

Meta-Learning for Rapid Adaptation in Reference Tracking of Uncertain Nonlinear Systems

Yan, Jiaqi; Chakrabarty, Ankush; Schmid, Niklas; Lygeros, John; Rupenyan, Alisa

TR2026-159 October 29, 2026

Abstract

In this paper, we address the problem of reference tracking for uncertain nonlinear systems. Since collecting data from the target system (i.e., the system of interest) is often challenging, our objective is to design optimal controllers using only limited target system data. Meta-learning provides a promising paradigm by leveraging offline data from source systems (systems sharing structural similarities with the target system) to accelerate training and enhance control performance. Motivated by this idea, we propose a meta-learning-based control framework that tailors the implicit model-agnostic meta-learning (iMAML) algorithm to the control setting. The framework operates in two phases: an (offline) meta-training phase, where an aggregated representation is learned from source data to capture the shared system dynamics among similar systems, and an (online) meta-adaptation phase, where this representation is fine-tuned on the target system using only a few data samples and limited adaptation steps. We formulate this framework as a bi-level optimization problem and provide an efficient solution with reduced storage complexity and few approximations. The developed framework is general, allowing various learning algorithms to be integrated. To demonstrate this flexibility, we further propose two specific learning algorithms to be incorporated into our framework based on a neural state-space model and a deep Q-network, respectively. The primary distinction between these approaches is whether explicit system identification is required. Numerical simulations and hardware experiments demonstrate that the proposed methods enhance control performance and consistently outperform baseline approaches.

Transactions on Automation Science and Engineering (T-ASE) 2026

© 2026 MERL. This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Meta-Learning for Rapid Adaptation in Reference Tracking of Uncertain Nonlinear Systems

Jiaqi Yan, Ankush Chakrabarty, Niklas Schmid, John Lygeros, and Alisa Rupenyan

Abstract—In this paper, we address the problem of reference tracking for uncertain nonlinear systems. Since collecting data from the target system (i.e., the system of interest) is often challenging, our objective is to design optimal controllers using only limited target system data. Meta-learning provides a promising paradigm by leveraging offline data from source systems (systems sharing structural similarities with the target system) to accelerate training and enhance control performance. Motivated by this idea, we propose a meta-learning-based control framework that tailors the implicit model-agnostic meta-learning (iMAML) algorithm to the control setting. The framework operates in two phases: an (offline) meta-training phase, where an aggregated representation is learned from source data to capture the shared system dynamics among similar systems, and an (online) meta-adaptation phase, where this representation is fine-tuned on the target system using only a few data samples and limited adaptation steps. We formulate this framework as a bi-level optimization problem and provide an efficient solution with reduced storage complexity and few approximations. The developed framework is general, allowing various learning algorithms to be integrated. To demonstrate this flexibility, we further propose two specific learning algorithms to be incorporated into our framework based on a neural state-space model and a deep Q-network, respectively. The primary distinction between these approaches is whether explicit system identification is required. Numerical simulations and hardware experiments demonstrate that the proposed methods enhance control performance and consistently outperform baseline approaches.

Note to Practitioners—This work is motivated by the fact that collecting sufficient data from physical systems for control design is often costly and time-consuming. To address this issue, we propose a meta-learning-based control framework that enhances adaptation speed and control performance when only limited target system data are available. The proposed approach leverages historical data from similar systems to pre-train a model offline, which can then be efficiently adapted to new systems online with minimal data and computation. Although we demonstrate the framework using two specific algorithms, it is general and readily extendable to various learning-based control methods. Practitioners can apply this framework to improve control performance and data efficiency in real-world applications such as autonomous driving, robotic manipulation, and process control, where conventional data-driven methods typically require extensive online training. Experimental results

show that our methods outperform meta-learning and data-driven baselines in simulation, and significantly reduce the real-world data required on hardware. These results also highlight the effectiveness of the proposed approach in bridging the sim-to-real gap. Future work will focus on extending this framework to multi-agent systems and enabling real-time adaptation in highly dynamic environments.

Index Terms—Meta-learning, optimal control, data efficiency, data-driven control.

I. INTRODUCTION

Control systems that can easily adapt to new scenarios and changing dynamics are crucial for enabling flexible automation in various application fields, in particular manufacturing and robotics. Learning-based approaches to adaptation, such as reinforcement learning, adaptive dynamic programming, and stochastic optimization-based control [1]–[3] offer a promising approach to this problem. Meta-learning [4] can complement these approaches when there is limited access to data from the system to be controlled (*target system*). It combines data from *source systems* that share structural similarities with the target system. Thus, data from numerical models, digital twins, or similar physical systems can be used to pretrain a model/policy before deployment on the target system. The model is then adapted with limited evaluations from the target system. Therefore, meta-learning comprises two phases: a meta-training phase that pretrains a model to capture similarities among source systems, and a meta-adaptation phase that fine-tunes the pretrained model on the target system. When the target and the source systems are similar, it is also possible to use transfer learning to pass knowledge from different source systems. Transfer learning typically requires sufficient target system data to fine-tune a pre-trained model, and the transferred knowledge often comes from a single source task. On the other hand, multi-task learning assumes all tasks are available during training and optimizes for joint performance rather than rapid adaptation [5], [6]. In contrast, meta-learning optimizes for rapid adaptation to new tasks with minimal data, which is reflected in the bi-level optimization structure of meta-learning: the inner loop simulates adaptation on individual tasks, while the outer loop optimizes for fast adaptation capability itself. This structure is absent in standard transfer or multi-task learning formulations.

The meta-learning approach is particularly suitable for nonlinear systems with uncertainties, as it eliminates the need for a complete re-design of the controller. Many meta-learning-

J. Yan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, P. R. China (jqyan@buaa.edu.cn).

A. Chakrabarty is with the Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA (achakrabarty@ieee.org).

N. Schmid and J. Lygeros are with the Automatic Control Laboratory, ETH Zurich, Switzerland (nikschmid@ethz.ch, jlygeros@ethz.ch).

A. Rupenyan is with the ZHAW Centre for Artificial Intelligence, Zurich University of Applied Sciences, Switzerland (alisa.rupenyan@zhaw.ch).

This work is supported by the Swiss National Science Foundation through NCCR Automation under Grant agreement 51NF40_180545.

based control algorithms rely on the Model Agnostic Meta Learning (MAML) [7] to enable rapid adaptation. Recent works have extended these methodologies to various control paradigms. For example, a policy gradient MAML approach was developed in [8] for model-free linear quadratic regulator (LQR) design, while meta-reinforcement learning was successfully utilized in [9] to minimize domain discrepancies across the sim-to-real gap in robotic tasks. Furthermore, MAML has been applied to synthesize Lyapunov functions [10], optimize Kalman gains [11] in changing environments, and to find robust initializations for learning Model Predictive Control (MPC) with uncertain systems and objectives [12]. In the bi-level optimization problem involved with MAML, the inner problem calculates and propagates derivatives of the training loss function along the full optimization path (the length of which is the same as the number of inner-loop updates/steps). The outer problem then accumulates information from these inner-loop updates and computes an outer-loop gradient direction. By repeating these two steps, the MAML learner is expected to asymptotically converge to a pre-trained controller from which rapid adaptation is possible on the target system. Choosing a large number of inner-loop steps, while leading to better outer-loop directions, incurs high memory complexity, and is not practical for a large number of inner-loop adaptation steps. Existing approaches propose various approximations to obtain efficient outer loop estimates, or cast the problem in mathematical frameworks that reduce training time. For example, in [7], a first-order approximation is used for specific rectified linear unit (ReLU) activation functions. However, such approximations sometimes oversimplify the computations and degrade learning performance.

To overcome this limitation, the implicit MAML (iMAML) algorithm leverages the implicit function theorem to achieve efficient outer-loop gradient computation with fewer approximations [13]. This approach solves the inner-loop optimization over many iterations without incurring excessive memory overhead, as it is agnostic to the optimization path and depends only on the final gradient direction. Moreover, minimizing such approximations can enhance adaptation performance. The problem of reference tracking in uncertain nonlinear systems making use of iMAML was discussed in [14]. The gradients were computed exactly during training, and achieved good predictive performance for model predictive control (MPC) in different (related) systems.

Building upon the approach in [14], this paper propose a general meta-learning-based control framework. This framework is algorithm-agnostic, accommodating *any* learning-based control method formulated with a differentiable loss function. Specifically, it unifies:

- *Indirect data-driven methods*, where a model is first identified by data and then used for control design.
- *Direct data-driven methods*, where the control policy is learned directly from data.

Fig. 1 illustrates the distinction between these two paradigms. While [14] focused exclusively on an indirect implementation using neural state-space models (NSSMs) for system identification, our framework broadens this scope, effectively

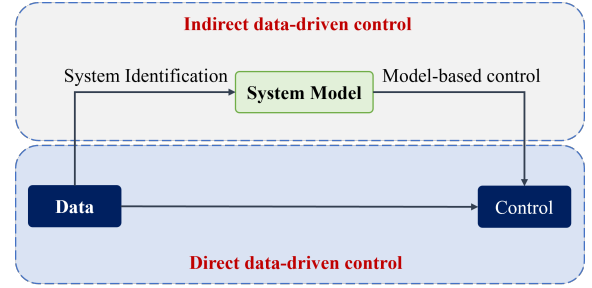


Fig. 1: Direct vs. indirect data-driven control methods.

rendering [14] a special case of this approach.

The main contributions of this paper are:

- 1) While meta-learning has been shown to be effective for both model adaptation [10], [15]–[17] and control policy adaptation [8], [9], [11], [18]–[20], existing implementations are typically applied on a case-by-case basis. In contrast, our framework is general and flexible, capable of incorporating various learning algorithms. Unlike existing implementations that are algorithm-specific, our framework provides a unified mathematical structure based on bi-level optimization that remains invariant across different learning algorithms—only the specific form of the loss function changes to reflect the particular control approach employed.
- 2) To demonstrate the flexibility of our framework, we propose two representative algorithms: an *indirect data-driven control* method based on a neural state-space model (NSSM) and MPC, and a *direct data-driven control* method based on a Deep Q-Network (DQN). The key difference lies in whether explicit system identification is required, which is reflected in the corresponding loss function design. We show that both methods enable efficient adaptation to unseen target tasks while maintaining good tracking performance under system uncertainties.
- 3) Under standard assumptions, we establish convergence guarantees for the proposed framework and provide stability analysis for a terminal-constraint variant of the resulting MPC controller. Formal proofs are provided in Appendix A.

Numerical simulations demonstrate that our iMAML-based method outperforms multiple baselines, including both MAML-trained models and models trained purely on target system data. Moreover, hardware experiments on the ball-on-a-plate system are particularly designed to evaluate the direct meta-learning approach. Experimental results verify that by pre-training the controller in simulation, the proposed algorithm enables faster adaptation to the real system while requiring significantly less real-world data compared to training solely on physical system data. This approach effectively reduces the sim-to-real gap in control system implementation.

The remainder of this paper is organized as follows: Section II formulates the problem and introduces the datasets. Section III presents the general meta-learning framework and its solution. Section IV develops the indirect and direct data-driven algorithms, respectively, that can be integrated into

the proposed framework. Finally, numerical simulations and hardware experiments are given in Section V to validate the proposed algorithms, and Section VI concludes the paper.

II. PROBLEM FORMULATION

We consider a family of parameterized discrete-time non-linear systems of the form

$$\begin{aligned} x_{t+1} &= f(x_t, u_{t+1}, \theta_f), \\ y_t &= g(x_t, \theta_g), \end{aligned} \quad (1)$$

where $x \in \mathbb{R}^n$ and $y \in \mathcal{Y} \subseteq \mathbb{R}^m$ represent the system state and (constrained) output, and $u \in \mathcal{U} \subseteq \mathbb{R}^p$ is the system input with $\mathcal{U}$ and $\mathcal{Y}$ being compact, convex sets. Both f and g are nonlinear functions, and $\theta := [\theta_f; \theta_g] \in \mathbb{R}^w$ denotes a vector of unknown parameters.

The target system is described by (1) with parameters θ^0 . The true value of θ^0 is unknown but is drawn from a distribution $\theta^0 \sim \Theta$, where Θ is often informed by domain knowledge.

Let $U(\theta^0, T^0)$ and $Y(\theta^0, T^0)$ represent the input and output trajectories generated from the target system over a horizon T^0 . We construct the target dataset as

$$\mathcal{D}_{\text{target}} := \{U(\theta^0, T^0), Y(\theta^0, T^0)\}. \quad (2)$$

Despite the uncertain dynamics, our objective is to design a controller to asymptotically track some reference signal $\bar{y}$:

$$\lim_{t \rightarrow \infty} y_t = \bar{y}_t. \quad (3)$$

In many applications, collecting data from the target system is expensive. Therefore, $\mathcal{D}_{\text{target}}$ is typically of limited size. Training with such limited data can lead to poor control performance. To overcome this challenge, we propose a meta-learning-based control framework. The key idea is to learn an *aggregated representation*—typically offline—using data from multiple source systems that share structural similarities with the target system. The aggregated representation captures structural similarities among the source systems and serves as an informative initialization for rapid adaptation to a target system. From an engineering perspective, this design is driven by practical challenges in automation and robotics: collecting extensive data directly from a physical target system is often prohibitively expensive or unsafe. In contrast, offline data from simulations, digital twins, or other source systems is typically abundant. By leveraging this accessible source data to learn a shared representation, our framework significantly improves data efficiency during real-world deployment.

Also note that the effectiveness of the aggregated representation depends on the similarity between source and target systems. If the target dynamics differ significantly from the source systems, the adaptation performance may deteriorate. As such, representative source-system selection is important for successful deployment. Accordingly, we assume that each source system follows the dynamics in (1), where the k -th system is parameterized by θ^k with $\theta^k \sim \Theta$. It is important to note that the values of θ^k need not be known.

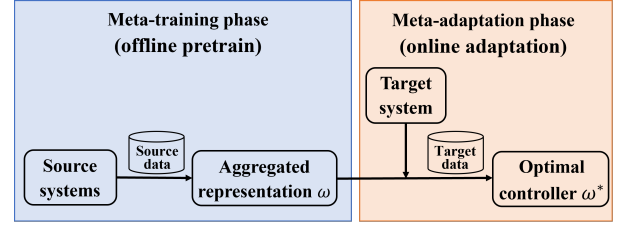


Fig. 2: The two phases in the proposed meta-learning framework.

Assume that we have access to a dataset consisting of input–output trajectories collected from N_s different source systems. We denote the resulting source dataset as

$$\begin{aligned} \mathcal{D}_{\text{source}} &= \{(U(\theta^k, T^k), Y(\theta^k, T^k))\}_{k=1}^{N_s} \\ &:= \{\mathcal{D}^k\}_{k=1}^{N_s}, \end{aligned} \quad (4)$$

where T^k is the length of trajectory collected from the k -th system. Note that when the source dataset contains only a single system ($N_s = 1$), our framework reduces to standard transfer learning. Meta-learning generalizes this by learning from multiple source systems and optimizing for adaptation efficiency through a bi-level optimization.

III. A META-LEARNING FRAMEWORK

The proposed meta-learning framework is illustrated in Fig. 2. The framework comprises two phases: an *offline* meta-training phase and an *online* meta-adaptation phase. During the meta-training phase, an aggregated representation is learned offline to capture similarities among the source systems. In the meta-adaptation phase, this representation is quickly fine-tuned for the target system with limited target data and few online adaptation steps. In this section, we do not restrict the control objective to any specific loss function. Instead, the framework accommodates any learning-based control approach formulated via a differentiable loss function $\hat{\ell}(\cdot)$. This includes both indirect methods (e.g., model-based approaches such as neural state-space models combined with MPC) and direct methods (e.g., reinforcement learning policies such as DQN, actor-critic methods, or policy gradient algorithms). The only requirement is that the inner and outer loss functions be differentiable with respect to their parameters, thereby enabling gradient-based optimization.

In the sequel, we show how to effectively obtain the aggregated representation in the meta-training phase. Note that only source data is used in this stage. Later in Section IV, we will design specific loss functions and incorporate target data to achieve the tracking objective (3).

A. Bi-level optimization problem in meta-training

As shown in Fig. 3, the meta-training phase has two loops. In each outer-loop iteration, we draw a batch of trajectories $\{\mathcal{D}^b\}_{b=1}^B$ from the source dataset $\mathcal{D}_{\text{source}}$, corresponding to B different source systems. A *meta-learner* updates the aggregated representation ω by evaluating performance across these source systems. In the inner loop, a *base-learner* adapts

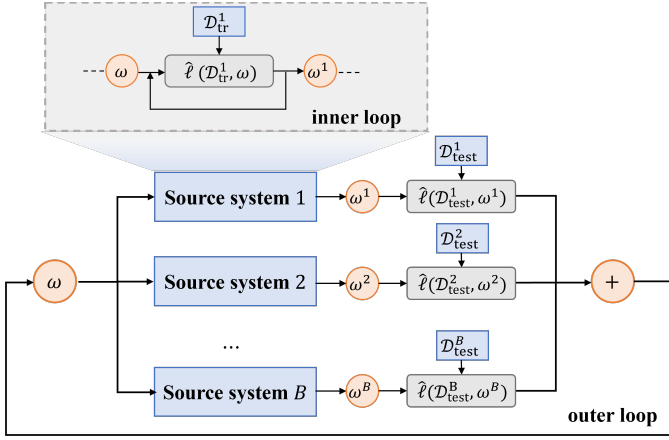


Fig. 3: Information flow in the meta-training phase.

the aggregated representation ω , generating system-specific parameters ω^b using each source data $\mathcal{D}^b$. Next, we describe the procedure in detail.

Inner problem: The base-learner in the inner problem focuses on adapting the aggregated representation ω to each individual source system. We partition the dataset associated with the b -th source system, denoted by $\mathcal{D}^b$, into a training set $\mathcal{D}_{\text{tr}}^b$ and a testing set $\mathcal{D}_{\text{test}}^b$. The training set is used to fine-tune the aggregated parameters ω into system-specific parameters $\tilde{\omega}^b$ on each source system by solving the following optimization problem:

$$\text{Inner Problem: } \tilde{\omega}^b(\omega) := \underset{\psi}{\operatorname{argmin}} \ell(\mathcal{D}_{\text{tr}}^b; \psi) + \frac{\gamma}{2} \|\psi - \omega\|^2. \quad (5)$$

Note the dependence of $\tilde{\omega}^b(\omega)$ on the aggregated representation ω . In this formulation, $\ell(\cdot)$ denotes the inner-problem loss function that quantifies the adaptation performance for the b -th source system. Again, we do not assume any specific form for $\ell(\cdot)$ at this point. The regularization term $\|\psi - \omega\|^2$ encourages the adapted parameters to remain close to the aggregated representation, with the regularization strength governed by $\gamma > 0$. It determines the size of the trust region during inner-loop adaptation. A larger γ limits the adapted parameters from deviating too far from the aggregated representation, which is crucial when source tasks are highly diverse (preventing overfitting to a single task). Conversely, a smaller value is preferred when the target environment demands aggressive policy shifts. As will be shown in Section III-B, this regularization plays a critical role by enabling a closed-form expression for the outer-problem gradient via the Implicit Function Theorem.

Outer problem: The testing datasets from multiple source systems, denoted by $\{\mathcal{D}_{\text{test}}^b\}_{b=1}^B$, are utilized in the outer problem to update the aggregated representation ω . This update is performed by the meta-learner that assesses the adapted performance across all source systems, as defined by the following objective:

$$\text{Outer Problem: } \min_{\omega} L(\omega) := \sum_{b=1}^B \ell(\mathcal{D}_{\text{test}}^b; \tilde{\omega}^b(\omega)). \quad (6)$$

The meta-learner aims to learn a set of parameters ω that can

produce good task-specific parameters in different systems.

Combining (5) and (6), the meta-training phase is formulated as a bi-level optimization problem. As shown in Fig. 3, these two problems are optimized alternately until convergence is achieved.

B. Solution to the bi-level optimization problem

The inner optimization problem (5) can be solved using iterative methods such as gradient descent:

$$\omega^b(\omega) \leftarrow \omega^b(\omega) - \beta_{\text{in}} (\nabla_{\psi} \ell(\mathcal{D}_{\text{tr}}^b; \psi)|_{\psi=\omega^b(\omega)} + \gamma(\omega^b(\omega) - \omega)), \quad (7)$$

where β_{in} denotes the inner-loop learning rate. The same is true for (6):

$$\omega \leftarrow \omega - \beta_{\text{out}} \sum_{b=1}^B \nabla_{\omega} \ell(\mathcal{D}_{\text{test}}^b; \omega^b(\omega)), \quad (8)$$

where β_{out} is the outer-loop learning rate. The inner optimization is assumed to approach a local optimum within a limited number of adaptation steps. As a result, the inner-loop learning rate β_{in} should be chosen to balance fast convergence and optimization stability. A common choice is 10^{-4} – 10^{-3} . On the other hand, the outer-loop learning rate β_{out} , which updates the shared initialization, is set to be one order of magnitude larger than β_{in} and is optimized with Adam to ensure stable meta-training across diverse tasks. In the equations above, the tilde notation $\tilde{(\cdot)}$ denotes the exact optimizer, whereas the unmodified variables represent the solutions obtained via iterative approximation. Note that the dependence of ω^b on ω complicates the gradient computation in (8). By applying the chain rule, we expand each gradient term as

$$\nabla_{\omega} \ell(\mathcal{D}_{\text{test}}^b; \omega^b(\omega)) = \frac{d\omega^b(\omega)}{d\omega} \cdot P^b \quad (9)$$

with

$$P^b := \nabla_{\psi} \ell(\mathcal{D}_{\text{test}}^b; \psi)|_{\psi=\omega^b(\omega)}. \quad (10)$$

It is noted that P^b can be efficiently computed via automatic differentiation [13]. However, computing $\frac{d\omega^b(\omega)}{d\omega}$ is non-trivial since $\omega^b(\omega)$ is defined via an optimization process (5). One possible approach is to backpropagate through the iteration (7), but this requires storing the full optimization trajectory, which can be memory-intensive or even infeasible for many steps or non-differentiable solvers.

To address this challenge, the following lemma is introduced to efficiently (in both computation and memory) compute (9):

Lemma 1 (Implicit Function Lemma [13]). *Define*

$$Q^b := I + \frac{1}{\gamma} \nabla_{\psi}^2 \ell(\mathcal{D}_{\text{tr}}^b; \psi)|_{\psi=\omega^b(\omega)}. \quad (11)$$

If Q^b is invertible, then

$$\frac{d\omega^b(\omega)}{d\omega} = (Q^b)^{-1}. \quad (12)$$

Combining (9) and (12), we obtain

$$\nabla_{\omega} \ell(\mathcal{D}_{\text{test}}^b; \omega^b(\omega)) = (Q^b)^{-1} P^b. \quad (13)$$

Since Q^b depends only on $\omega^b(\omega)$, Lemma 1 allows efficient computation of $\frac{d\omega^b(\omega)}{d\omega}$ without storing the entire optimization path. However, obtaining the exact solution $\tilde{\omega}^b(\omega)$ of the inner problem (5) requires full convergence, which is often impractical. Therefore, we approximate $\tilde{\omega}^b(\omega)$ by using a few steps of gradient descent (7).

In practice, explicitly computing the inverse of Q^b can also be costly, especially in large-scale systems. To avoid matrix inversion, one can directly find $(Q^b)^{-1}P^b$ by solving the following optimization problem:

$$g^b := \underset{\phi}{\operatorname{argmin}} \phi^T Q^b \phi - \phi^T P^b. \quad (14)$$

Thus, $(Q^b)^{-1}P^b$ can be efficiently approximated by finding g^b using an iterative method. To enforce the capability of “few-shot” adaptation, the number of iterations is generally restricted to a small integer.

Remark 1. Besides iMAML used here, another widely used meta-learning algorithm is MAML [7]. Instead of (5), the inner loop of MAML solves

$$\omega^b(\omega) := \underset{\psi}{\operatorname{argmin}} \widehat{\ell}(\mathcal{D}_{tr}^b; \psi). \quad (15)$$

This formulation omits the regularization term in (5), which makes Lemma 1 inapplicable. As a result, the inner loop must compute and propagate derivatives of the training loss along the full optimization trajectory. To mitigate the high complexity, MAML employs a first-order approximation. However, these approximations can oversimplify the problem, potentially degrading the learning quality, as we will demonstrate through simulations in Section V.

Note that traditional offline learning approaches are often limited by structural assumptions. However, this limitation does not apply to the offline meta-training method proposed in this section. Specifically, our approach does not rely on offline learning to directly solve the target task; instead, it is embedded within a meta-learning framework to learn a transferable prior that facilitates rapid adaptation to new structural scenarios. By training on a distribution of tasks with varying system dynamics and structural characteristics, the learned meta-parameters capture common structures across scenarios. When applied to a target system, the method performs online adaptation using only a small amount of target data, significantly reducing data collection costs while maintaining flexibility to structural variations.

IV. META-LEARNING FOR CONTROL

In Section III, we propose a general meta-learning framework. To achieve the control objective, this section aims to design both indirect and direct data-driven methods within the proposed framework. The key difference between the two approaches lies in whether an explicit system model is required, which directly influences the design of the loss functions for the base-learner and meta-learner. Particularly, we design specific formulations of $\widehat{\ell}(\cdot)$ for each approach to achieve the desired objective (3). These algorithms highlight

the generalizability of our framework, demonstrating that various controllers can be readily integrated.

A. Indirect meta-learning for control

In the indirect data-driven approach, a common system model is learned during the meta-training phase. During meta-adaptation, the model is rapidly adapted using target data to approximate the target system dynamics, after which a model-based controller is applied to compute the optimal controller.

1) *Preliminaries: NSSM and MPC:* To identify the system dynamics in (1), various representations can be employed, such as neural differential equations and Hammerstein-Wiener models [21], [22]. Although our framework is agnostic to the choice of representation, we illustrate the controller design by identifying the system with an NSSM, which extends traditional state-space models by leveraging neural networks to capture nonlinearity:

$$z_t = f_{\text{enc}}(U_{t-H+1:t}, Y_{t-H+1:t}), \quad (16a)$$

$$z_{t+1} = A_z z_t + B_z u_{t+1}, \quad (16b)$$

$$\hat{y}_t = C_z z_t, \quad (16c)$$

where $z_t \in \mathbb{R}^{n_z}$ is a latent state learned from past inputs and outputs:

$$\begin{aligned} U_{t-H+1:t} &:= \{u_{t-H+1}, u_{t-H+2}, \dots, u_t\}, \\ Y_{t-H+1:t} &:= \{y_{t-H+1}, y_{t-H+2}, \dots, y_t\}. \end{aligned} \quad (17)$$

Here, n_z and H are user-defined design parameters. Since the encoder f_{enc} maps a history of length H to the latent state z_t , the dimension of z_t must be sufficient to encode all relevant information from the input-output history. A practical rule of thumb is: $n_z \lesssim mH$, where m is the output dimension. This ensures that the latent state can capture the essential dynamics contained in the past H observations without being excessively large. Note that in (16), f_{enc} captures the system nonlinearity by fixing the initial condition of a linear system using past observations. The linear dynamics (16b)–(16c) then model temporal evolution.

Training NSSMs involves optimizing parameters $\omega := \{f_{\text{enc}}, A_z, B_z, C_z\}$ to minimize the identification error. Given a dataset of length $H + T$, the procedure is:

- 1) Construct a dataset $\mathcal{D}$ from input-output trajectories of length $H + T$:

$$\begin{aligned} U_{t-H+1:t+T} &:= \{u_{t-H+1}, u_{t-H+2}, \dots, u_{t+T}\}, \\ Y_{t-H+1:t+T} &:= \{y_{t-H+1}, y_{t-H+2}, \dots, y_{t+T}\}. \end{aligned}$$

- 2) Use past data $U_{t-H+1:t}$ and $Y_{t-H+1:t}$ to estimate the latent state z_t via (16a).
- 3) Predict future outputs $\hat{Y}_{t+1:t+T} = \{\hat{y}_{t+1}, \dots, \hat{y}_{t+T}\}$ recursively using (16b) and (16c).
- 4) Evaluate the identification accuracy using the loss:

$$\ell_{\text{SSM}}(\mathcal{D}; \omega) = \frac{1}{T} \|Y_{t+1:t+T} - \hat{Y}_{t+1:t+T}\|^2. \quad (18)$$

Since the NSSM captures the input-output dynamics of the system, it enables the design of model-based controllers. To this end, we rewrite (16b) and (16c) in compact form:

$$s_{t+1} = A s_t + B u_{t+1}, \quad (19)$$

where

$$s_{t+1} := \begin{bmatrix} z_{t+1} \\ \hat{y}_{t+1} \end{bmatrix}, \quad A := \begin{bmatrix} A_z & 0 \\ C_z A_z & 0 \end{bmatrix}, \quad B := \begin{bmatrix} B_z \\ C_z B_z \end{bmatrix}.$$

To track a reference trajectory, various controllers can be used; we focus on MPC for illustration due to the following reasons:

- MPC naturally accommodates input and output constraints and tracking objectives, which are essential in practical automation systems.
- The receding-horizon mechanism improves robustness against modeling inaccuracies, which is particularly beneficial during early adaptation stages.
- The NSSM structure results in linear latent dynamics (19), which ensures that the MPC formulation remains convex and computationally tractable.

Therefore, MPC is an implementable controller that aligns with the identified NSSM model and practical control requirements. It computes the control input by solving the optimization problem

$$\begin{aligned} \min_u & \tilde{y}_{t+N|t}^T \cdot \mathbb{P} \cdot \tilde{y}_{t+N|t} \\ & + \sum_{k=0}^{N-1} (\tilde{y}_{t+k|t}^T \cdot \mathbb{Q} \cdot \tilde{y}_{t+k|t} + \Delta u_{t+k|t}^T \cdot \mathbb{R} \cdot \Delta u_{t+k|t}) \\ \text{s.t. } & (19) \text{ and } u_{t+k|t} \in \mathcal{U}, \hat{y}_{t+k|t} \in \mathcal{Y}, k = 0, \dots, N-1, \end{aligned} \quad (20)$$

with $\tilde{y}_{t+k|t} := \hat{y}_{t+k|t} - \bar{y}_{t+k}$, $\Delta u_{t+k|t} := u_{t+k|t} - u_{t+k-1|t}$. Here, $\bar{y}_{t+k}$ is the reference signal, N is prediction horizon, and $\mathbb{Q} \succeq 0$ and $\mathbb{R} \succ 0$ are tuning weights. The choice of N involves a trade-off between control performance and computational burden. The tuning of $\mathbb{Q}$ and $\mathbb{R}$ follows standard linear quadratic regulatory principles. An aggressive state penalty ($\mathbb{Q} \gg \mathbb{R}$) forces the controller to minimize tracking errors rapidly, suitable for stabilization under large unmodeled disturbances. Conversely, heavily penalizing the control input ($\mathbb{R} \gg \mathbb{Q}$) yields smoother, more energy-efficient trajectories but may result in poor adaptation. Finally, the matrix $\mathbb{P}$ enables MPC stability guarantees and is typically chosen as the solution to a discrete-time algebraic Riccati equation [23].

2) *An MPC-based meta-learning control:* To address limited data in the target system, we first pretrain an aggregated NSSM using source system data, then fine-tune it on the target system to identify its dynamics before applying MPC for reference tracking. Based on the meta-learning framework in Section III, we now present the loss function design for both the base-learner and meta-learner.

Meta-training phase: In the scenario of indirect data-driven control, the meta-training phase aims to identify a general model that can represent a group of nonlinear systems by using the source datasets. In the inner problem, the base-learner evaluates the identification performance on each single source system leading to the loss function

$$\hat{\ell}(\mathcal{D}_{\text{tr}}^b; \psi) = \ell_{\text{SSM}}(\mathcal{D}_{\text{tr}}^b; \psi), \quad (21)$$

where $\ell_{\text{SSM}}(\cdot)$ is given in (18).

In the outer problem, the meta-learner evaluates the identi-

fication performance across different source systems. It solves the problem (6), where

$$\ell(\mathcal{D}_{\text{test}}^b; \tilde{\omega}^b(\omega)) = \ell_{\text{SSM}}(\mathcal{D}_{\text{test}}^b; \tilde{\omega}^b(\omega)). \quad (22)$$

Meta-adaptation phase: Upon convergence or reaching the maximum number of iterations, meta-training yields the aggregated NSSM with parameters denoted by ω that represent the shared similarities among source systems. In the meta-adaptation phase, ω is rapidly adapted to capture the target system's dynamics.

We adapt the aggregated NSSM to the target system by performing an update similar to (5), using the target dataset $\mathcal{D}_{\text{target}}$:

$$\omega^* := \underset{\psi}{\operatorname{argmin}} \ell_{\text{SSM}}(\mathcal{D}_{\text{target}}; \psi) + \frac{\gamma}{2} \|\psi - \omega\|^2. \quad (23)$$

In practice, this is efficiently approximated with a few gradient steps, using only limited data:

$$\omega^* \leftarrow \omega^* - \beta_{\text{in}} (\nabla_{\psi} \ell_{\text{SSM}}(\mathcal{D}_{\text{target}}; \psi)|_{\psi=\omega^*} + \gamma(\omega^* - \omega)). \quad (24)$$

The adapted NSSM, parameterized by ω^* , is then used for MPC-based reference tracking. To account for the bounded model mismatch arising from adaptation (see Proposition 2 in Appendix A), we employ a robust MPC formulation that replaces the nominal MPC (20) with:

$$\begin{aligned} \min_u & \tilde{y}_{t+N|t}^T \cdot \mathbb{P} \cdot \tilde{y}_{t+N|t} \\ & + \sum_{k=0}^{N-1} (\tilde{y}_{t+k|t}^T \cdot \mathbb{Q} \cdot \tilde{y}_{t+k|t} + \Delta u_{t+k|t}^T \cdot \mathbb{R} \cdot \Delta u_{t+k|t}) \\ \text{s.t. } & s_{t+k+1|t} = A s_{t+k|t} + B u_{t+k+1|t}, \\ & u_{t+k|t} \in \mathcal{U}, \quad \hat{y}_{t+k|t} \in \mathcal{Y} \ominus \mathcal{E}, \quad k = 0, \dots, N-1, \\ & s_{t+N|t} \in \mathcal{X}_f, \end{aligned} \quad (25)$$

where $\mathcal{E}$ is a compact set that bounds the prediction error induced by model uncertainty, and $\mathcal{X}_f$ is a robustly positively invariant terminal set required to guarantee recursive feasibility. In practice, let $\Delta_{\text{val}} := \max_{i \in \mathcal{D}_{\text{val}}} \|\hat{y}_i - y_i\|_{\infty}$, where $\mathcal{D}_{\text{val}} \subseteq \mathcal{D}_{\text{target}}$ is a small validation set collected from the target system. The tightening set can be set as $\mathcal{E} = \{e \in \mathbb{R}^m : \|e\|_{\infty} \leq \Delta_{\text{val}}\}$. Under standard robust MPC assumptions, this formulation guarantees input-to-state stability (ISS) of the closed-loop system, with tracking error proportional to the model mismatch (see Proposition 3 in Appendix A). The complete procedure is summarized in Algorithm 1.

Remark 2. *Stability guarantee is fundamental in optimal control. Since the proposed controller is designed based on a meta-learned model, the closed-loop performance depends directly on the learning quality. To clarify this relationship, our stability analysis establishes the following pathway:*

- (1) *The meta-training phase converges under standard smoothness and regularization assumptions and produces a well-defined shared initialization (i.e., the aggregated representation).*
- (2) *The subsequent task-specific adaptation in the meta-inference phase identifies the target system with a*

Algorithm 1 Indirect meta-learning-based control with NSSM and MPC implementation

Input: $\omega \leftarrow$ randomly initialize parameters of the NSSM
Input: $\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{target}} \leftarrow$ source dataset, target dataset
Input: Regularization strength γ , learning rates $\beta_{\text{in}}, \beta_{\text{out}}$, the maximum numbers of iterations $K_{\text{train}}, K_{\text{adapt}}$, and design parameters $H, n_z, \mathbb{Q}, \mathbb{R}, N$
Output: $\omega^* \leftarrow$ NSSM parameters obtained via identification on the target system
Output: $u_t^* \leftarrow$ optimal control for the target system
 % Meta-training phase
 $k = 0$
while not converge and $k < K_{\text{train}}$ **do** ▷ outer-loop
 Sample batch $\{\mathcal{D}^b\}_{b=1}^B$ from $\mathcal{D}_{\text{source}}$
 for $b = 1 : B$ **do** ▷ inner-loop
 Partition $\mathcal{D}^b$ into $\mathcal{D}_{\text{tr}}^b$ and $\mathcal{D}_{\text{test}}^b$
 $\omega^b \leftarrow$ solve (5) with loss function (21)
 $g^b \leftarrow$ solve (14) with Q^b, P^b defined by the loss function (22)
 end for
 $\omega \leftarrow \omega - \beta_{\text{out}} \frac{1}{B} \sum_{b=1}^B g^b$
 $k \leftarrow k + 1$
end while
 % Meta-adaptation phase
while $k < K_{\text{adapt}}$ **do**
 $\omega^* \leftarrow$ update ω^* by (24)
end while
 % MPC to find the optimal control input
for $t = 1, 2, \dots$ **do**
 $u_t^* \leftarrow$ solve the robust MPC (25) using the NSSM parameterized by ω^*
end for

bounded learning error on the target dataset (Proposition 2).

- (3) *When the adapted model is used for MPC and the model mismatch remains bounded in closed loop (Assumption 4), standard MPC theory guarantees input-to-state stability (Proposition 3).*

Therefore, stabilization follows from the combination of bounded learning error and standard robust MPC results. As compared to supervised learning without pretraining stage, meta-learning reduces adaptation error, which in turn tightens stability and tracking bounds in closed-loop control. For readability, the detailed theoretical analysis is provided in Appendix A.

B. Direct meta-learning for control

The method proposed in Section IV belongs to the category of indirect data-driven control. Such approaches are modular and naturally compatible with constraint handling through MPC. However, their performance may degrade due to model bias and identification errors. Moreover, the identified model may not be optimal for control design [24]. Direct approaches aim to address these limitations by learning control policies directly from data in an end-to-end manner. Reinforcement learning (RL), as one of the most widely adopted end-to-end paradigms, is therefore introduced to provide a direct data-driven realization of the same optimal control objective when accurate system models are subject to significant uncertainty.

Here we adopt a standard RL algorithm, the deep Q-network (DQN) [25], [26] as the base-learner. The conceptual simplicity of DQN makes the integration with our meta-learning framework transparent, and it has well-established convergence properties under appropriate conditions [27]. Importantly, our framework is not limited to DQN. The bi-level optimization structure (5)–(6) requires only that the loss function $\hat{\ell}(\cdot)$ is differentiable. More advanced RL algorithms, such as soft actor-critic methods [28], proximal policy optimization [29], and model-based RL approaches [30], can be integrated by defining $\hat{\ell}(\cdot)$ as their policy gradient or value function losses.

1) *A DQN-based RL reformulation:* To apply DQN, we reformulate the reference tracking task as an RL problem, where input and output constraints are naturally embedded in the Markov Decision Process (MDP) formulation by restricting the admissible state and action spaces. During meta-training, a so-called Q-network is pretrained to capture the similarities across similar systems. In the meta-adaptation phase, the network is fine-tuned with limited target data to directly learn an optimal control policy. To achieve reference tracking, the objective is to minimize the discounted cumulative cost $\sum_{t=0}^{\infty} \eta^t c(y_t, u_t)$, where $\eta \in (0, 1)$ is the discount factor that balances immediate and future losses. Moreover, $c(y_t, u_t)$ represents the stage cost at time t , defined as

$$c(y_t, u_t) := (y_t - \bar{y}_t)^T \mathbb{Q} (y_t - \bar{y}_t) + \Delta u_t^T \mathbb{R} \Delta u_t \quad (26)$$

with $\Delta u_t := u_t - u_{t-1}$.

Let $\mu(y)$ be the action taken under y . We can define the action-value function $Q^\mu(y, u)$ that takes the current action into account:

$$Q^\mu(y, u) = \sum_{t=0}^{\infty} \eta^t c(y_t, u_t), \quad (27)$$

where $y_0 = y, u_1 = u, u_t = \mu(y_t), t = 1, \dots, \infty$.

The optimal action-value function over all possible actions is denoted as $Q^*(y, u) = \min_{\mu} Q^\mu(y, u)$, which follows the well-known Bellman equation [31]:

$$Q^*(y, u) = c(y, u) + \min_{u'} \eta Q^*(y', u'). \quad (28)$$

Here y' denotes the next state of y , following the system dynamics (1) after applying the control input u' . If Q^* is available we can compute the optimal policy by

$$\mu(y) = \operatorname{argmin}_u Q^*(y, u).$$

However, since both y and u are real valued, computing Q^* involves solving equation (28) in the infinite dimensional space of real-valued functions. As this is practically impossible, we restrict attention to a finitely parameterised class of functions $Q(\cdot, \cdot | \omega) : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}$, where ω represents the adaptable parameter. In DQN, this is achieved using a neural network known as the Q-network. Suppose that a dataset $\mathcal{D} := \{U(T), Y(T)\}$, containing a T -length trajectory of system inputs and outputs, is available for evaluation. We first rearrange it as

$$\mathcal{D} = \{(y^{(i)}, u^{(i)}, y'^{(i)})\}_{i=1}^T,$$

that is, each data tuple consists of the output $y^{(i)}$, the input $u^{(i)}$, and the subsequent output $y'^{(i)}$. Inspired by (28), the Q-network is trained by adjusting its weights ω to reduce the mean-squared Bellman error

$$\ell_{\text{DQN}}(\mathcal{D}; \omega) = \frac{1}{2} \sum_{i=1}^T (\tilde{Q}(y^{(i)}, u^{(i)} | \omega_-) - Q(y^{(i)}, u^{(i)} | \omega))^2, \quad (29)$$

where

$$\tilde{Q}(y^{(i)}, u^{(i)} | \omega_-) = c(y^{(i)}, u^{(i)}) + \min_{u'} \eta Q(y'^{(i)}, u' | \omega_-). \quad (30)$$

In this equation, the optimal target value $c(y^{(i)}, u^{(i)}) + \min_{u'} \eta Q^*(y'^{(i)}, u')$ is substituted with the approximated target value $c(y^{(i)}, u^{(i)}) + \min_{u'} \eta Q(y'^{(i)}, u' | \omega_-)$, with parameter ω_- from the previous iteration. At each stage of optimization, we hold the parameters from the previous iteration ω_- fixed when optimizing the loss function $\ell_{\text{DQN}}(\mathcal{D}; \omega)$.

Differentiating the loss function (30) with respect to the weights yields the following gradient:

$$\nabla_{\omega} \ell_{\text{DQN}}(\mathcal{D}; \omega) = - \sum_{i=1}^T (\tilde{Q}(y^{(i)}, u^{(i)} | \omega_-) - Q(y^{(i)}, u^{(i)} | \omega)) \nabla_{\omega} Q(y^{(i)}, u^{(i)} | \omega). \quad (31)$$

We then update ω by minimizing $\ell_{\text{DQN}}(\mathcal{D}; \omega)$ using the gradient descent algorithm:

$$\omega \leftarrow \omega - \alpha \nabla_{\omega} \ell_{\text{DQN}}(\mathcal{D}; \omega). \quad (32)$$

As suggested by [27], the Q-network is further updated by

$$\omega_- \leftarrow \beta \omega_- + (1 - \beta) \omega, \quad (33)$$

where $\beta \in [0.5, 1)$. In alignment with standard DQN implementations [25], the target network update coefficient β is usually set approaching 1 (typically $\beta \geq 0.99$) to avoid instability in the Bellman targets during online adaptation. Conversely, a small learning rate α is generally recommended to ensure smooth and monotonic policy improvement without inducing gradient divergence.

Remark 3. *The convergence of DQN to the optimal action-value function Q^* requires the Markov property: the information available at each decision point must fully determine the system state. For system (1), when $g(\cdot)$ is injective, the current output y_t uniquely determines the state x_t , and we use y_t as input to the Q-network. When $g(\cdot)$ is not injective, we use an output history $Y_t = [y_{t-n-1}, \dots, y_t]$ as the Q-network input. For observable systems, this output history fully reconstructs the state, satisfying the Markov property. Under these conditions and standard RL assumptions [27], DQN provably converges to the optimal Q^* . Throughout this subsection, the derivations apply with y replaced by Y when the non-injective case is considered. Here, “optimal” indicates that the solution satisfies the Bellman equation (28), and convergence can be achieved when the Q-network gradients are exactly computed. Thus, practical neural network approximation may introduce errors.*

2) *A DQN-based meta-learning control:* To enable rapid adaptation in dynamic environments, we embed DQN within the proposed meta-learning framework. During the meta-training phase, a shared parameter ω is learned from different source systems using $\mathcal{D}_{\text{source}}$. Subsequently, the meta-adaptation phase fine-tunes ω to determine the optimal control policy for the target system, as specified below.

Meta-training phase: By using DQN, the base-learner for each source system, labeled as b , solves the inner problem (5) to yield $\omega^b(\omega)$ with

$$\hat{\ell}(\mathcal{D}_{\text{tr}}^b; \psi) = \ell_{\text{DQN}}(\mathcal{D}_{\text{tr}}^b; \psi). \quad (34)$$

On the other hand, in the outer problem, the meta-learner solves (6), where

$$\ell(\mathcal{D}_{\text{test}}^b; \omega^b(\omega)) = \ell_{\text{DQN}}(\mathcal{D}_{\text{test}}^b; \omega^b(\omega)). \quad (35)$$

By (7) and (10), this requires evaluating the gradient (31).

As standard in RL, in the meta-training phase, the control input u^b is selected using the ϵ -greedy policy that balances the exploration and exploitation:

$$u^b = \begin{cases} \operatorname{argmin}_u Q(y^b, u, \omega^b(\omega)), & \text{with probability } 1 - \epsilon, \\ \tilde{u}^b \sim \mathcal{N}(0, \Sigma^b), & \text{with probability } \epsilon. \end{cases} \quad (36)$$

That is, the ϵ -greedy policy generates new sample data based on latest ω^b .

Algorithm 2 Direct meta-learning-based control with DQN implementation

Input: $\omega \leftarrow$ randomly initialize the weights to parameterize the optimal action-value function
Input: $\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{target}} \leftarrow$ source dataset, target dataset
Input: Regularization strength γ , learning rates $\beta_{\text{in}}, \beta_{\text{out}}$, the maximum numbers of iterations $K_{\text{train}}, K_{\text{adapt}}$, and design parameters $Q, \mathbb{R}, \alpha, \beta, \eta$
Output: $\omega^* \leftarrow$ weights of the optimal action-value function for the target system
Output: $u_t^* \leftarrow$ optimal control for the target system
% Meta-training phase
 $k = 0$
while not converge and $k < K_{\text{train}}$ **do** ▷ outer-loop
 Sample batch $\{\mathcal{D}_{\text{tr}}^b\}_{b=1}^B$ from $\mathcal{D}_{\text{source}}$
 for $b = 1 : B$ **do** ▷ inner-loop
 Partition $\mathcal{D}^b$ into $\mathcal{D}_{\text{tr}}^b$ and $\mathcal{D}_{\text{test}}^b$
 $\omega^b \leftarrow$ solve (5) with loss function (34)
 $\omega_-^b \leftarrow$ update ω_-^b via (33)
 $g^b \leftarrow$ solve (14) with Q^b, P^b defined by the loss function (35)
 $u^b \leftarrow$ choose the new input by (36)
 $y^b \leftarrow$ simulate the system (1) using u^b with $\theta = \theta^b$
 Update $\mathcal{D}_{\text{source}}$ by adding (u^b, y^b)
 end for
 $\omega \leftarrow \omega - \beta_{\text{out}} \frac{1}{B} \sum_{b=1}^B g^b$
end while
% Meta-adaptation phase
while $k < K_{\text{adapt}}$ **do**
 $\omega^* \leftarrow$ update ω^* by (37)
end while
% Directly find the control input
for $t = 1, 2, \dots$ **do**
 $u_t^* \leftarrow$ solve (38)
end for

Meta-adaptation phase: Once the meta-training phase completes, ω is learned as a general approximation of action-value functions across source systems. Similar to (24), in the meta-adaptation phase, ω is fine-tuned using a small amount of data from the target system:

$$\omega^* \leftarrow \omega^* - \beta_{\text{in}} (\nabla_{\psi} \ell_{\text{DQN}}(\mathcal{D}_{\text{target}}; \psi)|_{\psi=\omega^*} + \gamma(\omega^* - \omega)). \quad (37)$$

Then, during the real-time operation, the optimal control input is directly obtained by solving

$$u_t^* = \operatorname{argmin}_u Q(y_t, u, \omega^*). \quad (38)$$

The full procedure is summarized in Algorithm 2. Compared with Algorithm 1, Algorithm 2 does not aim to identify the system model. Instead, it implicitly learns the control policy by optimizing the action-value functions.

As shown in this section, the framework proposed in Section III can accommodate both indirect and direct learning algorithms for controller design. From a structural perspective, indirect methods parameterize the system model and subsequently perform model-based control design, whereas direct methods parameterize the control policy (or value function) and optimize it directly from data. From an optimization viewpoint, both approaches solve the same bi-level meta-learning problem. The difference lies in the choice of decision variables (model parameters versus policy parameters) and in how approximation errors propagate (model bias versus policy approximation bias). The motivation for studying both paradigms is therefore to demonstrate that the proposed framework is sufficiently general to accommodate both realizations under a unified optimization structure. It should also be emphasized that the applicability of the framework is not limited to the examples discussed in Sections IV-A and IV-B.

V. SIMULATION AND EXPERIMENT

In this section, we will illustrate the performance of our proposed algorithms by using both numerical simulations and experiments.

A. Numerical verification of Algorithm 1

We first present numerical examples to evaluate the performance of Algorithm 1. Let us consider a family of van der Pol oscillators whose dynamics are given by

$$\begin{aligned} \dot{x}_1 &= x_2 + u_1, \quad \dot{x}_2 = \theta x_2 (1 - x_1^2) - x_1 + u_2, \\ y &= x, \end{aligned} \quad (39)$$

where $\theta \sim \mathcal{N}(0, 1)$ is the unknown damping ratio.

To ensure full reproducibility, the detailed hyperparameter configurations for the proposed algorithm are summarized below.

- **NSSM Architecture:** To build the NSSM, the dimension of the latent state z_t in (16) is set to $n_z = 5$. The history length for the encoder is set to $H = 10$. The encoder f_{enc} is parameterized by a Multi-Layer Perceptron (MLP) with one input layer, one hidden layer, and one output layer, each containing 128 neurons with ReLU activation

functions. The linear state-space matrices A_z , B_z , and C_z are randomly initialized using a normal distribution scaled by 0.1.

- **iMAML:** During the meta-training phase, 10 source systems are generated by sampling $\theta^b \sim \mathcal{N}(0, 1)$. The dataset of each source system contains a trajectory of length 10000. In each outer loop, a batch of $B = 16$ datasets is sampled from $\mathcal{D}_{\text{source}}$. The regularization strength is set to $\gamma = 100$, while the learning rates are $\beta_{\text{in}} = 10^{-4}$ and $\beta_{\text{out}} = 10^{-3}$, respectively. The number of inner-loop gradient steps is set to $M = 2$. The meta-learner is optimized using the Adam optimizer, and gradients are clipped at a maximum norm of 0.5 to ensure training stability.
- **MPC Configuration:** The prediction horizon in MPC is set to $N = 5$. Both $\mathbb{Q}$ and $\mathbb{R}$ are chosen as identity matrices.

All numerical simulations and network training are repeated across 10 different random seeds.

1) *Comparison with learning algorithms:* We first compare Algorithm 1 with a few learning algorithms, all of which have the same NSSM architecture:

- **iMAML (Our approach):** The NSSM parameters are pretrained on the source data $\mathcal{D}_{\text{source}}$ and subsequently adapted to the target data $\mathcal{D}_{\text{target}}$ using Algorithm 1.
- **MAML:** MAML also employs a meta-learning framework. However, as noted in Remark 1, it solves a different bi-level optimization problem from iMAML and applies a first-order approximation.
- **Supervised learning:** A standard learning approach without pretraining. NSSM parameters are initialized randomly and trained solely on target data $\mathcal{D}_{\text{target}}$.

For consistency, the meta-training phase in **MAML** is conducted using the exact same source dataset as in **iMAML**.

After pretraining (**iMAML** and **MAML**) or random initialization (**Supervised learning**), we adapt the NSSMs using the target dataset $\mathcal{D}_{\text{target}}$ containing 300 data points. Fig. 4 illustrates the tracking performance on the target system. The goal is to track a circular trajectory using MPC with NSSMs adapted after 0, 100, and 3000 iterations of (24), respectively. From the first column, we conclude that when the meta-training phase completes, both **iMAML** and **MAML** produce good aggregated models that capture the dynamics of the target system as evidenced by their ability to follow the overall trend of the reference trajectory from the start. The performance is further improved after the adaptation steps using target data $\mathcal{D}_{\text{target}}$. Notably, **iMAML** shows faster convergence and superior tracking, particularly during early meta-adaptation. This is attributed to its more accurate computation of meta-gradient than **MAML** (see Remark 1), which leads to better initialization and more effective adaptation. In contrast, since **supervised learning** does not benefit from the source systems, it performs poorly at the beginning and converges slower.

2) *Comparison with adaptive control algorithms:* Alongside learning algorithms, adaptive control remains a common approach for reference tracking tasks in uncertain systems. Therefore, we include numerical comparisons with two representative adaptive control methods: Model-Free Adaptive

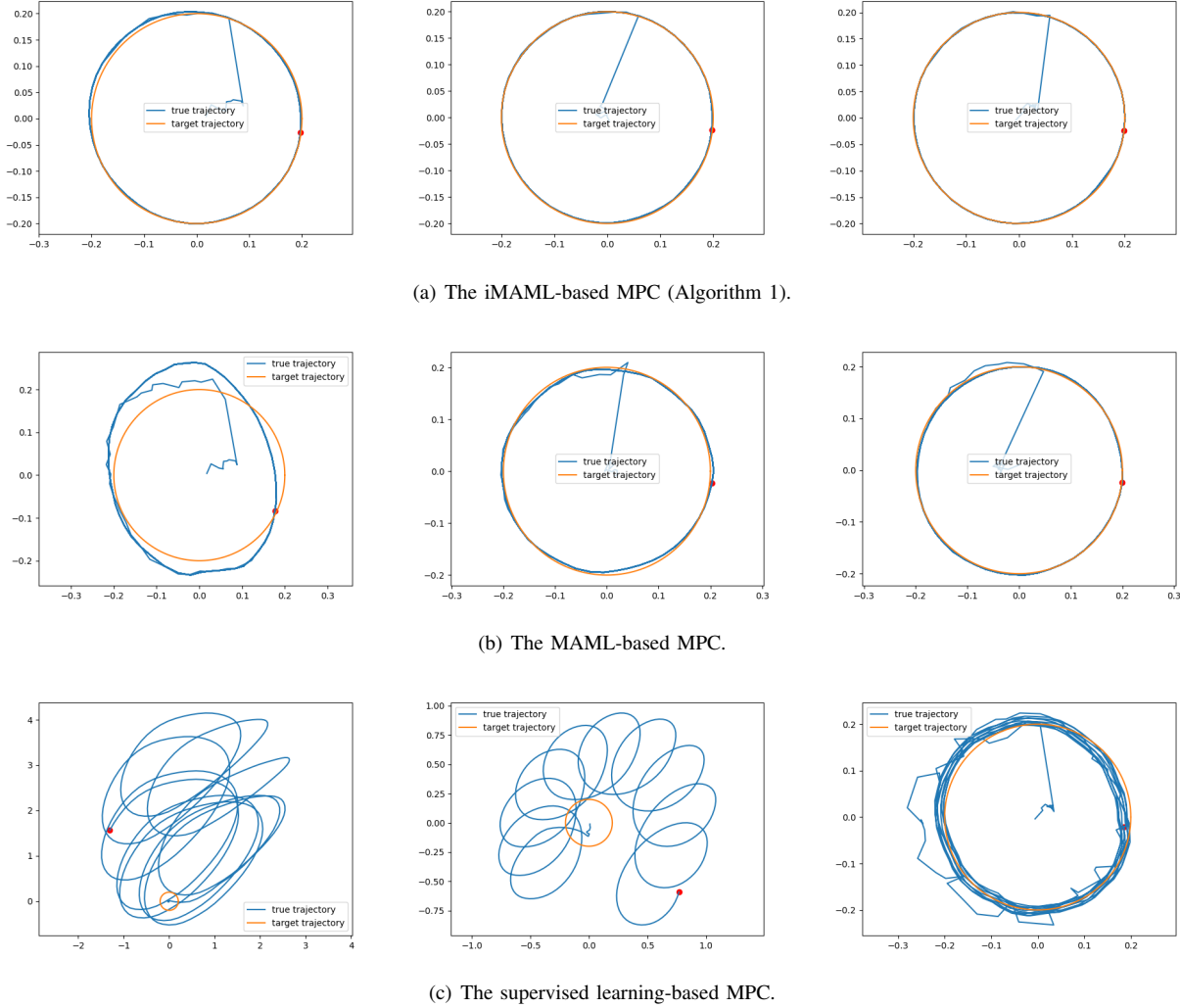


Fig. 4: Comparison of the tracking performance on the target system, where the three columns show the performance of different algorithms by using the NSSMs adapted after 10, 100, and 3000 steps, respectively.

Control (MFAC) [32] and backstepping-type adaptive control [33]. These baseline algorithms are respectively evaluated against Algorithm 1 on cloverleaf and Figure-8 trajectories, as illustrated in Fig. 5. To minimize the impact of the initial transient phase, both the full Root Mean Square Error (RMSE) and the steady-state RMSE after $K_b = 80$ burn-in steps are computed to evaluate tracking performance:

$$\text{Full RMSE} = \sqrt{\frac{1}{500} \sum_{k=1}^{500} \|y_k - \bar{y}_k\|^2},$$

$$\text{Steady RMSE} = \sqrt{\frac{1}{500 - K_b} \sum_{k=K_b+1}^{500} \|y_k - \bar{y}_k\|^2}.$$

As detailed in Table I, Algorithm 1 consistently outperforms both adaptive control baselines. By transferring knowledge from the source systems, it achieves lower steady-state tracking errors across different reference trajectories.

TABLE I: Tracking performance under different trajectories

Method	Cloverleaf		Figure-8	
	Full RMSE	Steady RMSE	Full RMSE	Steady RMSE
MFAC	0.0102	0.0098	0.0121	0.0124
Backstepping	0.0107	0.0104	0.0133	0.0122
Ours	0.0019	0.0020	0.0027	0.0027

3) *Hyperparameter sensitivity*: Finally, we test the performance of Algorithm 1 under different hyperparameters. Specifically, we vary three key hyperparameters: the inner-loop learning rate β_{in} , the regularization strength γ , and the MPC prediction horizon N . The steady-state RMSE and the average computational time per iteration under different settings are summarized in Tables II and III.

Based on these results, several insights can be drawn:

- **Sensitivity to β_{in}** : The learning rate determines the update magnitude of the NSSM parameters during the online adaptation phase. In the considered tasks, a relatively small learning rate (e.g., $\beta_{\text{in}} = 5 \times 10^{-4}$) yields the

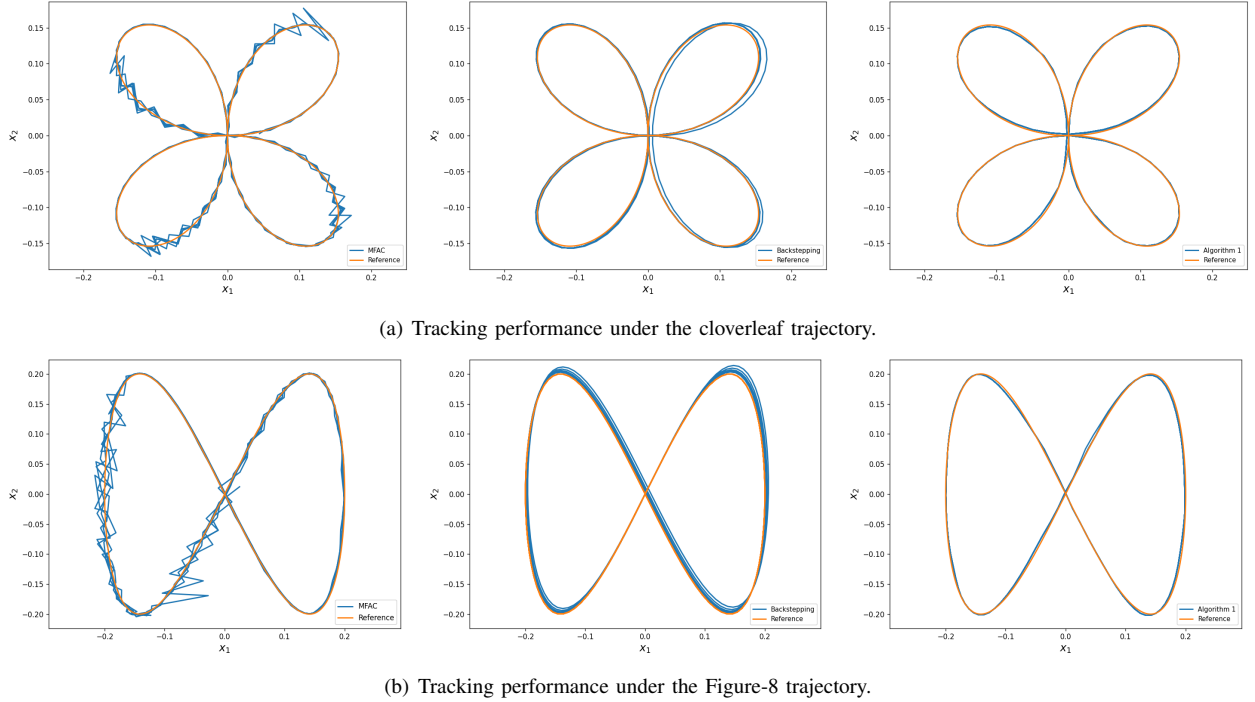


Fig. 5: Comparison of the tracking performance under different trajectories. From left to right, the columns present the performance of MFAC, backstepping control, and Algorithm 1.

TABLE II: Steady RMSE under different β_{in}

Prediction Horizon	Learning Rate β_{in}			Comp. Time / Iter (ms)
	0.0005	0.001	0.002	
$N = 1$	0.1033	0.1041	0.1037	1.68
$N = 2$	0.0168	0.0214	0.0190	4.52
$N = 3$	0.0033	0.0083	0.0081	6.64
$N = 5$	0.0022	0.0070	0.0080	11.01
$N = 10$	0.0022	0.0070	0.0078	22.16

TABLE III: Steady RMSE under different γ

Prediction Horizon	Regularization Strength γ				Comp. Time / Iter (ms)
	10	100	200	500	
$N = 1$	0.1033	0.1033	0.1036	0.1033	1.01
$N = 2$	0.0169	0.0164	0.0170	0.0160	4.18
$N = 3$	0.0042	0.0034	0.0038	0.0043	6.34
$N = 5$	0.0035	0.0027	0.0026	0.0043	11.77
$N = 10$	0.0035	0.0028	0.0026	0.0043	26.76

best tracking accuracy. When β_{in} is increased to 10^{-3} or 2×10^{-3} , the controller still maintains stable tracking, but the steady-state RMSE slightly increases. This indicates that an overly aggressive adaptation step size may cause the fine-tuned model to deviate from the meta-learned initialization, thereby degrading the prediction accuracy of MPC.

- **Sensitivity to γ :** The regularization strength γ governs the target adaptation process. A small γ may cause the adapted NSSM model to deviate significantly from the pre-trained meta-initialization, whereas an overly large γ restricts the adaptation capability. The optimal perfor-

mance is observed as $\gamma \in [100, 200]$. Furthermore, within a reasonable range, the proposed method achieves low sensitivity to γ .

- **Sensitivity to N and computational efficiency:** The prediction horizon N primarily impacts tracking performance when it is very small. Specifically, an insufficient horizon (e.g., $N = 1$) results in inadequate predictive capability, leading to a large steady-state RMSE. As N increases to 3 and 5, the tracking error decreases significantly. However, further increasing N from 5 to 10 yields marginal improvements in tracking accuracy while significantly increasing the computational time.

Overall, these results confirm that the proposed framework does not require meticulously fine-tuned hyperparameters or excessively large prediction horizons to succeed. The computational overhead is strictly bounded and scalable, ensuring practical implementations.

B. Experimental verification of Algorithm 2

To evaluate the performance of the direct meta-learning-based controller, we test Algorithm 2 on a physical ball-on-a-plate system, as shown in Fig. 6. The system comprises a square plate, actuated by two servo motors that control the plate's rotation about the x - and y -axes. A ball rests in the plate and can be moved by tilting the plate correspondingly. A camera mounted over the plate can be used to track the ball position. The control loop is closed through a raspberry pi board.

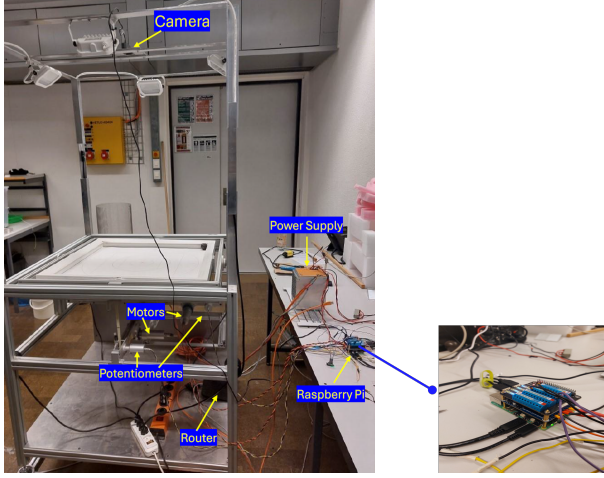


Fig. 6: The ball-on-a-plate system [34].

The system is nonlinear with uncertainties introduced by friction. The dynamics can be described by [35] and [34]:

$$\begin{aligned}\ddot{x} &= \frac{5}{7}g \sin \beta \cos \alpha + \frac{F_{\text{friction},x}}{m}, \\ \ddot{y} &= -\frac{5}{7}g \sin \alpha + \frac{F_{\text{friction},y}}{m},\end{aligned}\quad (40)$$

where x and y represent the ball's position along the x - and y -axes; α and β denote the plate angles of rotation about the x - and y -axes; $F_{\text{friction},x}$ and $F_{\text{friction},y}$ are the frictional forces projected along the x - and y -directions; and m is the mass of the ball.

We adopt the classical Stribeck curve as the friction model, following the formulation in [35]. The mathematical expression is given by

$$F_{\text{friction}} = \left(F_C + (F_S - F_C) e^{-\left(\frac{\|v\|}{v_S}\right)^{\delta_S}} \right) \text{sgn}(v) + F_v v,$$

where F_C is the kinetic friction coefficient, F_S is the static friction coefficient, F_v denotes the viscous friction coefficient, v_S represents the Stribeck velocity, δ_S depends on the contact surface geometry, and $v = [\dot{x}_s, \dot{y}_s, 0]^T$ is the ball's velocity. A graphical illustration of these parameters is provided in Fig. 7.

Since the true friction parameters are unknown and collecting data from the real system is challenging, we create a batch of source systems in simulation. For each source system, F_s , F_c , and F_v are independently drawn from a uniform distribution $\mathcal{U}(0, 0.01)$, while $v_s = 0.01$ and $\delta_S = 1$ are fixed. Moreover, in the source systems, the control input (desired plate angles) is strictly bounded within $[-10^\circ, 10^\circ]$, and the maximum angular rate of the actuators is limited to $\dot{\alpha}_{\max} = \dot{\beta}_{\max} = 30^\circ/\text{s}$ to respect hardware limitations. Furthermore, to accurately simulate boundary collisions, the elasticity coefficient is set to 0.6, meaning the ball retains 60% of its pre-impact normal velocity to account for the realistic kinetic energy loss at the boundaries. These systems share the similar model as the real system but have different, unknown friction parameters.

1) *Comparison with standard DQN*: To evaluate the effectiveness of our algorithms, we consider two scenarios:

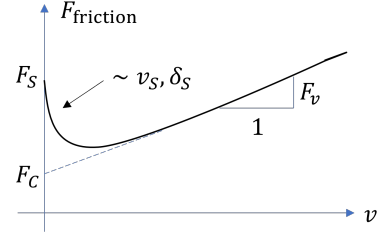


Fig. 7: Velocity-dependent friction model.

- 1) The Q-network is first pretrained in simulation using data from source systems and subsequently adapted to the real physical system using Algorithm 2.
- 2) A standard DQN is trained directly on the real system with randomly initialized weights.

The state and input penalty matrices are configured as $\mathbb{Q} = \text{diag}(8, 1, 8, 1)$ and $\mathbb{R} = \text{diag}(7, 7)$, respectively. The discount factor for the infinite-horizon cost is set to $\eta = 0.9$. Following standard setups in DQN, we choose $\alpha = 0.001$ and $\beta = 0.99$ in (32) and (33).

During the offline meta-training phase, Algorithm 2 samples 10 source systems (representing varying friction conditions) in simulation per iteration. We maintain independent experience replay buffers for each source system. Each buffer has a capacity of 10^3 transitions of the form $((x, y), u, c, (x', y'))$. New transitions generated using the ϵ -greedy policy (with ϵ decaying from 1.0 to 0.05) are appended to the corresponding buffer following a First-In-First-Out (FIFO) rule. Crucially, during meta-training, mini-batches uniformly sampled from each buffer are strictly partitioned into disjoint training sets ($\mathcal{D}_{\text{tr}}^b$, size 16) and testing sets ($\mathcal{D}_{\text{test}}^b$, size 8) to prevent data leakage.

The inner learning rate is set to $\beta_{\text{in}} = 0.001$. We further let the regularization strength be $\gamma = 10$. After evaluating the inner-loop adaptations for these tasks, the outer learning rate is set to $\beta_{\text{out}} = 0.01$ to aggregate the task-specific gradients and compute the optimal global initialization weights.

The learning algorithms compute and send the desired plate angles in the x - and y -directions based on the ball's current state. These angles serve as control inputs that ultimately adjust the ball's position accordingly.

For exploration, in addition to the ϵ -greedy policy used in Algorithm 2, we introduce an "input intermezzo" mechanism. That is, after a fixed operating period, we intentionally override the current policy by commanding a predefined plate angle based on the ball's reference position. The input intermezzo is used to repeatedly reset the ball to positions, which are otherwise rarely attained under ϵ -greedy actions paired with a well-trained tracking controller. This, in particular, includes collisions with the plate boundary. The input intermezzo thereby empirically achieved convergence speed.

Figs. 8 and 9 compare the tracking performance on the real system under different algorithms. The plots show the reference signal, which follows a square trajectory (green line), alongside the actual position of the ball (blue line). The short intervals between the dotted red lines represent input intermezzos used for exploration. It is evident that pretraining

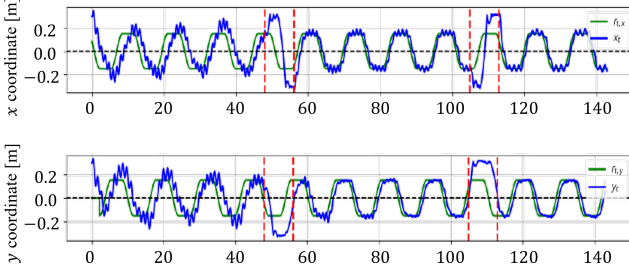


Fig. 8: Ball position on the real system after pre-training with Algorithm 2. The red dashed lines indicate the input intermezzo.

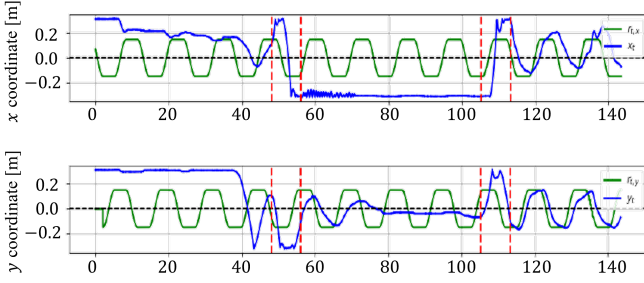


Fig. 9: Ball position of DQN by only using data from the real system. The red dashed lines indicate the input intermezzo.

the network weights using Algorithm 2 offers a significant advantage: the overall trend of the reference trajectory is captured from the very beginning (Fig. 8). Moreover, over the 150-second experiment, the pretrained weights lead to a 73.58% reduction in mean cost compared to direct deployment on the real system without any pretraining (Fig. 9). While oscillations are observable in the results, these are expected to diminish with more training data. Our experiments confirm that the meta-learning framework enhances tracking performance in real-world applications, highlighting its effectiveness for sim-to-real transfer. This is in line with hardware validations of MAML-based control approaches in the literature, e.g., [12], where MAML improves learning speed and performance of a learning MPC to unseen control objectives and nonlinearities.

VI. CONCLUSION

This paper proposes a meta-learning framework based on the iMAML algorithm for optimal control in uncertain nonlinear systems. By pretraining with offline source data and fine-tuning with (limited) online target data, the framework enables fast adaptation and improved control performance. We also introduce two algorithms that can be integrated into our framework for controller design, utilizing MPC and deep Q-networks respectively. Theoretical analysis on convergence of the meta-learning framework and stability of the learning-based control algorithms is further given. Numerical and hardware experiments finally confirm the superiority of our methods over baseline approaches. Future work will focus on improving the scalability of the proposed framework and extending its applicability to more complex, real-world robotic

systems, such as legged robots and autonomous vehicles. Additionally, we will explore integration with state-of-the-art RL algorithms such as soft actor-critic and model-based planning methods to further improve sample efficiency and tracking performance.

APPENDIX A THEORETICAL ADDITIONS

In this section, we present additional theoretical results on the convergence of the proposed framework and the stability of the resulting control algorithms. To highlight the generality of the analysis, we adopt the notation used in the general framework and thus denote the loss function by $\hat{\ell}(\cdot)$, which corresponds to $\ell_{\text{SSM}}(\cdot)$ and $\ell_{\text{DQN}}(\cdot)$ in Algorithms 1 and 2, respectively.

(1) Convergence of the meta-training phase

In Section III, the inner problem (5) and outer problem (6) are solved using iterative methods (7) and (8), respectively. To show their convergence, we adopt the standard assumptions:

Assumption 1. For all $\mathcal{D}$ and ψ , the loss functions $\hat{\ell}(\mathcal{D}; \psi)$ satisfy:

- (i) (***L-smoothness***) The function $\hat{\ell}(\mathcal{D}; \psi)$ is L -smooth in ψ , i.e.,

$$\|\nabla \hat{\ell}(\mathcal{D}; \psi_1) - \nabla \hat{\ell}(\mathcal{D}; \psi_2)\| \leq L \|\psi_1 - \psi_2\|.$$

- (ii) (***Bounded Hessian***) The Hessian of $\hat{\ell}(\mathcal{D}; \psi)$ is uniformly bounded, i.e., there exists $\hbar > 0$ such that

$$\|\nabla^2 \hat{\ell}(\mathcal{D}; \psi)\| \leq \hbar.$$

Assumption 2. The regularization parameter satisfies $\gamma > \hbar$.

For neural network losses, $\hbar$ is not directly verifiable; in practice, γ is a design parameter chosen large enough to yield well-conditioned inner problems, and the sensitivity study in Table III shows low sensitivity across an order of magnitude in γ . Assumptions 1–2 ensures the inner problem (5) is $(\gamma - \hbar)$ -strongly convex, which guarantees: (i) linear convergence of gradient descent, as proved below, and (ii) invertibility of $Q^b = I + \frac{1}{\gamma} \nabla^2 \hat{\ell} \succ 0$ in Lemma 1, which is required for implicit differentiation.

The convergence of the meta-training phase is guaranteed by the following proposition:

Proposition 1 (Meta-Training Convergence). *Let Assumptions 1–2 hold. With appropriate learning rate β_{out} and M inner-loop gradient steps with step size $\beta_{\text{in}} = 1/(L + \gamma)$, it follows that*

$$\min_{k < K} \|\nabla L(\omega_k)\|^2 \leq \underbrace{\frac{C}{K}}_{\text{outer-loop rate}} + \underbrace{O\left(\left(1 - \frac{\gamma - \hbar}{L + \gamma}\right)^M\right)}_{\text{inner-loop approximation error}}$$

where $C := 2L_{\text{meta}}(L(\omega_0) - L^*)$ with the constant L_{meta} depends on L , $\hbar$, and γ .

Proof. The proof quantifies the optimization errors arising from the inner-loop adaptation and the outer-loop meta-update, respectively.

Step 1 (Inner-loop approximation error). Let $f(\psi)$ denote the regularized inner problem objective in (5). Its Hessian is given by:

$$\nabla^2 f(\psi) = \nabla^2 \ell(\psi) + \gamma I.$$

By Assumption 1(ii), the eigenvalues of $\nabla^2 \ell$ are bounded within $[-\hbar, \hbar]$, which implies $\nabla^2 f(\psi) \succeq (\gamma - \hbar)I$. Consequently, the inner problem is $(\gamma - \hbar)$ -strongly convex and $(L + \gamma)$ -smooth. For strongly convex and smooth functions, applying gradient descent with a step size of $\beta_{\text{in}} = 1/(L + \gamma)$ guarantees that the distance to the exact optimum contracts linearly. After M steps, the approximation error decays exponentially at a rate of $\mathcal{O}\left(\left(1 - \frac{\gamma - \hbar}{L + \gamma}\right)^M\right)$.

Step 2 (Outer-loop convergence). Under Assumption 1 and applying the chain rule alongside the implicit function theorem (see, e.g., [13, Theorem 1]), the meta-objective $L(\omega)$ is L_{meta} -smooth. Furthermore, Assumptions 1–2 ensure $L_{\text{meta}} < \infty$. Using the standard descent lemma for smooth, non-convex functions [36], each outer-loop gradient step decreases the objective function L bounded by:

$$L(\omega_{k+1}) \leq L(\omega_k) - \frac{1}{2L_{\text{meta}}} \|\nabla L(\omega_k)\|^2.$$

Therefore, it follows

$$\frac{1}{2L_{\text{meta}}} \|\nabla L(\omega_k)\|^2 \leq L(\omega_k) - L(\omega_{k+1}).$$

Summing this inequality from $k = 0$ to $K - 1$, yields

$$\frac{1}{2L_{\text{meta}}} \sum_{k=0}^{K-1} \|\nabla L(\omega_k)\|^2 \leq L(\omega_0) - L(\omega_K).$$

Since $L(\omega_K) \geq L^*$, we can bound the right side as $L(\omega_0) - L(\omega_K) \leq L(\omega_0) - L^*$, which gives

$$\sum_{k=0}^{K-1} \|\nabla L(\omega_k)\|^2 \leq 2L_{\text{meta}}(L(\omega_0) - L^*).$$

As a result, we obtain

$$\begin{aligned} \min_{k < K} \|\nabla L(\omega_k)\|^2 &\leq \frac{1}{K} \sum_{k=0}^{K-1} \|\nabla L(\omega_k)\|^2 \\ &\leq \frac{2L_{\text{meta}}(L(\omega_0) - L^*)}{K}. \end{aligned}$$

By substituting the constant $C := 2L_{\text{meta}}(L(\omega_0) - L^*)$, we exactly recover the outer-loop convergence rate of $\frac{C}{K}$.

The overall meta-gradient evaluated during training is subject to the inexactness of the inner-loop solution. Incorporating the exponentially decaying inner-loop approximation error into the outer-loop gradient bound yields the final upper bound, completing the proof. $\square$

The bound is on $\min_{k < K} \|\nabla L(\omega_k)\|^2$ instead of $L(\omega_K) - L^*$, because the meta-objective is nonconvex: gradient descent can only guarantee convergence to a stationary point (some iterate of the gradient descent where the gradient becomes small), not to the global minimum.

The step size $\beta_{\text{in}} = 1/(L + \gamma)$ is the standard choice for gradient descent on smooth functions: since the regularized

inner problem has smoothness constant $L + \gamma$, this step size guarantees convergence and achieves the linear rate stated above.

(2) Bounded adaptation error in meta-inference phase

By Proposition 1, after sufficient number of iterations, the meta-training phase converges to some ω_∞ which will then be fine-tuned on the target system:

$$\tilde{\omega}^* := \underset{\psi}{\operatorname{argmin}} \hat{\ell}(\mathcal{D}_{\text{target}}; \psi) + \frac{\gamma}{2} \|\psi - \omega_\infty\|^2. \quad (41)$$

In practice, this is solved with a few gradient steps, using only limited data:

$$\omega^* \leftarrow \omega^* - \beta_{\text{in}} (\nabla_\psi \hat{\ell}(\mathcal{D}_{\text{target}}; \psi)|_{\psi=\omega^*} + \gamma(\omega^* - \omega_\infty)). \quad (42)$$

We further denote the optimal parameters for the target system by ω_{opt}^* . The key assumption for adaptation is *task similarity*: the meta-learned initialization ω_∞ should be close to ω_{opt}^* . Therefore, we assume some ρ exists such that $\|\omega_\infty - \omega_{\text{opt}}^*\| \leq \rho$.

We can prove that the adaptation error is bounded:

Proposition 2 (Adaptation Error). *Let Assumptions 1–2 hold. After K_{adapt} gradient steps of (42) with step size $\beta_{\text{in}} = 1/(L + \gamma)$, the adapted parameters ω^* satisfy:*

$$\begin{aligned} &\ell(\mathcal{D}_{\text{target}}; \omega^*) - \ell(\mathcal{D}_{\text{target}}; \omega_{\text{opt}}^*) \\ &\leq \underbrace{\frac{\gamma \rho^2}{2}}_{\text{regularization bias}} + \underbrace{\frac{L + \gamma}{2} \left(1 - \frac{\gamma - \hbar}{L + \gamma}\right)^{K_{\text{adapt}}}}_{\text{optimization error}} \|\omega_\infty - \tilde{\omega}^*\|^2. \end{aligned}$$

The proof compares the adapted parameters ω^* to two reference points: (i) the exact minimizer $\tilde{\omega}^*$ of the regularized objective (41), and (ii) the unconstrained optimum ω_{opt}^* . The regularization bias arises from (ii), while the optimization error arises from (i).

Proof. Let $f(\psi) := \hat{\ell}(\mathcal{D}_{\text{target}}; \psi) + \frac{\gamma}{2} \|\psi - \omega_\infty\|^2$ denote the regularized objective.

Step 1 (Optimality gap for regularized objective). By the optimality of $\tilde{\omega}^*$:

$$f(\tilde{\omega}^*) \leq f(\omega_{\text{opt}}^*) = \ell(\mathcal{D}_{\text{target}}; \omega_{\text{opt}}^*) + \frac{\gamma}{2} \|\omega_{\text{opt}}^* - \omega_\infty\|^2.$$

Step 2 (Optimization error). Since the regularized objective f is $(\gamma - \hbar)$ -strongly convex and $(L + \gamma)$ -smooth, gradient descent with step size $1/(L + \gamma)$ ensures [36]:

$$\|\omega^* - \tilde{\omega}^*\|^2 \leq \left(1 - \frac{\gamma - \hbar}{L + \gamma}\right)^{K_{\text{adapt}}} \|\omega_\infty - \tilde{\omega}^*\|^2.$$

By $(L + \gamma)$ -smoothness of f :

$$f(\omega^*) \leq f(\tilde{\omega}^*) + \frac{L + \gamma}{2} \|\omega^* - \tilde{\omega}^*\|^2.$$

Step 3 (Combining). From Steps 1 and 2, it follows

$$f(\omega^*) \leq f(\omega_{\text{opt}}^*) + \frac{L + \gamma}{2} \|\omega^* - \tilde{\omega}^*\|^2.$$

Expanding f and using $\|\omega_{\text{opt}}^* - \omega_{\infty}\| \leq \rho$:

$$\begin{aligned} \ell(\mathcal{D}_{\text{target}}; \omega^*) &+ \frac{\gamma}{2} \|\omega^* - \omega_{\infty}\|^2 \\ &\leq \ell(\mathcal{D}_{\text{target}}; \omega_{\text{opt}}^*) + \frac{\gamma \rho^2}{2} + \frac{L + \gamma}{2} \|\omega^* - \tilde{\omega}^*\|^2. \end{aligned}$$

Since $\frac{\gamma}{2} \|\omega^* - \omega_{\infty}\|^2 \geq 0$, we obtain the result. $\square$

The bound in Proposition 2 has two terms:

- **Regularization bias** ($\gamma \rho^2/2$): The cost of pulling ω^* toward ω_{∞} . The bias is small when the source systems are similar to the target system (ρ small).
- **Optimization error**: This decays exponentially with K_{adapt} .

This formalizes the “few-shot” benefit: when ρ is small, good performance is achievable with few adaptation steps.

(3) Closed-Loop Stability

We finally establish the stability results for Algorithm 1 based on robust MPC theory. The following results apply directly to the robust MPC formulation defined in (25), which incorporates a tightened state constraint and a terminal constraint $s_{t+N|t} \in \mathcal{X}_f$ to ensure recursive feasibility.

Assumption 3. The MPC (20) with terminal constraint $s_{t+N|t} \in \mathcal{X}_f$ satisfies:

- 1) (A_z, B_z) is stabilizable and (A_z, C_z) is detectable.
- 2) $\mathcal{X}_f$ is robustly positively invariant under the local linear stabilizing controller associated with $\mathbb{P}$, for all $\|\hat{y}_{t+k|t} - y_{t+k}\| \leq \bar{e}$.

Assumption 4. There exists $\bar{e} > 0$ such that during closed-loop operation, the model mismatch satisfies $\|\hat{y}_{t+k|t} - y_{t+k}\| \leq \bar{e}$ for $k = 1, \dots, N$ and all $t \geq 0$.

Proposition 3 (Closed-Loop Stability). *Let Assumptions 3 and 4 hold. The closed-loop system under the robust MPC (25) is input-to-state stable (ISS) with respect to the prediction error, and the tracking error is bounded as*

$$\limsup_{t \rightarrow \infty} \|y_t - \bar{y}_t\| \leq \kappa \bar{e},$$

where $\kappa > 0$ depends on $\mathbb{P}$, $\mathbb{Q}$, $\mathbb{R}$, N , and $\mathcal{X}_f$.

Proof Sketch. Under Assumption 3, the terminal constraint $s_{t+N|t} \in \mathcal{X}_f$ paired with the tightened output constraints $\mathcal{Y} \ominus \mathcal{E}$ ensures recursive feasibility for bounded disturbances. Consequently, the optimal MPC value function serves as an ISS-Lyapunov function. The prediction error acts as a bounded additive disturbance propagating over the horizon via C_z . Standard robust tube-based MPC arguments [37], [38] then yield the ISS bound. $\square$

Propositions 2 and 3 yield the following conditional pathway: a small ρ gives a small adaptation loss on the target dataset (Proposition 2); when this translates into a small closed-loop prediction bound $\bar{e}$ (Assumption 4), then Proposition 3 gives tight tracking. This is consistent with Fig. 4, where iMAML achieves lower tracking error than MAML and supervised learning, with the largest advantage during early adaptation.

Limitations and Future Work: The constant $\bar{e}$ in Assumption 4 depends on the covariate shift between $\mathcal{D}_{\text{target}}$ and the closed-loop state distribution; bounding it for NSSM architectures combining a nonlinear encoder with linear latent dynamics is identified as future work [39], [40]. Moreover, the learned dynamics (A_z, B_z, C_z) must satisfy stabilizability and detectability for MPC stability; see Assumption 3.1). We observe this empirically in our experiments; however, enforcing these properties during training remains an important direction for future work. Recent advances in learning stable deep dynamics models, such as [41], provide a promising path forward by jointly learning a dynamics model and a Lyapunov function that guarantees non-expansiveness across the entire state space. Integrating such stability-preserving constraints into our meta-learning framework could further enhance closed-loop guarantees. For DQN (Algorithm 2), rigorous guarantees under neural function approximation remain an open problem, as discussed in Remark 3.

APPENDIX B REPRODUCIBILITY STATEMENT

To ensure the full reproducibility of the experimental results, the complete Python source code, the ball-on-a-plate physical simulation environments, and the pre-trained meta-initialization weights will be made publicly available at GitHub upon the publication of this manuscript (temporarily withheld to strictly maintain anonymity during the double-blind review process of IEEE T-ASE).

The proposed framework is developed within a Python-based software ecosystem. Specifically, the deep RL network and the underlying training pipelines for DQN are constructed using PyTorch. All critical hyperparameters have been exhaustively documented in Section V to facilitate transparent verification and future extensions.

REFERENCES

- [1] T. Dierks and S. Jagannathan, “Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1118–1129, 2012.
- [2] H. Modares, F. L. Lewis, W. Kang, and A. Davoudi, “Optimal synchronization of heterogeneous nonlinear systems with unknown dynamics,” *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 117–131, 2017.
- [3] D. L. Pepyne and C. G. Cassandras, “Optimal control of hybrid systems in manufacturing,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1108–1123, 2000.
- [4] R. Vilalta and Y. Drissi, “A perspective view and survey of meta-learning,” *Artificial intelligence review*, vol. 18, pp. 77–95, 2002.
- [5] J. Cui, N.-M. T. Kokolakis, K. G. Vamvoudakis, and P. A. Vela, “Multi-task transfer learning in trajectory tracking control problems using iterative learning control,” *IEEE Transactions on Artificial Intelligence*, pp. 1–15, 2025.
- [6] Y. Pei, J. Zhao, Y. Yao, and F. Ding, “Multi-task reinforcement learning for distribution system voltage control with topology changes,” *IEEE Transactions on Smart Grid*, vol. 14, no. 3, pp. 2481–2484, 2023.
- [7] 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.
- [8] L. F. Toso, D. Zhan, J. Anderson, and H. Wang, “Meta-learning linear quadratic regulators: a policy gradient maml approach for model-free LQR,” in *6th Annual Learning for Dynamics & Control Conference*. PMLR, 2024, pp. 902–915.

- [9] J. Zhang, C. Wei, C. Luo, and J. Li, "CDAMetaRL: Continuous dynamical adaptive meta reinforcement learning," *IEEE/ASME Transactions on Mechatronics*, pp. 1–11, 2026.
- [10] A. Jena, D. Kalathil, and L. Xie, "Meta-learning-based adaptive stability certificates for dynamical systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 801–12 809.
- [11] S. Chen, Y. Zheng, D. Lin, P. Cai, Y. Xiao, and S. Wang, "MAML-KalmanNet: A neural network-assisted Kalman Filter based on model-agnostic meta-learning," *IEEE Transactions on Signal Processing*, vol. 73, pp. 988–1003, 2025.
- [12] S. Elfeki, R. Zuliani, N. Schmid, E. C. Balta, and J. Lygeros, "Model-agnostic meta learning for differentiable mpc," 2026. [Online]. Available: <https://arxiv.org/abs/2607.19271>
- [13] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [14] J. Yan, A. Chakrabarty, A. Rupenyan, and J. Lygeros, "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.
- [15] S. Lin, H. Wang, and J. Zhang, "System identification via meta-learning in linear time-varying environments," *arXiv preprint arXiv:2010.14664*, 2020.
- [16] A. Chakrabarty, G. Wichern, V. M. Deshpande, A. P. Vinod, K. Berntorp, and C. R. Laughman, "Meta-learning for physically-constrained neural system identification," *arXiv preprint arXiv:2501.06167*, 2025.
- [17] A. Chakrabarty, G. Wichern, and C. R. Laughman, "Meta-learning of neural state-space models using data from similar systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1490–1495, 2023.
- [18] J. Harrison, A. Sharma, R. Calandra, and M. Pavone, "Control adaptation via meta-learning dynamics," in *Workshop on Meta-Learning at NeurIPS*, vol. 2018, 2018.
- [19] A. Ghadirzadeh, X. Chen, P. Poklukur, C. Finn, M. Björkman, and D. Kragic, "Bayesian meta-learning for few-shot policy adaptation across robotic platforms," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1274–1280.
- [20] C. Chen, C. Li, H. Lu, Y. Wang, and R. Xiong, "Meta reinforcement learning of locomotion policy for quadruped robots with motor stuck," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [21] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of hammerstein-wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.
- [22] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [23] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
- [24] F. Dörfler, J. Coulson, and I. Markovsky, "Bridging direct and indirect data-driven control formulations via regularizations and relaxations," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 883–897, 2022.
- [25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:205242740>
- [26] P. Gautier, J. Laurent, and J.-P. Digue, "Deep q-learning-based dynamic management of a robotic cluster," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 4, pp. 2503–2515, 2022.
- [27] F. S. Melo, "Convergence of q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep.*, pp. 1–4, 2001.
- [28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. Pmlr, 2018, pp. 1861–1870.
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [30] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based reinforcement learning: A survey," *Foundations and Trends in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.
- [31] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I*, 3rd ed. Athena Scientific, 2005.
- [32] Z. Hou and S. Jin, *Model free adaptive control: theory and applications*. CRC press, 2013.
- [33] J. Zhou and C. Wen, "Adaptive backstepping control," *Adaptive Backstepping Control of Uncertain Systems: Nonsmooth Nonlinearities, Interactions or Time-Variations*, pp. 9–31, 2008.
- [34] S. Froelich, "Modernization of the ball-on-a-plate system," Bachelor Thesis, ETH Zurich, 2024.
- [35] R. Waldvogel, "MPC control of a ball on plate system," Master Thesis, ETH Zurich, 2010.
- [36] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.
- [37] D. Limon, T. Alamo, D. M. Raimondo, D. M. De La Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho, "Input-to-state stability: a unifying framework for robust model predictive control," in *Nonlinear model predictive control: Towards new challenging applications*. Springer, 2009, pp. 1–26.
- [38] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, Wisconsin: Nob Hill Publishing, 2017.
- [39] H. Mania, M. I. Jordan, and B. Recht, "Active learning for nonlinear system identification with guarantees," *Journal of Machine Learning Research*, vol. 23, no. 32, pp. 1–30, 2022.
- [40] M. Simchowitz, H. Mania, S. Tu, M. I. Jordan, and B. Recht, "Learning without mixing: Towards a sharp analysis of linear system identification," in *Proceedings of the 31st Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, vol. 75, 2018, pp. 439–473.
- [41] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.