameters. We assume that Θ, the set of admissible parameters for the system, is known from domain expertise; *e.g.*, we know a vehicle has a tire radius and an approximate range for it based on observations of similar vehicles. Motivated by our application, we further assume that the state contains positional information, i.e., 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.

Of the family of systems described above, suppose the actual target system exhibits dynamics (1) with unknown $\theta^\star \in \overline{\Theta}$. We focus on the applications where $f_{\theta^\star}$ and

Corresponding author. Phone: +1 (617) 758-6175. Email: achakrabarty@ieee.org. All authors are affiliated with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA.

$h_{\theta^\star}$ may not be explicitly accessed due to various reasons: for example, a general structure of h may be unknown in magnetic positioning applications and needs to be learned online via measurement data, as described in the introduction. Another example is where f and h are represented in a high-fidelity simulation software such as 'digital twin', may contain black-box components that can not be directly accessed. Such digital twins typically allow simulations of trajectories from (1) for a range of specified parameters and control inputs, and this simulation data can be used for neural system identification.

In many practical settings, we need to estimate the state x online from measurements y without completely knowing f, h, or θ. As shown previously in [7], [20], a viable approach is to propose NSSMs for f and h, and subsequently use these models for model-based state estimation. Motivated by indoor positioning applications using the magnetic field, we are faced with two further challenges that prevent straightforward application of the method in [7]. The first is that the function h in such applications is based on a magnetic field, and therefore should satisfy physics-informed constraints: specifically it needs to be curl-free [21]; i.e., $\vec{\nabla} \times h = 0$, 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. Second, with the help of expensive offline instrumentation, we can acquire very limited state-output data from the actual system, denoted by

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

obtained over the time span $\{0, \dots, T^\star\}$. Due to this scarcity of target system data, it is unrealistic to expect that an NSSM trained solely on this data will exhibit satisfactory predictive performance or be able to generalize. These challenges define our problem: to identify a neural predictive model for the target system with limited data, capable of enforcing physics-informed constraints within the NSSM architecture, and to subsequently employ this NSSM for online state estimation despite sensor noise and process uncertainty.

B. Proposed Solution

To reduce the data required from the target $\theta^\star$-system, we leverage simulation models to obtain data from $N_{\text{src}} \in \mathbb{N}$ dynamical systems that are similar to the target system. These source systems adhere to the form (1) and are parameterized by $\theta^\ell \in \Theta \setminus \theta^\star$ for $\ell = 1, 2, \dots, N_{\text{src}}$. 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

$$\mathcal{D}_{\text{src}} \triangleq \left\{(x_{0:T_\ell}, u_{0:T_\ell}, y_{0:T_\ell}) | \theta^\ell\right\}_{\ell=0}^{N_{\text{src}}},$$

obtained over a time-span from zero to $T_\ell \in \mathbb{N}$ for some (not necessarily identical) initial state and control inputs $u_{0:T_\ell} := \{u_k\}_{k=0}^{T_\ell}$. Note that in practice, the target system data is collected with expensive instrumentation over a small time span, so $T^\star \ll \inf_\ell T_\ell$.

More formally, we propose *meta-learning* to leverage the source data $\mathcal{D}_{\text{src}}$ to learn NSSMs that can represent the different parameters and also adapt rapidly (i.e., with few online training updates) to the target system dynamics with unknown parameters $\theta^\star$ despite limited target data. This adaptive identification approach also has the added practical benefit that no explicit parameter estimation is required. Our hypothesis is that the rapid adaptation enabled by meta-learning will result in an accurate prediction model of the target system, which is expected to consequently improve the predictions required for updating state estimates via, for example, an extended Kalman filter, which exploits the differentiability of the meta-learned NSSM.

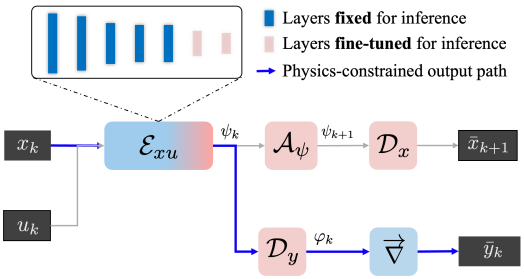

Fig. 1: Neural state-space model with mixed encoder and partial tuning at inference.

III. Physics-Constrained Meta-Learning for Deep Neural State-Space Models

A. Physics-Constrained NSSMs

As the base learner for our approach, we employ an autoencoder-like neural architecture, illustrated in Fig. 1. The key operations in this physics-constrained NSSM are:

$$\psi_k = \mathcal{E}_{xu}(x_k, u_k), \quad \psi_{k+1} = \mathcal{A}_\psi(\psi_k) \tag{2a}$$

$$\bar{x}_k = \mathcal{D}_x(\psi_k), \quad \varphi_k = \mathcal{D}_y(\psi_k), \quad \bar{y}_k = \vec{\nabla}\varphi_k, \tag{2b}$$

where $\psi_k \in \mathbb{R}^{n_\psi}$ denotes the latent encoding obtained using the encoder $\mathcal{E}_{xu} : \mathbb{R}^{n_x+n_u} \mapsto \mathbb{R}^{n_\psi}$. The latent state update ψ_{k+1} is determined by (2a) using the state transition operator $\mathcal{A}_\psi : \mathbb{R}^{n_\psi} \to \mathbb{R}^{n_\psi}$. The latent encoding ψ_k is mapped back to the original state-space by the decoder $\mathcal{D}_x : \mathbb{R}^{n_\psi} \to \mathbb{R}^{n_x}$. A separate decoding operator $\mathcal{D}_y : \mathbb{R}^{n_\psi} \mapsto \mathbb{R}$ maps the latent to a scalar φ_k, whose gradient $\vec{\nabla}\varphi_k$ yields the reconstructed output vector $\bar{y}_k$. Next, we discuss how the output path of the autoencoder implicitly enforces the curl-free constraint on the measurement function h.

The addition of physics-informed vector-field constraints into deep neural networks has been described more generally in [22], and we follow their method to enforce specific curl-free constraints into our NSSM, with some additional modifications to improve training performance. Clearly, such a constraint would have to be enforced on the neural network mapping from components of the state x to the output y. To this end, we invoke the following result [23, pp. 444].

Theorem 1. *Let h be a vector field of class $\mathcal{C}^1$ 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 fact implies that if we want to model a vector-field h that is curl-free using a neural network, a reasonable strategy is to learn a scalar function $\varphi(x)$, whose gradient $\vec{\nabla}\varphi$ will yield the desired curl-free vector field. Since modern deep-learning toolkits are equipped with capable automatic differentiation (AD) modules, computing y from φ is straightforward as long as the components of the state x corresponding to Cartesian coordinates $[p_X, p_Y, p_Z] \subset x$ are known. If so, computing $\vec{\nabla}\varphi$ merely involves executing an automatic differentiation with respect to these components. This motivates the design of the output path in our proposed physics-constrained NSSM—see Fig. 1. To reiterate, the estimate of the output function h, guaranteed by Theorem 1 to be curl-free, is

$$\bar{h}(x, u) = \vec{\nabla}\, \mathcal{D}_y \circ \mathcal{E}_{xu}(x, u). \tag{3}$$

Remark 1. In case the output is known to be solely a function of x, that is, $y = h_\theta(x)$, one could use split encoders $\mathcal{E}_x(x)$ and $\mathcal{E}_u(u)$ rather than a mixed encoder. With split encoders, $\bar{h}(x) = \nabla \mathcal{D}_y \circ \mathcal{E}_x(x)$. Consequently, the smooth activation functions will only be required for designing $\mathcal{E}_x$ and $\mathcal{D}_y$, allowing a more flexible choice of $\mathcal{E}_u$. ■

In order to represent non-constant h, one requires the output of the neural network to exist, and be non-constant. This motivates the use of activation functions that are smooth, with non-constant derivatives. As we shall see in Section III-C, we will use the Jacobian of the network for model-based state estimation, motivating not only first-order derivatives to be smooth, but second-order derivatives also. This is why we use the `swish` activation function [24],

$$\varsigma(z) = \frac{z}{1 + \exp(-\beta z)}, \tag{4}$$

which is non-monotonic, smooth, unbounded above, and bounded below; usually β is a trainable parameter, but we will proceed with $\beta = 1$ for simplicity. Being unbounded above avoids output saturation which minimizes near-zero gradients for large values similar to the ReLU function, and because of smoothness and non-monotonicity, the `swish` activation exhibits a soft self-gating property that does not impede the gradient flow, and therefore, stabilizes the training procedure [24].

We now describe how to train the neural SSM (2), a procedure that involves batch gradient-based optimization of weights for $\mathcal{E}_{xu}$, $\mathcal{A}_\psi$, $\mathcal{D}_x$, and $\mathcal{D}_y$. We use the differentiable training loss

$$\bar{\mathsf{L}}_{\text{NSSM}} = \bar{\mathsf{L}}_{\text{recon}} + \bar{\mathsf{L}}_{\text{pred},x} + \bar{\mathsf{L}}_{\text{pred},y}, \tag{5}$$

where $\bar{\mathsf{L}}_{\text{recon}} = \mathsf{MSE}_{0:N}(x_k\,, \mathcal{D}_x \circ \mathcal{E}_{xu}(x_k, u_k))$ denotes the reconstruction loss of the auto-encoder pair $(\mathcal{E}_{xu}, \mathcal{D}_x)$ over the state space x. The term $\bar{\mathsf{L}}_{\text{pred},x}$ in (5) denotes the one-step prediction error loss in the state vector, i.e.,

$$\bar{\mathsf{L}}_{\text{pred},x} = \mathsf{MSE}_{0:N-1}\big(x_{k+1}, \mathcal{D}_x \circ \mathcal{A}_\psi \circ \mathcal{E}_{xu}(x_k, u_k)\big),$$

and $\bar{\mathsf{L}}_{\text{pred},y}$ denotes the prediction error loss in outputs,

$$\bar{\mathsf{L}}_{\text{pred},y} = \mathsf{MSE}_{0:N}(y_k, \vec{\nabla}\, \mathcal{D}_y \circ \mathcal{E}_{xu}(x_k, u_k)). \tag{6}$$

Individual terms in the loss functions often need to be weighted by non-negative scalars to improve numerical conditioning or assign relative importance.

Remark 2. ReLU or LeakyReLU functions cannot be used as their second derivatives are zero. ■

B. Meta-Learning and Fine-Tuning in Online Adaptation

Owing to the inherent compute and hyperparameter tuning costs required in training neural networks, it is often preferable to avoid training models from scratch. However, because of their dependence on training data, model performance is extremely sensitive to small distribution shifts between the training and testing data. This has led to the popularity of so-called "fine-tuning" methods, where pre-trained models are updated on-the-fly using a small labeled dataset that better matches the test data distribution. However, updating all parameters in the model can lead to catastrophic forgetting and eliminate some of the benefits of initializing training with a pre-trained model. For this reason, fine-tuning only a subset of network parameters based on a-priori knowledge of the cause of the distribution shift, an approach known as "surgical fine-tuning" [25], can balance the competing objectives of accounting for distribution shift while maintaining properties of the pre-trained model. Herein, we fine-tune the later encoder and decoder layers of our NSSM as shown in Figure 1, as we expect distribution shifts between system identification tasks to result from changes in the observed data, that is, an output distribution shift in machine learning parlance.

If we know in advance that our model will be fine-tuned at inference time, we can also incorporate this knowledge into the training process using a meta-learning algorithm. In particular, we consider model-agnostic meta-learning [11] (MAML), which uses a nested training scheme with an inner-loop that fine-tunes a common set of initial model parameters on a small set of adaptation data, and an outer loop that updates the initial set of model parameters across multiple system identification tasks. During inference, we then only run the inner fine-tuning loop on a small target dataset. It is also possible to fine-tune only a subset of all trainable model parameters in the inner loop, a process referred to as the almost- no-inner-loop meta-learning algorithm [26]. However, this algorithm only fine-tunes the final layer in the inner loop, thus we use the more general "fine-tuning" (MAML-FT) terminology in this work.

The MAML-FT meta-training algorithm, summarized in Algorithm 1, operates as follows. For each source system dataset $\mathcal{D}^\ell_{\text{src}} \in \mathcal{D}_{\text{src}}$ we partition the first T_c samples in the trajectory into a context set $\mathcal{C}^\ell = (x^\ell_{0:T_c-1}, u_{0:T_c-1}, y^\ell_{0:T_c-1})$, and the remaining $T_\ell - T_c$ samples into a target set $\mathcal{T}^\ell = (x^\ell_{T_c:T_\ell}, u_{T_c:T_\ell}, y^\ell_{T_c:T_\ell})$. We denote by ω the set of all learnable model parameters from Figure 1, and $\omega_{\text{ft}} \subseteq \omega$ the set of model parameters that will be updated during inference

fine-tuning and inner loop meta-training (i.e., the parameters corresponding to the pink blocks in Figure 1). We then define the inner-loop updates for source task ℓ as

$$(\omega_{\mathsf{ft}})_m^\ell = (\omega_{\mathsf{ft}})_{m-1}^\ell - \beta_{\mathsf{in}} \nabla_{(\omega_{\mathsf{ft}})_{m-1}^\ell} \bar{\mathsf{L}}_{\mathrm{NSSM}}\big(C^\ell; (\omega_{\mathsf{ft}})_{m-1}^\ell\big), \quad (7)$$

where β_{in} is the inner-loop learning rate, m is the inner-loop update index, and $\bar{\mathsf{L}}_{\mathrm{NSSM}}\big(C^\ell; (\omega_{\mathsf{ft}})_{m-1}^\ell\big)$ is the loss function from (5) evaluated on the context set C^ℓ, i.e., the trajectory range $0 : T_c$, using the weights computed after $m-1$ inner-loop updates. From (7) we see that the inner-loop updates are performed individually for each source task, while the outer-loop update operates over a batch of tasks and for the entire set of trainable parameters ω, i.e.,

$$\omega = \omega - \beta_{\mathsf{out}} \nabla_\omega \sum_{b=1}^{B} \bar{\mathsf{L}}_{\mathrm{NSSM}}\big(\mathcal{T}^b; \omega_m^b\big) \quad (8)$$

where B is the number of tasks (i.e., trajectories) in a training mini-batch, β_{out} is the outer-loop learning rate, and $\bar{\mathsf{L}}_{\mathrm{NSSM}}\big(\mathcal{T}^b; \omega_m^b\big)$ is the loss function using the weights after completing m inner-loop iterations, and computed on the target set, i.e., the $T_\ell - T_c$ samples from each task that were not seen in the inner loop. Because the outer-loop updates parameters across B tasks, it encourages the learning of a parameter set ω that can be quickly adapted at inference time. The MAML-FT inference-time procedure performs M inner-loop updates (7) using the limited data from the target system $\mathcal{D}_{\mathrm{tgt}}$. Finally, model performance is evaluated using $\omega_M^\star$, the parameters fine-tuned in the inner loop.

Algorithm 1 OFFLINE META-TRAINING

Require: ω ▷ initial weights of NSSM
Require: $\mathcal{D}_{\mathrm{source}}$ ▷ source dataset
Require: $\omega_{\mathsf{ft}} \subseteq \omega$ ▷ weights to be fine-tuned
Require: $\beta_{\mathsf{in}}, \beta_{\mathsf{out}}, M$ ▷ learning rates and # iters
while not done **do** ▷ outer-loop
  Sample batch $\{\mathcal{D}_{\mathrm{src}}^b\}_{b=1}^B$ from $\mathcal{D}_{\mathrm{src}}$
  for $b = 1$ to B **do** ▷ inner-loop
    Partition $\mathcal{D}_{\mathrm{src}}^b$ into C^b and $\mathcal{T}^b$
    $\omega_0^b \leftarrow \omega$ ▷ copy current weights
    for $m = 1$ to M **do** ▷ adaptation steps
      for $\omega_l \in \omega_{\mathsf{ft}}$ **do** ▷ loop over parameters
        $(\omega_l)_m^b \leftarrow$ update using (7)
      end for
    end for
  end for
  $\omega \leftarrow$ update using (8)
end while
Return $\omega_\infty \leftarrow$ final trained weights

C. METAL-EKF: *Extended Kalman Filtering with Meta-Learned NSSMs*

After the NSSM (2) is adapted for the target system, we use it for state estimation. We adopt an EKF framework in this paper, but note that it can be readily extended to other nonlinear filtering approaches. We rewrite (2) in a form similar to (1) that is amenable to filtering,

$$x_{k+1} = \mathcal{D}_x \circ \mathcal{A}_\psi \circ \mathcal{E}_{xu}(x_k, u_k) + w_k, \quad (9a)$$
$$y_k = \vec{\nabla}\, \mathcal{D}_y \circ \mathcal{E}_{xu}(x_k, u_k) + \eta_k, \quad (9b)$$

where $\circ$ denotes function composition, and w_k and η_k denote the zero mean Gaussian uncertainties with respective covariance matrices Q_k^w and Q_k^η. For the EKF estimates, (x_k^-, P_k^-) and (x_k^+, P_k^+) denote the prior (measurements assimilated up to time step $k-1$) and posterior (measurements assimilated up to time step k) mean-covariance pairs of the state vector at time step k, respectively, i.e.,

$$x_{k|k-1} \sim \mathcal{N}(x_k^-, P_k^-) \text{ and } x_{k|k} \sim \mathcal{N}(x_k^+, P_k^+), \quad (10)$$

where $\mathcal{N}(\cdot)$ denotes the multivariate normal distribution.

For a given (x_0^-, P_0^-) and neural predictive model (9), the EKF equations are given by the measurement update:

$$H_k = \frac{\partial}{\partial x} \vec{\nabla}\, \mathcal{D}_y \circ \mathcal{E}_{xu}(x_k^-, u_k), \quad (11a)$$
$$K_k = (P_k^- H_k^\top)(H_k P_k^- H_k^\top + Q_k^\eta)^{-1} \quad (11b)$$
$$P_k^+ = (I - K_k H_k) P_k^- \quad (11c)$$
$$x_k^+ = x_k^- + K_k\big(y_k - \vec{\nabla}\, \mathcal{D}_y \circ \mathcal{E}_{xu}(x_k^-, u_k)\big) \quad (11d)$$

and the time update:

$$F_k = \frac{\partial}{\partial x} \mathcal{D}_x \circ \mathcal{A}_\psi \circ \mathcal{E}_{xu}(x_k^+, u_k), \quad (12a)$$
$$P_{k+1}^- = F_k P_k^+ F_k^\top + Q_k^w \quad (12b)$$
$$x_{k+1}^- = \mathcal{D}_x \circ \mathcal{A}_\psi \circ \mathcal{E}_{xu}(x_k^+, u_k). \quad (12c)$$

Note that computing the Jacobian matrices H_k and F_k in (11a) and (12a) requires differentiation of the outputs of the deep neural networks with respect to their inputs. To this end, we utilize AD capabilities available in standard deep-learning libraries, such as `PyTorch`.

IV. CASE STUDY: POSITIONING UNDER UNCERTAIN MOTION MODELS AND UNKNOWN MAGNETIC FIELDS

In this section, we evaluate our proposed methodology to magnetic-field localization using simulated data. First, we go through the modeling aspects of the application. This is followed by implementation aspects, and we conclude with numerical results and accompanying discussion. The scenario we consider is the example used in [21], tailored to the recursive setting similar to [17]. Specifically, the magnetic-field model is from [21], for which we have ground truth and can accurately evaluate our proposed approach.

Consider a sphere centered at the origin with a radius of r_0m having a uniform magnetization of $\mathcal{M} = [m_X, m_Y, 0]^\top$A/m; see Fig. 2(A). Then, the magnetic-field function h_θ mapping the position $p := [p_X, p_Y]$ to the magnetic field y is described by

$$h_\theta = \begin{cases} -\mathcal{M}/3 & \text{if } r < r_0 \\ \frac{m_0}{4\pi}\left(\mathcal{M}/r^3 + 3/r^5(\mathcal{M}^\top p)p\right) & \text{if } r \geq r_0 \end{cases} \quad (13)$$

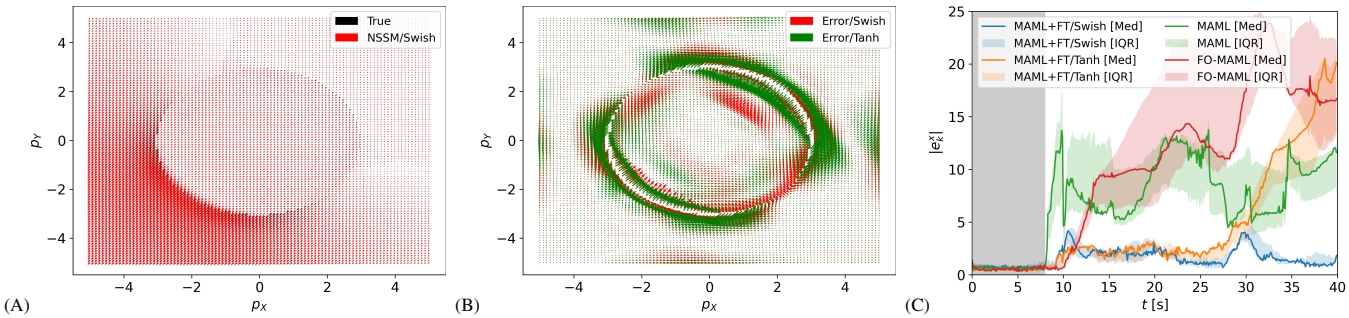

Fig. 2: (A) Illustration of actual and NSSM-learned magnetic field function h_θ. (B) Comparison of errors between swish and tanh NSSMs. Note that only around a small neighborhood of the region of discontinuity at $r_0 = 3$m do we observe significant discrepancy between the two. (C) Comparison of state estimation error norm (median, interquartile range) for different meta-learned NSSM models. Gray shade indicates time span used for adaptation.

where $m_0 = 4/3\pi r_0^3$ and $r = \|p\|$. With zero-mean Gaussian noise $\eta_k \sim \mathcal{N}(0, Q_k^\eta)$ with covariance Q_k^η on the measurements, we write the measurement equation as

$$y_k = h_\theta(x_k) + \eta_k. \tag{14}$$

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

$$\dot{x} = \begin{bmatrix} v_X \cos(\psi + \beta)/\cos(\beta) \\ v^X \sin(\psi + \beta)/\cos(\beta) \\ v^X \tan(\delta)/L \end{bmatrix}, \tag{15}$$

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. After time discretization, we write (15) concisely as

$$x_{k+1} = f_\theta(x_k, u_k) + w_k, \tag{16}$$

with Gaussian zero-mean process noise, $w_k \sim \mathcal{N}(0, Q_k^w)$, to account for model mismatch, $u_k = [\delta, \omega_f, \omega_r]$ being control action. The steering controller is a pure-pursuit controller tracking a reference path $p_r(t)$. In the context of (1), the unknown parameters θ comprise the wheelbase parameters l_f, l_r, the maximum steering angle δ: these affect the vehicle dynamics; and the magnetization m_X, m_Y, the magnetic-field radius r_0: these affect the magnetic vector field.

Data generation: In order to construct a meta-learning dataset, we need to generate source and target systems. To this end, we assume that the range of $l_f, l_r \in [0.02, 0.25]^2$, $m_X, m_Y \in [-1.5, 1.5]^2$, the maximum steering angle $\delta \in [10, 20]$, and $r_0 \in [1.5, 4.5]$. Then we construct 80 source and 20 target systems by randomly sampling parameters from these ranges, and executing simulations that yield x, u, and y data, for each source and target system. The simulations last for 40s, with a sampling period of 0.1s and identical references to be tracked by the vehicle. Despite the common reference to be tracked, we have observed (we do not show a plot due to limited space) significant variation within $\mathcal{D}_{\text{src}}$ and $\mathcal{D}_{\text{tgt}}$ due to variation in θ.

NSSM architecture: After a preliminary neural architecture search, the dimension of the latent space was selected to be $n_\psi = 128$. Each of the components in Fig. 1 was implemented with fully connected layers activated by `swish` functions. The graph of the encoder and two decoders were selected to be $\mathcal{E}_{xu} : 6-256-128-128-64-32-32-128$, $\mathcal{D}_x : 128-32-64-128-256-3$ with 3 updated states as outputs, and $\mathcal{D}_y : 128-64-64-128-128-256-256-1$, since the output of this is a scalar potential. The state transition operator $\mathcal{A}_\psi$ was implemented network with six fully-connected layers of identical hidden dimension of 256. For meta-inference, the last 2 layers of $\mathcal{E}_{xu}$ and all the layers of $\mathcal{A}_\psi$, $\mathcal{D}_x$ and $\mathcal{D}_y$ were adapted with target system data. The first few layers of $\mathcal{E}_{xu}$ were fixed after meta-training. We reiterate that $\vec{\nabla}$ in Fig. 1 is not a trainable neural network, it merely denotes the vector differential operator that differentiates the output of $\mathcal{D}_y$ with respect to p.

Results and Discussion: The meta-learned NSSM was trained offline on the source dataset for 30000 epochs using the Adam optimizer with fixed learning rates $\beta_{\text{out}} = 10^{-4}$, $\beta_{\text{in}} = 10^{-3}$, and $M = 10$ adaptation iterations; an NVIDIA 3090X GPU was used for training, and the total training time was at most 22 hours. The weights of the NSSM were saved when the validation loss decreased. For meta-inference, we use 20% of the target system data, which is the initial $T_c = 8$s out of the total simulation time of $T_\ell = 40$s. The performance of the NSSM was evaluated in terms of prediction and estimation errors for the remaining 80% of the target system's trajectory never seen by the NSSM.

In Fig. 2(A) we illustrate the data obtained from the actual magnetic field h_θ, obtained by evaluating h_θ in (13) parameterized with $m_X = 1$, $m_Y = 1$, $r_0 = 3$m using black arrows. Most of this data is overlayed by the estimate of h_θ learned by the `swish`-activated NSSM, evaluated over a regular 100×100 grid in a two-dimensional spatial domain; 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. This is corroborated by Fig. 2(B), which shows the vector plot of errors in the magnetic field estimate, i.e. the error between the actual, and `swish`- and `tanh`-activated NSSMs. The `tanh`-activated NSSM (green) induces larger approximation errors than the `swish`-activated NSSM (red). This is established 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.

Fig. 2(C) compares of estimation accuracy of MetaL-EKF implemented with different meta-learned NSSMs. Statistical metrics of performance, i.e., median and interquartile range of the state estimation error norm were calculated over 20 target systems. In particular, we consider four variants of meta-learning, two of which are the most popular meta-learning algorithms. The first (blue) is the proposed approach with the proposed `swish` activation. As described in § III-B, this is the MAML-FT approach that fine tunes a subnetwork; the green line in the subplot corresponds to MAML, where the entire network is fine-tuned, not a subnetwork. The orange line shows the performance of MAML+FT with the `tanh` activation. The final comparison is with first-order (FO) MAML (red line), which is an efficient approximation of the MAML algorithm that does not backpropagate through the inner-loop in Algorithm 1 during meta-training, and has been shown to sometimes perform as well as MAML [11], [27]. Different runs of MetaL-EKFs using different NSSMs were subject to identical reference trajectories, identical measurements, and identical initialization for a fixed target system. The parameters of NSSMs were adapted using the target system's trajectory data of initial 8s, and the adapted network was used for state estimation as the predictive model in MetaL-EKF. In Fig. 2(C), the first 8s of data has been used for adaptation by all the networks: this is why the state estimation error norm $\|e_k^x\|$ is small in this shaded area. After 8s, as expected, we see signficantly different performance amongst the different filters. The comparison of MAML+FT/Swish and MAML+FT/Tanh indicates that the `swish` activation may be more suitable for modeling magnetic fields in positioning applications as the MAML+FT/Tanh version exhibited significantly larger estimation errors, especially in later parts of the trajectories. This is potentially attributed to the fact that the unboundedness of the `swish` function leads to improved gradient flow in this non-trivially deep network, which leads to improved meta-training. The importance of fine tuning, i.e., adapting only a carefully selected subset of network parameters, is highlighted by the significantly smaller estimation error demonstrated by MAML+FT/Swish compared to classical MAML. Furthermore, FO-MAML is outperformed by all other MAML variants, potentially because the non-constant second-derivatives in the swish activation are beneficial when backpropagating through the nested training scheme in Algorithm 1.

REFERENCES

[1] M. Forgione, M. Mejari, and D. Piga, "Learning neural state-space models: do we need a state estimator?" *arXiv preprint arXiv:2206.12928*, 2022.

[2] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, p. 109666, 2021.

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

[4] G. I. Beintema, M. Schoukens, and R. Tóth, "Deep subspace encoders for nonlinear system identification," *Automatica*, vol. 156, p. 111210, 2023.

[5] C. Legaard, T. Schranz, G. Schweiger *et al.*, "Constructing neural network based models for simulating dynamical systems," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–34, 2023.

[6] T. X. Nghiem, J. Drgoňa, C. Jones *et al.*, "Physics-informed machine learning for modeling and control of dynamical systems," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3735–3750.

[7] 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*, vol. 7, pp. 3483–3488, 2023.

[8] 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.

[9] M. Forgione, F. Pura, and D. Piga, "From system models to class models: An in-context learning paradigm," *IEEE Control Systems Letters*, vol. 7, pp. 3513–3518, 2023.

[10] A. Chakrabarty, "Optimizing closed-loop performance with data from similar systems: A bayesian meta-learning approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 130–136.

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

[12] S. Zhan, G. Wichern, C. Laughman *et al.*, "Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization," *Energy and Buildings*, vol. 270, p. 112278, 2022.

[13] S. M. Richards, N. Azizan, J.-J. Slotine *et al.*, "Adaptive-control-oriented meta-learning for nonlinear systems," *arXiv preprint arXiv:2103.04490*, 2021.

[14] E. Arcari, A. Carron, and M. N. Zeilinger, "Meta learning MPC using finite-dimensional Gaussian process approximations," *arXiv preprint arXiv:2008.05984*, 2020.

[15] D. Muthirayan and P. P. Khargonekar, "Meta-learning guarantees for online receding horizon learning control," *arXiv preprint arXiv:2010.11327*, 2020.

[16] Y. Gu, S. Cheng, and N. Hovakimyan, "Proto-MPC: An encoder-prototype-decoder approach for quadrotor control in challenging winds," *arXiv preprint arXiv:2401.15508*, 2024.

[17] K. Berntorp and 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.

[18] 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.

[19] 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.

[20] A. Chakrabarty, A. P. Vinod, H. Mansour *et al.*, "Moving horizon estimation for digital twins using deep autoencoders," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5500–5505, 2023.

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

[22] J. Hendriks, C. Jidling, A. Wills *et al.*, "Linearly constrained neural networks," *arXiv preprint arXiv:2002.01600*, 2020.

[23] S. J. Colley, *Vector Calculus*, 4th ed. Upper Saddle River, NJ: Pearson, Sep. 2011.

[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[25] Y. Lee, A. S. Chen, F. Tajwar *et al.*, "Surgical fine-tuning improves adaptation to distribution shifts," *arXiv preprint arXiv:2210.11466*, 2022.

[26] 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.

[27] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.