Adaptive-Control-Oriented Meta-Learning for Nonlinear Systems

Spencer M. Richards*, Navid Azizan*†, Jean-Jacques Slotine†, and Marco Pavone*
*Department of Aeronautics & Astronautics, Stanford University, California, U.S.A.
†Department of Mechanical Engineering, Massachusetts Institute of Technology, Massachusetts, U.S.A.
Email: *{spenrich,pavone}@stanford.edu, †{azizan,jjs}@mit.edu

***Abstract*—Real-time adaptation is imperative to the control of robots operating in complex, dynamic environments. Adaptive control laws can endow even nonlinear systems with good trajectory tracking performance, provided that any uncertain dynamics terms are linearly parameterizable with known nonlinear features. However, it is often difficult to specify such features a priori, such as for aerodynamic disturbances on rotorcraft or interaction forces between a manipulator arm and various objects. In this paper, we turn to data-driven modeling with neural networks to learn, offline from past data, an adaptive controller with an internal parametric model of these nonlinear features. Our key insight is that we can better prepare the controller for deployment with control-oriented meta-learning of features in closed-loop simulation, rather than regression-oriented meta-learning of features to fit input-output data. Specifically, we meta-learn the adaptive controller with closed-loop tracking simulation as the base-learner and the average tracking error as the meta-objective. With a nonlinear planar rotorcraft subject to wind, we demonstrate that our adaptive controller outperforms other controllers trained with regression-oriented meta-learning when deployed in closed-loop for trajectory tracking control.**

I. Introduction

Performant control in robotics is impeded by the complexity of the dynamical system consisting of the robot itself (i.e., its nonlinear equations of motion) and the interactions with its environment. Roboticists can often derive a physics-based robot model, and then choose from a suite of nonlinear control laws that each offer desirable control-theoretic properties (e.g., good tracking performance) in known, simple environments. Even in the face of model uncertainty, nonlinear control can still yield such properties with the help of *real-time adaptation* to *online* measurements, provided the uncertainty enters the system in a known, structured manner.

However, when a robot is deployed in complex scenarios, it is generally intractable to know even the structure of all possible configurations and interactions that the robot may experience. To address this, system identification and data-driven control seek to learn an accurate input-output model from past measurements. Recent years have also seen a dramatic proliferation of research in machine learning for control by leveraging powerful approximation architectures to predict and optimize the behaviour of dynamical systems. In general, such rich models require extensive data and computation to back-propagate gradients for many layers of parameters, and thus usually cannot be used in fast nonlinear control loops.

Moreover, machine learning of dynamical system models often prioritizes fitting input-output data, i.e., it is *regression-oriented*, with the rationale that designing a controller for a highly accurate model engenders better closed-loop performance on the real system. However, decades of work in system identification and adaptive control recognize that, since a model is often learned for the purpose of control, the learning process itself should be tailored to the *downstream control objective*. This concept of *control-oriented* learning is exemplified by fundamental results in adaptive control theory for linearly parameterizable systems; guarantees on tracking convergence can be attained *without* convergence of the parameter estimates to those of the true system.

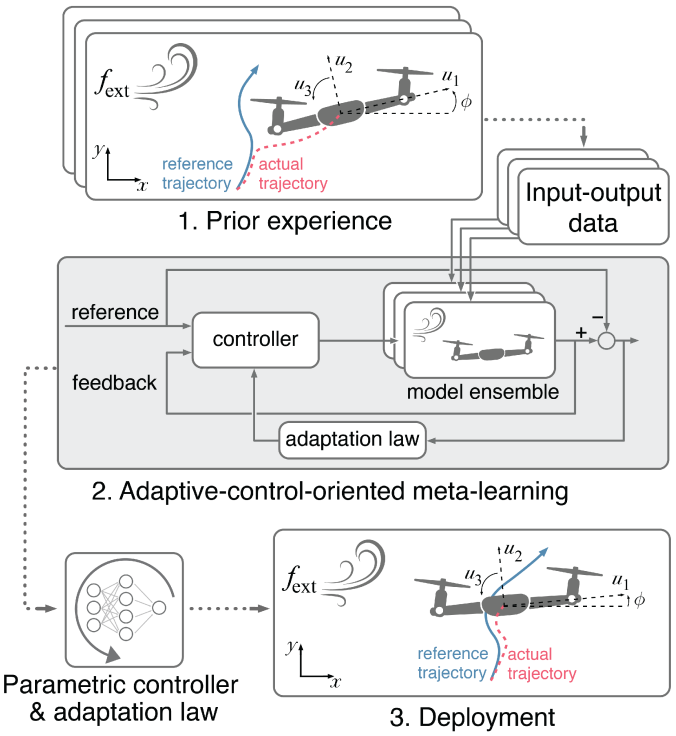

Fig. 1. While roboticists can often derive a model for how control inputs affect the system state, it is much more difficult to model prevalent external forces (e.g., from aerodynamics, contact, and friction) that adversely affect tracking performance. In this work, we present a method to meta-learn an adaptive controller offline from previously collected data. Our meta-learning is *control-oriented* rather than regression-oriented; specifically, we: 1) collect input-output data on the true system, 2) train a parametric adaptive controller in closed-loop simulation to adapt well to each model of an ensemble constructed from past input-output data, and 3) test our adaptive controller on the real system. Our method contextualizes training within the downstream control objective, thereby engendering good tracking results at test time, which we demonstrate on a Planar Fully-Actuated Rotorcraft (PFAR) subject to wind.

Contributions: In this work, we acknowledge this distinction between regression-oriented and control-oriented learning, and propose a control-oriented method to learn a parametric adaptive controller that performs well in closed-loop at test time. Critically, our method (outlined in Figure 1) focuses on *offline* learning from past trajectory data. We formalize training the adaptive controller as a semi-supervised, bi-level meta-learning problem, with the average integrated tracking error across chosen reference trajectories as the meta-objective. We use a closed-loop simulation with our adaptive controller as a base-learner, which we then back-propagate gradients through. We discuss how our formulation can be applied to adaptive controllers for general dynamical systems, then specialize it to the case of nonlinear mechanical systems. Through our experiments, we show that by injecting the downstream control objective into offline meta-learning of an adaptive controller, we improve closed-loop trajectory tracking performance at test time in the presence of widely varying disturbances. We provide code to reproduce our results at https://github.com/StanfordASL/Adaptive-Control-Oriented-Meta-Learning.

II. Related Work

In this section, we review three key areas of work related to this paper: control-oriented system identification, adaptive control, and meta-learning.

A. Control-Oriented System Identification

Learning a system model for the express purpose of closed-loop control has been a hallmark of linear system identification since at least the early 1970s [61]. Due to the sheer amount of literature in this area, we direct readers to Ljung [43] and Gevers [24]. Some salient works are the demonstrations by Skelton [68] on how large open-loop modelling errors do not necessarily cause poor closed-loop prediction, and the theory and practice from Hjalmarsson et al. [30] and Forssell and Ljung [22] for iterative online closed-loop experiments that encourage convergence to a model with optimal closed-loop behaviour. In this paper, we focus on *offline meta-learning* targeting a downstream closed-loop control objective, to train adaptive controllers for *nonlinear* systems.

In nonlinear system identification, there is an emerging body of literature on data-driven, constrained learning for dynamical systems that encourages learned models and controllers to perform well in closed-loop. Khansari-Zadeh and Billard [36] and Medina and Billard [47] train controllers to imitate known invertible dynamical systems while constraining the closed-loop system to be stable. Chang et al. [18] and Sun et al. [73] jointly learn a controller and a stability certificate for known dynamics to encourage good performance in the resulting closed-loop system. Singh et al. [66] jointly learn a dynamics model and a stabilizability certificate that regularizes the model to perform well in closed-loop, even with a controller designed a posteriori. Overall, these works concern learning a fixed model-controller pair. Instead, with offline meta-learning, we train an *adaptive* controller that can update its internal representation of the dynamics online. We discuss future work explicitly incorporating stability constraints in Section VII.

B. Adaptive Control

Broadly speaking, adaptive control concerns parametric controllers paired with an *adaptation law* that dictates how the parameters are adjusted online in response to signals in a dynamical system [71, 51, 41, 32]. Since at least the 1950s, researchers in adaptive control have focused on parameter adaptation prioritizing control performance over parameter identification [7]. Indeed, one of the oldest adaptation laws, the so-called MIT rule, is essentially gradient descent on the integrated squared *tracking* error [46]. Tracking convergence to a reference signal is the primary result in Lyapunov stability analyses of adaptive control designs [52, 53], with parameter convergence as a secondary result if the reference is persistently exciting [5, 14]. In the absence of persistent excitation, Boffi and Slotine [12] show certain adaptive controllers also "implicitly regularize" [8, 9] the learned parameters to have small Euclidean norm; moreover, different forms of implicit regularization (e.g., sparsity-promoting) can be achieved by certain modifications of the adaptation law. Overall, adaptive control prioritizes control performance while learning parameters on a "need-to-know" basis, which is a principle that can be extended to many learning-based control contexts [75].

Stable adaptive control of nonlinear systems often relies on linearly parameterizable dynamics with known nonlinear basis functions, i.e., *features*, and the ability to cancel these nonlinearities stably with the control input when the parameters are known exactly [69, 70, 71, 44]. When such features cannot be derived a priori, function approximators such as neural networks [65, 33, 34] and Gaussian processes [25, 23] can be used and updated online in the adaptive control loop. However, *fast* closed-loop adaptive control with complex function approximators is hindered by the computational effort required to train them; this issue is exacerbated by the practical need for controller gain tuning. In our paper, we focus on offline meta-training of neural network features and controller gains from collected data, with well-known controller structures that can operate in fast closed-loops.

C. Meta-Learning

Meta-learning is the tool we use to inject the downstream adaptive control application into offline learning from data. Informally, meta-learning or "learning to learn" improves knowledge of how to best optimize a given meta-objective across *different* tasks. In the literature, meta-learning has been formalized in various manners; we refer readers to Hospedales et al. [31] for a survey of them. In general, the algorithm chosen to solve a specific task is the *base-learner*, while the algorithm used to optimize the meta-objective is the *meta-learner*. In our work, when trying to make a dynamical system track several reference trajectories, each trajectory is associated with a "task", the adaptive tracking controller is the base-learner, and the average tracking error across all of these trajectories is the meta-objective we want to minimize.

Many works try to meta-learn a dynamics model offline that can best fit new input-output data gathered during a particular

task. That is, the base- and meta-learners are *regression-oriented*. Bertinetto et al. [11] and Lee et al. [42] back-propagate through closed-form ridge regression solutions for few-shot learning, with a maximum likelihood meta-objective. O'Connell et al. [55] apply this same method to learn neural network features for nonlinear mechanical systems. Harrison et al. [28, 27] more generally back-propagate through a Bayesian regression solution to train a Bayesian prior dynamics model with nonlinear features. Nagabandi et al. [50] use a maximum likelihood meta-objective, and gradient descent on a multi-step likelihood objective as the base-learner. Belkhale et al. [10] also use a maximum likelihood meta-objective, albeit with the base-learner as a maximization of the Evidence Lower BOund (ELBO) over parameterized, task-specific variational posteriors; at test time, they perform latent variable inference online in a slow control loop.

Finn et al. [21], Rajeswaran et al. [58], and Clavera et al. [20] meta-train a policy with the expected accumulated reward as the meta-objective, and a policy gradient step as the base-learner. These works are similar to ours in that they infuse offline learning with a more control-oriented flavour. However, while policy gradient methods are amenable to purely data-driven models, they beget slow control-loops due to the sampling and gradients required for each update. Instead, we back-propagate gradients through *offline* closed-loop simulations to train adaptive controllers with well-known designs meant for fast online implementation. This yields a meta-trained adaptive controller that enjoys the performance of principled design inspired by the rich body of control-theoretical literature.

III. Problem Statement

In this paper, we are interested in controlling the continuous-time, nonlinear dynamical system

$$\dot{x} = f(x, u, w), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the control input, and $w(t) \in \mathbb{R}^d$ is some unknown disturbance. Specifically, for a given reference trajectory $r(t) \in \mathbb{R}^n$, we want to choose $u(t)$ such that $x(t)$ converges to $r(t)$; we then say $u(t)$ makes the system (1) track $r(t)$.

Since $w(t)$ is unknown and possibly time-varying, we want to design a feedback law $u = \pi(x, r, a)$ with parameters $a(t)$ that are updated *online* according to a chosen *adaptation law* $\dot{a} = \rho(x, r, a)$. We refer to (π, ρ) together as an *adaptive controller*. For example, consider the control-affine system

$$\dot{x} = f_0(x) + B(x)(u + Y(x)w), \tag{2}$$

where f_0, B, and Y are known, possibly nonlinear maps. A reasonable feedback law choice would be

$$u = \pi_0(x, r) - Y(x)a, \tag{3}$$

where π_0 ensures $\dot{x} = f_0(x) + B(x)\pi_0(x, r)$ tracks $r(t)$, and the term $Y(x)a$ is meant to cancel $Y(x)w$ in (2). For this reason, $Y(x)w$ is termed a matched uncertainty in the literature. If the adaptation law $\dot{a} = \rho(x, r, a)$ is designed such that $Y(x(t))a(t)$ converges to $Y(x(t))w(t)$, then we can use (3) to make (2) track $r(t)$. Critically, this is *not* the same as requiring $a(t)$ to converge to $w(t)$. Since $Y(x)w$ depends on $x(t)$ and hence indirectly on the target $r(t)$, the roles of feedback and adaptation are inextricably linked by the tracking control objective. Overall, learning in adaptive control is done on a "need-to-know" basis to cancel $Y(x)w$ in *closed-loop*, rather than to estimate w in open-loop.

IV. Bi-Level Meta-Learning

We now describe some preliminaries on meta-learning akin to Finn et al. [21] and Rajeswaran et al. [59], so that we can apply these ideas in the next section to the adaptive control problem (1) and in Section VI-A to our baselines.

In machine learning, we typically seek some optimal parameters $\varphi^* \in \arg\min_\varphi \ell(\varphi, \mathcal{D})$, where ℓ is a scalar-valued loss function and $\mathcal{D}$ is some data set. In meta-learning, we instead have a collection of loss functions $\{\ell_i\}_{i=1}^M$, training data sets $\{\mathcal{D}_i^{\text{train}}\}_{i=1}^M$, and evaluation data sets $\{\mathcal{D}_i^{\text{eval}}\}_{i=1}^M$, where each i corresponds to a task. Moreover, during each task i, we can apply an adaptation mechanism $\text{Adapt} : (\theta, \mathcal{D}_i^{\text{train}}) \mapsto \varphi_i$ to map so-called meta-parameters θ and the task-specific training data $\mathcal{D}_i^{\text{train}}$ to task-specific parameters φ_i. The crux of meta-learning is to solve the bi-level problem

$$\begin{aligned} \theta^* \in \arg\min_\theta \frac{1}{M}\left(\sum_{i=1}^M \ell_i(\varphi_i, \mathcal{D}_i^{\text{eval}}) + \mu_{\text{meta}}\|\theta\|_2^2\right), \\ \text{s.t. } \varphi_i = \text{Adapt}(\theta, \mathcal{D}_i^{\text{train}}) \end{aligned} \tag{4}$$

with regularization coefficient $\mu_{\text{meta}} \geq 0$, thereby producing meta-parameters θ^* that are on average well-suited to being adapted for each task. This motivates the moniker "learning to learn" for meta-learning. The optimization (4) is the meta-problem, while the average loss is the meta-loss. The adaptation mechanism Adapt is termed the *base-learner*, while the algorithm used to solve (4) is termed the *meta-learner* [31].

Generally, the meta-learner is chosen to be some gradient descent algorithm. Choosing a good base-learner is an open problem in meta-learning research. Finn et al. [21] propose using a gradient descent step as the base-learner, such that $\varphi_i = \theta - \eta \nabla_\theta \ell_i(\theta, \mathcal{D}_i^{\text{train}})$ in (4) with some learning rate $\eta > 0$. This approach is general in that it can be applied to any differentiable task loss functions. Bertinetto et al. [11] and Lee et al. [42] instead study when the base-learner can be expressed as a convex program with a differentiable closed-form solution. In particular, they consider ridge regression with the hypothesis $\hat{y} = Ag(x; \theta)$, where A is a matrix and $g(x; \theta)$ is some vector of nonlinear features parameterized by θ. For the base-learner, they use

$$\varphi_i = \arg\min_A \sum_{(x,y) \in \mathcal{D}_i^{\text{train}}} \|y - Ag(x; \theta)\|_2^2 + \mu_{\text{ridge}}\|A\|_F^2, \tag{5}$$

with regularization coefficient $\mu_{\text{ridge}} > 0$ for the Frobenius norm $\|A\|_F^2$, which admits a differentiable, closed-form solution. Instead of adapting θ to each task i with a single gradient step, this approach leverages the convexity of ridge regression tasks to minimize the task loss analytically.

V. Adaptive Control as a Base-Learner

We now present the key idea of our paper, which uses meta-learning concepts introduced in Section IV to tackle the problem of learning to control (1). For the moment, we assume we can simulate the dynamics function f in (1) offline and that we have M samples $\{w_j(t)\}_{j=1}^M$ for $t \in [0,T]$ in (1); we will eliminate these assumptions in Section V-B.

A. Meta-Learning from Feedback and Adaptation

In meta-learning vernacular, we treat a reference trajectory $r_i(t) \in \mathbb{R}^n$ and disturbance signal $w_j(t) \in \mathbb{R}^d$ together over some time horizon $T > 0$ as the training data $\mathcal{D}_{ij}^{\text{train}} = \{r_i(t), w_j(t)\}_{t\in[0,T]}$ for task (i,j). We wish to learn the static parameters $\theta := (\theta_\pi, \theta_\rho)$ of an adaptive controller

$$\begin{aligned} u &= \pi(x, r, a; \theta_\pi), \\ \dot{a} &= \rho(x, r, a; \theta_\rho), \end{aligned} \tag{6}$$

such that (π, ρ) engenders good tracking of $r_i(t)$ for $t \in [0,T]$ subject to the disturbance $w_j(t)$. Our adaptation mechanism is the forward-simulation of our closed-loop system, i.e., in (4) we have $\varphi_{ij} = \{x_{ij}(t), a_{ij}(t), u_{ij}(t)\}_{t\in[0,T]}$, where

$$\begin{aligned} x_{ij}(t) &= x_{ij}(0) + \int_0^T f(x_{ij}(t), u_{ij}(t), w_j(t))\, dt, \\ a_{ij}(t) &= a_{ij}(0) + \int_0^T \rho(x_{ij}(t), u_{ij}(t), w_j(t); \theta_\rho)\, dt, \\ u_{ij}(t) &= \pi(x_{ij}(t), r_i(t), a_{ij}(t); \theta_\pi), \end{aligned} \tag{7}$$

which we can compute with one of many Ordinary Differential Equation (ODE) solvers. For simplicity, we always set $x_{ij}(0) = r_i(0)$ and $a_{ij}(0) = 0$. Our task loss is simply the average tracking error for the same reference-disturbance pair, i.e., $\mathcal{D}_{ij}^{\text{eval}} = \{r_i(t), w_j(t)\}_{t\in[0,T]}$ and

$$\ell_{ij}(\varphi_{ij}, \mathcal{D}_{ij}^{\text{eval}}) = \frac{1}{T}\int_0^T \left(\|x_{ij}(t) - r_i(t)\|_2^2 + \alpha\|u_{ij}(t)\|_2^2\right) dt, \tag{8}$$

where $\alpha \geq 0$ regularizes the control effort $\frac{1}{T}\int_0^T \|u_{ij}(t)\|_2^2\, dt$. This loss is inspired by the Linear Quadratic Regulator (LQR) from optimal control, and can be generalized to weighted norms. Assume we construct N reference trajectories $\{r_i(t)\}_{i=1}^N$ and sample M disturbance signals $\{w_j(t)\}_{j=1}^M$, thereby creating NM tasks. Combining (7) and (8) for all (i,j) in the form of (4) then yields the meta-problem

$$\begin{aligned} \min_\theta \ & \frac{1}{NMT}\left(\sum_{i=1}^N\sum_{j=1}^M \int_0^T c_{ij}(t)\, dt + \mu_{\text{meta}}\|\theta\|_2^2\right) \\ \text{s.t.} \ & c_{ij} = \|x_{ij} - r_i\|_2^2 + \alpha\|u_{ij}\|_2^2 \\ & \dot{x}_{ij} = f(x_{ij}, u_{ij}, w_j), \ x_{ij}(0) = r_i(0) \\ & u_{ij} = \pi(x_{ij}, r_i, a_{ij}; \theta_\pi) \\ & \dot{a}_{ij} = \rho(x_{ij}, r_i, a_{ij}; \theta_\rho), \ a_{ij}(0) = 0 \end{aligned} \tag{9}$$

Solving (9) would yield parameters $\theta = (\theta_\pi, \theta_\rho)$ for the adaptive controller (π, ρ) such that it works well on average in closed-loop tracking of $\{r_i(t)\}_{i=1}^N$ for the dynamics f, subject to the disturbances $\{w_j(t)\}_{j=1}^M$. To learn the meta-parameters θ, we can perform gradient descent on (9). This requires back-propagating through an ODE solver, which can be done either directly or via the adjoint state method after solving the ODE forward in time [57, 19, 6, 49]. In addition, the learning problem (9) is *semi-supervised*, in that $\{w_j(t)\}_{j=1}^M$ are labelled samples and $\{r_i(t)\}_{i=1}^N$ can be chosen freely. If there are some specific reference trajectories we want to track at test time, we can use them in the meta-problem (9). This is an advantage of designing the offline learning problem in the context of the downstream control objective.

B. Model Ensembling as a Proxy for Feedback Offline

In practice, we cannot simulate the true dynamics f or sample an actual disturbance trajectory $w(t)$ offline. Instead, we can more reasonably assume we have past data collected with some other, possibly poorly tuned controller. In particular, we make the following assumptions:

- We have access to trajectory data $\{\mathcal{T}_j\}_{j=1}^M$, such that

$$\mathcal{T}_j = \left\{\left(t_k^{(j)}, x_k^{(j)}, u_k^{(j)}, t_{k+1}^{(j)}, x_{k+1}^{(j)}\right)\right\}_{k=0}^{N_j-1}, \tag{10}$$

  where $x_k^{(j)} \in \mathbb{R}^n$ and $u_k^{(j)} \in \mathbb{R}^m$ were the state and control input, respectively, at time $t_k^{(j)}$. Moreover, $u_k^{(j)}$ was applied in a zero-order hold over $[t_k^{(j)}, t_{k+1}^{(j)})$, i.e., $u(t) = u(t_k^{(j)})$ for all $t \in [t_k^{(j)}, t_{k+1}^{(j)})$ along each trajectory $\mathcal{T}_j$.
- During the collection of trajectory data $\mathcal{T}_j$, the disturbance $w(t)$ took on a fixed, unknown value w_j.

The second point is inspired by both meta-learning literature, where it is usually assumed the training data can be segmented according to the latent task, and adaptive control literature, where it is usually assumed that any unknown parameters are constant or slowly time-varying. These assumptions can be generalized to any collection of measured time-state-control transition tuples that can be segmented according to some latent task; in (10) we consider when such tuples can be grouped into trajectories, since this is a natural manner in which data is collected from dynamical systems.

Inspired by Clavera et al. [20], since we cannot simulate the true dynamics f offline, we propose to first train a *model ensemble* from the trajectory data $\{\mathcal{T}_j\}_{j=1}^M$ to roughly capture the distribution of $f(\cdot,\cdot,w)$ over possible values of the disturbance w. Specifically, we fit a model $\hat{f}_j(x,u;\psi_j)$ with parameters ψ_j to each trajectory $\mathcal{T}_j$, and use this as a proxy for $f(x,u,w_j)$ in (9). The meta-problem (9) is now

$$\begin{aligned} \min_\theta \ & \frac{1}{NMT}\left(\sum_{i=1}^N\sum_{j=1}^M \int_0^T c_{ij}(t)\, dt + \mu_{\text{meta}}\|\theta\|_2^2\right) \\ \text{s.t.} \ & c_{ij} = \|x_{ij} - r_i\|_2^2 + \alpha\|u_{ij}\|_2^2 \\ & \dot{x}_{ij} = \hat{f}_j(x_{ij}, u_{ij}; \psi_j), \ x_{ij}(0) = r_i(0) \\ & u_{ij} = \pi(x_{ij}, r_i, a_{ij}; \theta_\pi) \\ & \dot{a}_{ij} = \rho(x_{ij}, r_i, a_{ij}; \theta_\rho), \ a_{ij}(0) = 0 \end{aligned} \tag{11}$$

This form is still semi-supervised, since each model $\hat{f}_j$ is dependent on the trajectory data $\mathcal{T}_j$, while $\{r_i\}_{i=1}^N$ can be

chosen freely. The collection $\{\hat{f}_j\}_{j=1}^M$ is termed a model ensemble. Empirically, the use of model ensembles has been shown to improve robustness to model bias and train-test data shift of deep predictive models [40] and policies in reinforcement learning [58, 39, 20]. To train the parameters ψ_j of model $\hat{f}_j$ on the trajectory data $\mathcal{T}_j$, we do gradient descent on the one-step prediction problem

$$\begin{aligned} \min_{\psi_j} \frac{1}{N_j} \left(\sum_{k=0}^{N_j-1} \left\| x_{k+1}^{(j)} - \hat{x}_{k+1}^{(j)} \right\|_2^2 + \mu_{\text{ensem}} \|\psi_j\|_2^2 \right) \\ \text{s.t. } \hat{x}_{k+1}^{(j)} = x_k^{(j)} + \int_{t_k^{(j)}}^{t_{k+1}^{(j)}} \hat{f}_j(x(t), u_k^{(j)}; \psi_j)\, dt \end{aligned} \tag{12}$$

where $\mu_{\text{ensem}} > 0$ regularizes ψ_j. Since we meta-train θ in (12) to be adaptable to every model in the ensemble, we only need to roughly characterize how the dynamics $f(\cdot, \cdot, w)$ vary with the disturbance w, rather than do exact model fitting of $\hat{f}_j$ to $\mathcal{T}_j$. Thus, we approximate the integral in (12) with a single step of a chosen ODE integration scheme and back-propagate through this step, rather than use a full pass of an ODE solver.

C. Incorporating Prior Knowledge About Robot Dynamics

So far our method has been agnostic to the choice of adaptive controller (π, ρ). However, if we have some prior knowledge of the dynamical system (1), we can use this to make a good choice of structure for (π, ρ). Specifically, we now consider the large class of *Lagrangian dynamical systems*, which includes robots such as manipulator arms and drones. The state of such a system is $x \coloneqq (q, \dot{q})$, where $q(t) \in \mathbb{R}^{n_q}$ is the vector of generalized coordinates completely describing the configuration of the system at time $t \in \mathbb{R}$. The nonlinear dynamics of such systems are fully described by

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = f_{\text{ext}}(q, \dot{q}) + \tau(u), \tag{13}$$

where $H(q)$ is the positive-definite inertia matrix, $C(q, \dot{q})$ is the Coriolis matrix, $g(q)$ is the potential force, $\tau(u)$ is the generalized input force, and $f_{\text{ext}}(q, \dot{q})$ summarizes any other external generalized forces. The vector $C(q, \dot{q})\dot{q}$ is uniquely defined, and the matrix $C(q, \dot{q})$ can always be chosen such that $\dot{H}(q, \dot{q}) - 2C(q, \dot{q})$ is skew-symmetric [71]. Slotine and Li [69] studied adaptive control for (13) under the assumptions:

- The system (13) is fully-actuated, i.e., $u(t) \in \mathbb{R}^{n_q}$ and $\tau : \mathbb{R}^{n_q} \to \mathbb{R}^{n_q}$ is invertible.
- The dynamics in (13) are linearly parameterizable, i.e.,

$$H(q)\dot{v} + C(q, \dot{q})v + g(q) - f_{\text{ext}}(q, \dot{q}) = Y(q, \dot{q}, v, \dot{v})a, \tag{14}$$

  for some known matrix $Y(q, \dot{q}, v, \dot{v}) \in \mathbb{R}^{n_q \times p}$, any vectors $q, \dot{q}, v, \dot{v} \in \mathbb{R}^{n_q}$, and constant unknown parameters $a \in \mathbb{R}^p$.
- The reference trajectory for $x \coloneqq (q, \dot{q})$ is of the form $r = (q_d, \dot{q}_d)$, where q_d is twice-differentiable.

Under these assumptions, the adaptive controller

$$\begin{aligned} &\tilde{q} \coloneqq q - q_d, \;\; s \coloneqq \dot{\tilde{q}} + \Lambda \tilde{q}, \;\; v \coloneqq \dot{q}_d - \Lambda \tilde{q}, \\ &u = \tau^{-1}(Y(q, \dot{q}, v, \dot{v})\hat{a} - Ks), \\ &\dot{\hat{a}} = -\Gamma Y(q, \dot{q}, v, \dot{v})^\mathsf{T} s, \end{aligned} \tag{15}$$

ensures $x(t) = (q(t), \dot{q}(t))$ converges to $r(t) = (q_d(t), \dot{q}_d(t))$, where (Λ, K, Γ) are chosen positive-definite gain matrices.

The adaptive controller (3) requires the nonlinearities in the dynamics (13) to be known a priori. While Niemeyer and Slotine [54] showed these can be systematically derived for $H(q)$, $C(q, \dot{q})$, and $g(q)$, there exist many external forces $f_{\text{ext}}(q, \dot{q})$ of practical importance in robotics for which this is difficult to do, such as aerodynamic and contact forces. Thus, we consider the case when $H(q)$, $C(q, \dot{q})$, and $g(q)$ are known and $f_{\text{ext}}(q, \dot{q})$ is unknown. Moreover, we want to approximate $f_{\text{ext}}(q, \dot{q})$ with the neural network

$$\hat{f}_{\text{ext}}(q, \dot{q}; A, \theta_y) = Ay(q, \dot{q}; \theta_y), \tag{16}$$

where $y(q, \dot{q}; \theta) \in \mathbb{R}^p$ consists of all the hidden layers of the network parameterized by θ_y, and $A \in \mathbb{R}^{n_q \times p}$ is the output layer. Inspired by (15), we consider the adaptive controller

$$\begin{aligned} &\tilde{q} \coloneqq q - q_d, \;\; s \coloneqq \dot{\tilde{q}} + \Lambda \tilde{q}, \;\; v \coloneqq \dot{q}_d - \Lambda \tilde{q}, \\ &u = \tau^{-1}(H(q)\dot{v} + C(q, \dot{q})v + g(q) - Ay(q, \dot{q}; \theta_y) - Ks), \\ &\dot{A} = \Gamma s y(q, \dot{q}; \theta_y)^\mathsf{T}, \end{aligned} \tag{17}$$

If $f_{\text{ext}}(q, \dot{q}) = \hat{f}_{\text{ext}}(q, \dot{q}; A, \theta_y)$ for fixed values θ_y and A, then the adaptive controller (17) guarantees tracking convergence [69]. In general, we do not know such a value for θ_y, and we must choose the gains (Λ, K, Γ). Since (17) is parameterized by $\theta \coloneqq (\theta_y, \Lambda, K, \Gamma)$, we can train (17) with the method described in Sections V-A–V-B. While for simplicity we consider known $H(q)$, $C(q, \dot{q})$, and $g(q)$, we can extend to the case when they are linearly parameterizable, e.g., $H(q)\dot{v} + C(q, \dot{q})v + g(q) = Y(q, \dot{q}, v, \dot{v})a$ with $Y(q, \dot{q}, v, \dot{v})$ a matrix of known features systematically computed as by Niemeyer and Slotine [54]. In this case, we would then maintain a separate adaptation law $\dot{\hat{a}} = -PY(q, \dot{q}, v, \dot{v})^\mathsf{T} s$ with adaptation gain $P \succ 0$ in our proposed adaptive controller (17).

VI. Experiments

We evaluate our method in simulation on a Planar Fully-Actuated Rotorcraft (PFAR) with degrees of freedom $q \coloneqq (x, y, \phi)$ governed by the nonlinear equations of motion

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\phi} \end{pmatrix} + g = \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{=: R(\phi)} u + f_{\text{ext}}, \tag{18}$$

where (x, y) is the position of the center of mass in the inertial frame, ϕ is the roll angle, $g = (0, 9.81, 0)$ m/s^2 is the gravitational acceleration in vector form, $R(\phi)$ is a rotation matrix, f_{ext} is some unknown external force, and $u = (u_1, u_2, u_3)$ are the normalized thrust along the body x-axis, thrust along the body y-axis, and torque about the center of mass, respectively. We depict an exemplary PFAR design in Figure 1 inspired by thriving interest in fully- and over-actuated multirotor vehicles in the robotics literature [64, 35, 16, 77, 60]. The simplified system in (18) is a fully-actuated variant of the classic Planar Vertical Take-Off and Landing (PVTOL) vehicle [29]. In our simulations, f_{ext} is a mass-normalized quadratic drag force,

due to the velocity of the PFAR relative to wind blowing at a velocity $w \in \mathbb{R}$ along the inertial x-axis. Specifically,

$$\begin{aligned} v_1 &= (\dot{x} - w)\cos\phi + \dot{y}\sin\phi, \\ v_2 &= -(\dot{x} - w)\sin\phi + \dot{y}\cos\phi, \\ f_{\text{ext}} &= -\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \beta_1 v_1 |v_1| \\ \beta_2 v_2 |v_2| \end{pmatrix}, \end{aligned} \tag{19}$$

where $\beta_1, \beta_2 > 0$ are aggregate coefficients; we use $\beta_1 = 0.1$ and $\beta_2 = 1$ in all of our simulations.

A. Baselines

We compare our meta-trained adaptive controller in trajectory tracking tasks for (18) against two baseline controllers:

PID Control: Our simplest baseline is a Proportional-Integral-Derivative (PID) controller with feed-forward, i.e.,

$$u = R(\phi)^{\mathsf{T}}\left(g + \ddot{q}_d - K_P\tilde{q} - K_I\int_0^t \tilde{q}(\xi)\,d\xi - K_D\dot{\tilde{q}}\right), \tag{20}$$

with gains $K_P, K_I, K_D \succ 0$. For $f_{\text{ext}}(q, \dot{q}) \equiv 0$, this controller feedback-linearizes the dynamics (18) such that the error $\tilde{q} := q - q_d$ is governed by an exponentially stable ODE. The integral term must compensate for $f_{\text{ext}}(q, \dot{q}) \not\equiv 0$.

Adaptive Control via Meta-Ridge Regression (ACMRR): This baseline is a slightly modified[1] version of the approach taken by O'Connell et al. [55], which applies the work on using ridge regression as a base-learner from Bertinetto et al. [11], Lee et al. [42] and Harrison et al. [28] to learn the parametric features $y(q, \dot{q}; \theta_y)$. Specifically, for a given trajectory $\mathcal{T}_j$ and these features, the last layer A is specified as the best ridge regression fit to some subset of points in $\mathcal{T}_j$. The features $y(q, \dot{q}; \theta_y)$ are then used in the adaptive controller (17).

To clarify, we describe ACMRR with the meta-learning language from Section IV, specifically for the PFAR dynamics in (18). Let $\mathcal{K}_j^{\text{ridge}} \subset \{k\}_{k=0}^{|\mathcal{T}_j|-1}$ denote the indices of transition tuples in some subset of $\mathcal{T}_j$. The Euler approximation

$$\hat{\dot{q}}_{k+1}^{(j)}(A) := \dot{q}_k^{(j)} + \Delta t_k^{(j)}\big(R(\phi_k^{(j)})u_k^{(j)} + A y(q_k^{(j)}, \dot{q}_k^{(j)}; \theta_y)\big) \tag{21}$$

with $\Delta t_k^{(j)} := t_{k+1}^{(j)} - t_k^{(j)}$ is used in the task loss

$$\ell_j(A_j, \mathcal{T}_j) = \frac{1}{|\mathcal{T}_j|}\sum_{k=0}^{|\mathcal{T}_j|-1} \|\dot{q}_{k+1}^{(j)} - \hat{\dot{q}}_{k+1}^{(j)}(A_j)\|_2^2 \tag{22}$$

alongside the adaptation mechanism

$$A_j = \arg\min_A \sum_{k\in\mathcal{K}_j^{\text{ridge}}} \|\dot{q}_{k+1}^{(j)} - \hat{\dot{q}}_{k+1}^{(j)}(A)\|_2^2 + \mu_{\text{ridge}}\|A\|_F^2. \tag{23}$$

The adaptation mechanism (23) can be solved and differentiated in closed-form for any $\mu_{\text{ridge}} > 0$ via the normal equations, since $\hat{\dot{q}}_{k+1}^{(j)}(A)$ is linear in A; indeed, only *linear* integration schemes can be substituted into (21). The meta-problem for ACMRR takes the form of (4) with meta-parameters θ_y for the features $y(q, \dot{q}; \theta_y)$, the task loss (22), and the adaptation mechanism (23). The meta-parameters θ_y are trained via gradient descent on this meta-problem, and then deployed online in the adaptive controller (17).

ACMRR suffers from a *fundamental mismatch* between its regression-oriented meta-problem and the online problem of adaptive trajectory tracking control. The ridge regression base-learner (23) suggests that A should best fit the input-output trajectory data in a regression sense. However, as we mentioned in Section III, adaptive controllers such as the one in (17) learn on a "need-to-know" basis for the primary purpose of control rather than regression. As we discuss in Section VI-D, since our control-oriented approach uses a meta-objective indicative of the downstream closed-loop tracking control objective, we achieve better tracking performance than this baseline at test time.

B. Data Generation and Training

To train the meta-parameters $\theta_{\text{ours}} := (\theta_y, \Lambda, K, \Gamma)$ in our method, and the meta-parameters $\theta_{\text{ACMRR}} := \theta_y$ in the ACMRR baseline, we require trajectory data $\{\mathcal{T}_j\}_{j=1}^M$ of the form (10). To this end, we synthesize $\mathcal{T}_j$ as follows:

1) Generate a uniform random walk of six (x, y, ϕ) points.
2) Fit a 30-second, smooth, polynomial spline trajectory $q_d(t) \in \mathbb{R}^3$ to the random walk with minimum snap in (x, y) and minimum acceleration in ϕ, according to the work by Mellinger and Kumar [48] and Richter et al. [63].
3) Sample a wind velocity $w \in \mathbb{R}$ from the training distribution in Figure 2, and simulate the dynamics (18) with the external force (19) and the Proportional-Derivative (PD) tracking controller

$$u = R(\phi)^{\mathsf{T}}(g - k_P(q - q_r) - k_D(\dot{q} - \dot{q}_r)) \tag{24}$$

in a zero-order hold at 100 Hz, with $k_P = 10$ and $k_D = 0.1$. This controller represents a "first try" at controlling the system (18) in order to collect data. We record time, state, and control input measurements at 100 Hz.

We record 500 such trajectories and then sample M of them to form the training data $\{\mathcal{T}_j\}_{j=1}^M$ for various M to evaluate the sample efficiency of our method and the ACMRR baseline. We present hyperparameter choices and training details for each method in Appendix A. We highlight here that:

- To train the positive-definite gains (Λ, K, Γ) via gradient-based optimization in our method, we use an unconstrained log-Cholesky parameterization for each gain[2].
- To compute the integral in the meta-problem (11) for our method, we use a fourth-order Runge-Kutta scheme with a fixed time step of 0.01 s. We back-propagate gradients

[1] Unlike O'Connell et al. [55], we do not assume access to direct measurements of the external force f_{ext}. Also, they use a more complex form of (15) with better *parameter estimation* properties when the dynamics are linearly parameterizable with *known* nonlinear features [70]. While we could use a parametric form of this controller in place of (17), we forgo this in favour of a simpler presentation, since we focus on offline *control-oriented* meta-learning of *approximate* features.

[2] Any $n \times n$ positive-definite matrix Q can be uniquely defined by $\frac{1}{2}n(n+1)$ parameters. For $n = 2$, the log-Cholesky parameterization of Q is $Q = LL^{\mathsf{T}}$ with $L = \begin{bmatrix} \exp(\theta_1) & 0 \\ \theta_2 & \exp(\theta_3) \end{bmatrix}$ and unconstrained parameters $\theta \in \mathbb{R}^3$ [56].

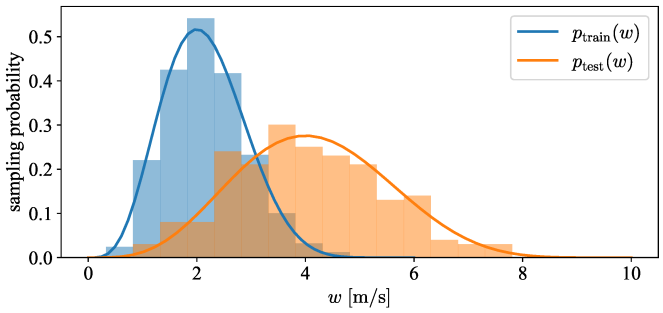

Fig. 2. Training distribution p_{train} and test distribution p_{test} for the wind velocity w along the inertial x-axis. Both are scaled beta distributions; p_{train} is supported on the interval $[0, 6]$ m/s with shape parameters $(\alpha, \beta) = (5, 9)$, while p_{test} is supported on the interval $[0, 10]$ m/s with shape parameters $(\alpha, \beta) = (5, 7)$. The normalized histograms show the distribution of the actual wind velocity samples in the training and test data for a single seed, and highlight the out-of-distribution samples (i.e., relative to p_{train}) that occur during testing.

through this computation in a manner similar to Zhuang et al. [78], rather than using the adjoint method for neural ODEs [19], due to our observation that the backward pass is sensitive to any numerical error accumulated along the forward pass during closed-loop control simulations.

C. Testing with Distribution Shift

For testing, we simulate tracking of 200 smooth, 10-second reference trajectories *different* from those in the training data, using our meta-parameters $\theta_{\text{ours}} := (\theta_y, \Lambda, K, \Gamma)$ with the adaptive controller (17), the ACMRR meta-parameters $\theta_{\text{ACMRR}} := \theta_y$ with the adaptive controller (17), and the PID controller (20); each controller is implemented at 100 Hz. As shown in Figure 2, we sample wind velocities at test time from a distribution *different* from that used for the training data $\{\mathcal{T}_j\}_{j=1}^{M}$; in particular, the test distribution has a higher mode and larger support than the training distribution, thereby producing so-called out-of-distribution wind velocities at test time. This ensures the generalizability of each approach is tested, which is a particularly important concept in meta-learning literature [31].

A strength of our method is that it meta-learns both features $y(q, \dot{q}; \theta)$ and gains (Λ, K, Γ) offline, while both the PID and ACMRR baselines require gain tuning in practice by interacting with the real system. However, for the sake of comparison, we test every method with various manually chosen gains and our method with our meta-learned gains, on the same set of test trajectories. In particular, for the PID controller (20), we set $K_P = K\Lambda + \Gamma$, $K_I = \Gamma\Lambda$, and $K_D = K + \Lambda$, which makes it equivalent to the adaptive controller (17) for the dynamics (18) with $y(q, \dot{q}; \theta_y) \equiv 1$ (i.e., constant features), $\tilde{q}(0) = 0$ (i.e., zero initial position error), and $A(0) = 0$ (i.e., adapted parameters are initially zero). We always set initial conditions for the system such that $\tilde{q}(0) = \dot{\tilde{q}}(0) = 0$, and $A(0) = 0$.

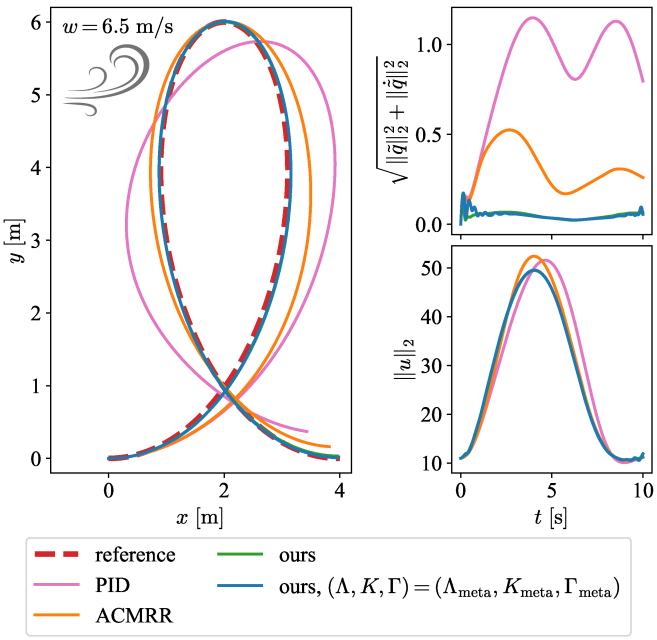

Fig. 3. Tracking results for the PFAR on a test trajectory with $w = 6.5$ m/s, $M = 10$, and $(\Lambda, K, \Gamma) = (I, 10I, 10I)$. We also apply our meta-learned features and our meta-learned gains $(\Lambda_{\text{meta}}, K_{\text{meta}}, \Gamma_{\text{meta}})$. All of the methods expend similar control efforts. However, with our meta-learned features, the tracking error $\|x - r\|_2 = \sqrt{\|\tilde{q}\|_2^2 + \|\dot{\tilde{q}}\|_2^2}$ (where x is overloaded to denote both position $x \in \mathbb{R}$ and state $x = (q, \dot{q})$) quickly decays after a short transient, while the effect of the wind pushing the vehicle to the right is more pronounced for the baselines.

D. Results and Discussion

We first provide a qualitative plot of tracking results for each method on a single test trajectory in Figure 3, which clearly shows that our meta-learned features induce better tracking results than the baselines, while requiring a similar expenditure of control effort. For our method, the initial transient decays quickly, thereby demonstrating *fast* adaptation and the potential to handle even time-varying disturbances.

For a more thorough analysis, we consider the Root-Mean-Squared (RMS) tracking error and control effort for each test trajectory $\{r_i\}_{i=1}^{N_{\text{test}}}$; for any vector-valued signal $h(t)$ and sampling times $\{t_k\}_{k=0}^{N}$, we define

$$\text{RMS}(h) := \sqrt{\frac{1}{N} \sum_{k=0}^{N} \|h(t_k)\|_2^2}. \tag{25}$$

We are interested in $\text{RMS}(x_i - r_i)$ and $\text{RMS}(u_i)$, where $x_i(t) = (q_i(t), \dot{q}_i(t))$ and $u_i(t)$ are the resultant state and control trajectories from tracking $r_i(t) = (q_{d,i}(t), \dot{q}_{d,i}(t))$. In Figure 4, we plot the averages $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \text{RMS}(x_i - r_i)$ and $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \text{RMS}(u_i)$ across $N_{\text{test}} = 200$ test trajectories for each method. For our method and the ACMRR baseline, we vary the number of training trajectories M, and thus the number of wind velocities from the training distribution in Figure 2 implicitly present in the training data. From Figure 4, we observe the PID controller generally yields the highest tracking error, thereby indicating the utility of meta-learning

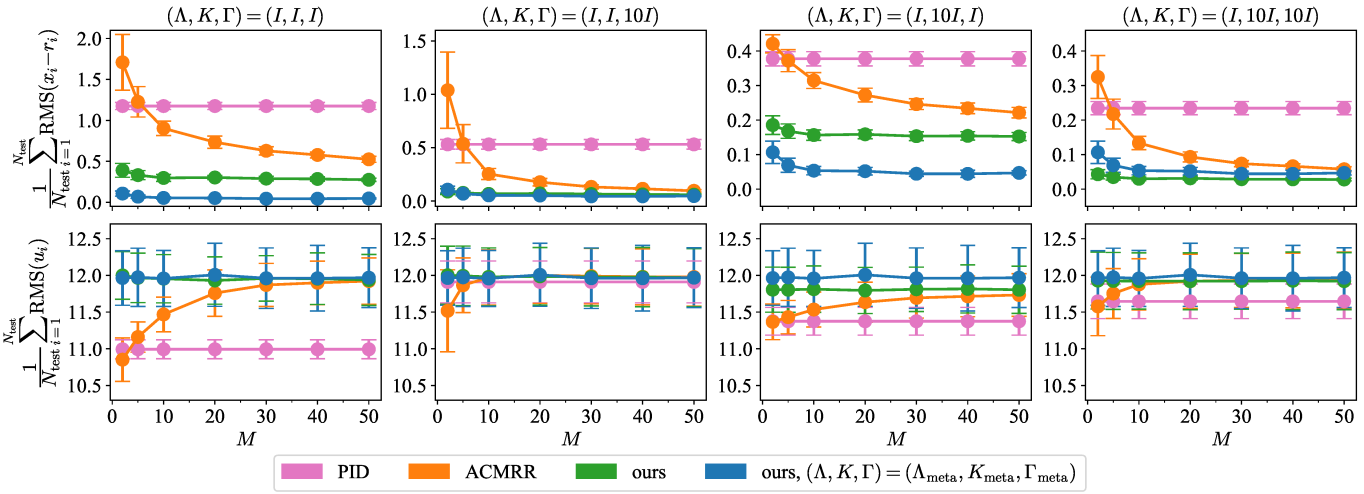

Fig. 4. Line plots of the average RMS tracking error $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \text{RMS}(x_i - r_i)$ and average control effort $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \text{RMS}(u_i)$ across $N_{\text{test}} = 200$ test trajectories versus the number of trajectories $M \in \{2, 5, 10, 20, 30, 40, 50\}$ in the training data. For each method, we try out various control gains (Λ, K, Γ). With our method, we also use our meta-learned gains $(\Lambda_{\text{meta}}, K_{\text{meta}}, \Gamma_{\text{meta}})$. The results for the PID controller do not vary with M since it does not require any training. Dots and error bars denote means and standard deviations, respectively, across 10 random seeds. Our experiments are done in Python using NumPy [26] and JAX [15]. We use the explicit nature of Pseudo-Random Number Generation (PRNG) in JAX to set a seed prior to training, and then methodically branch off the associated PRNG key as required throughout the train-test experiment pipeline. Thus, all of our results can be easily reproduced; code to do so is provided at https://github.com/StanfordASL/Adaptive-Control-Oriented-Meta-Learning.

features $y(q, \dot{q}; \theta_y)$ to better compensate for $f_{\text{ext}}(q, \dot{q})$. We further observe in Figure 4 that, regardless of the control gains, using our features $y(q, \dot{q}; \theta_y)$ in the adaptive controller (17) yields the lowest tracking error. Moreover, using our features with our meta-learned gains yields the lowest tracking error in all but one case. Our features induce a slightly higher control effort with a greater standard deviation across random seeds, especially when used with our meta-learned gains $(\Lambda_{\text{meta}}, K_{\text{meta}}, \Gamma_{\text{meta}})$. This is most likely since the controller can better match the disturbance $f_{\text{ext}}(q, \dot{q})$ with our features, and is an acceptable trade-off given that our primary objective is trajectory tracking. In addition, when using our features with manually chosen or our meta-learned gains, the tracking error remains relatively constant over M; conversely, the tracking error for the ACMRR baseline is higher for lower M, and only reaches the performance of our method with certain gains for large M. Overall, our results indicate:

- The features $y(q, \dot{q}; \theta_y)$ meta-learned by our control-oriented method are *better conditioned for closed-loop tracking control* across a range of controller gains, particularly in the face of a distributional shift between training and test scenarios.
- The gains meta-learned by our method are competitive *without manual tuning*, and thus can be deployed immediately or serve as a good initialization for further fine-tuning.
- Our control-oriented meta-learning method is *sample-efficient* with respect to how much system variability is implicitly captured in the training data.

We again stress these comparisons could only be done by tuning the control gains for the baselines, which in practice would require interaction with the real system and hence further data collection. Thus, the fact that our control-oriented method can meta-learn good control gains *offline* is a key advantage over regression-oriented meta-learning.

VII. Conclusions & Future Work

In this work, we formalized control-oriented meta-learning of adaptive controllers for nonlinear dynamical systems, offline from trajectory data. The procedure we presented is general and uses adaptive control as the base-learner to attune learning to the downstream control objective. We then specialized our procedure to the class of nonlinear *mechanical* systems, with a well-known adaptive controller parameterized by control gains and nonlinear model features. We demonstrated that our control-oriented meta-learning method engenders better closed-loop tracking control performance at test time than when learning is done for the purpose of model regression.

There are a number of exciting future directions for this work. In particular, we are interested in control-oriented meta-learning with *constraints*, such as for adaptive Model Predictive Control (MPC) [1, 17, 72, 38, 67] with state and input constraints. Back-propagating through such a controller would leverage recent work on differentiable convex optimization [4, 2, 3]. We could also back-propagate through parameter constraints; for example, physical consistency of adapted inertial parameters can be enforced as Linear Matrix Inequality (LMI) constraints that reduce overfitting and improve parameter convergence [76]. In addition, we want to extend our meta-learning approach to adaptive control for underactuated systems. Underactuation is a fundamental challenge for adaptive control since model uncertainties must satisfy certain matching conditions so that they can be cancelled stably by the controller [44, 67]. To this end, we want to explore how meta-learning can be used to learn adaptive controllers defined in part by parametric stability and stabilizability certificates, such as Lyapunov functions and Control Contraction Metrics (CCMs) [45]. This could build off of existing work on learning such certificates from data [62, 66, 13, 74].

Acknowledgements

We thank Masha Itkina for her invaluable feedback, and Matteo Zallio for his expertise in crafting Figure 1. This research was supported in part by the National Science Foundation (NSF) via Cyber-Physical Systems (CPS) award #1931815 and Energy, Power, Control, and Networks (EPCN) award #1809314, and the National Aeronautics and Space Administration (NASA) University Leadership Initiative via grant #80NSSC20M0163. Spencer M. Richards was also supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). This article solely reflects our own opinions and conclusions, and not those of any NSF, NASA, or NSERC entity.

References

[1] V. Adetola and M. Guay. Robust adaptive MPC for constrained uncertain nonlinear systems. *Int. Journal of Adaptive Control and Signal Processing*, 25(2):155–167, 2011.

[2] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a conic program. *Journal of Applied and Numerical Optimization*, 1(2):107–115, 2019.

[3] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato. Learning convex optimization control policies. In *Learning for Dynamics & Control*, 2020.

[4] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable MPC for end-to-end planning and control. In *Conf. on Neural Information Processing Systems*, 2018.

[5] B. D. O. Anderson and C. R. Johnson, Jr. Exponential convergence of adaptive identification and control algorithms. *Automatica*, 18(1):1–13, 1982.

[6] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[7] J. Aseltine, A. Mancini, and C. Sarture. A survey of adaptive control systems. *IRE Transactions on Automatic Control*, 6(1):102–108, 1958.

[8] N. Azizan and B. Hassibi. Stochastic gradient/mirror descent: Minimax optimality and implicit regularization. In *Int. Conf. on Learning Representations*, 2019.

[9] N. Azizan, S. Lale, and B. Hassibi. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. In *Int. Conf. on Machine Learning - Workshop on Generalization*, 2019.

[10] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robotics and Automation Letters*, 2021. In press.

[11] L. Bertinetto, J. Henriques, P. H. S. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *Int. Conf. on Learning Representations*, 2019.

[12] N. M. Boffi and J.-J. E. Slotine. Implicit regularization and momentum algorithms in nonlinearly parameterized adaptive control and prediction. *Neural Computation*, 33 (3):590–673, 2021.

[13] N. M. Boffi, S. Tu, N. Matni, J.-J. E. Slotine, and V. Sindhwani. Learning stability certificates from data. In *Conf. on Robot Learning*, 2020.

[14] S. Boyd and S. S. Sastry. Necessary and sufficient conditions for parameter convergence in adaptive control. *Automatica*, 22(6):629–639, 1986.

[15] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. Available at http://github.com/google/jax.

[16] D. Brescianini and R. D'Andrea. Computationally efficient trajectory generation for fully actuated multirotor vehicles. *IEEE Transactions on Robotics*, 34(3):555–571, 2018.

[17] M. Bujarbaruah, X. Zhang, U. Rosolia, and R. Borrelli. Adaptive MPC for iterative tasks. In *Proc. IEEE Conf. on Decision and Control*, 2018.

[18] Y.-C. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. In *Conf. on Neural Information Processing Systems*, 2019.

[19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *Conf. on Neural Information Processing Systems*, 2018.

[20] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conf. on Robot Learning*, 2018.

[21] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Int. Conf. on Machine Learning*, 2017.

[22] U. Forssell and L. Ljung. Some results on optimal experiment design. *Automatica*, 36(5):749–756, 2000.

[23] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. A. Theodorou. $\mathcal{L}_1$-$\mathcal{GP}$: $\mathcal{L}_1$ adaptive control with Bayesian learning. In *Learning for Dynamics & Control*, 2020.

[24] M. Gevers. Identification for control: From the early achievements to the revival of experiment design. *European Journal of Control*, 11(4–5):335–352, 2005.

[25] R. C. Grande, G. Chowdhary, and J. P. How. Nonparametric adaptive control using Gaussian processes with online hyperparameter estimation. In *Proc. IEEE Conf. on Decision and Control*, 2013.

[26] C. R. Harris, K. J. Millman, S. J. Van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. Van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585 (7825):357–362, 2020.

[27] J. Harrison, A. Sharma, R. Calandra, and M. Pavone. Control adaptation via meta-learning dynamics. In *Conf. on Neural Information Processing Systems - Workshop on Meta-Learning*, 2018.

[28] J. Harrison, A. Sharma, and M. Pavone. Meta-learning priors for efficient online bayesian regression. In *Workshop on Algorithmic Foundations of Robotics*, 2018.

[29] J. Hauser, S. Sastry, and G. Meyer. Nonlinear control design for slightly non-minimum phase systems: Application to V/STOL aircraft. *Automatica*, 28(4):665–679, 1992.

[30] H. Hjalmarsson, M. Gevers, and F. de Bruyne. For model-based control design, closed-loop identification gives better performance. *Automatica*, 32(12):1659–1673, 1996.

[31] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. Available at https://arxiv.org/abs/2004.05439, 2020.

[32] P. Ioannou and J. Sun. *Robust Adaptive Control*. Dover Publications, 2012.

[33] G. Joshi and G. Chowdhary. Deep model reference adaptive control. In *Proc. IEEE Conf. on Decision and Control*, 2019.

[34] G. Joshi, J. Virdi, and G. Chowdhary. Asynchronous deep model reference adaptive control. In *Conf. on Robot Learning*, 2020.

[35] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski. The Voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics and Automation Magazine*, 25(4):34–44, 2018.

[36] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[37] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Int. Conf. on Learning Representations*, 2015.

[38] J. Köhler, P. Kötting, R. Soloperto, F. Allgöwer, and M. A. Müller. A robust adaptive model predictive control framework for nonlinear uncertain systems. *Int. Journal of Robust and Nonlinear Control*, 2020. In press.

[39] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *Int. Conf. on Learning Representations*, 2018.

[40] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Conf. on Neural Information Processing Systems*, 2017.

[41] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi. *Adaptive Control: Algorithms, Analysis and Applications*. Springer-Verlag, 2 edition, 2011.

[42] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[43] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 2 edition, 1999.

[44] B. T. Lopez and J.-J. E. Slotine. Adaptive nonlinear control with contraction metrics. *IEEE Control Systems Letters*, 5(1):205–210, 2020.

[45] I. R. Manchester and J.-J. E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.

[46] I. M. Y. Mareels, B. D. O. Anderson, R. R. Bitmead, M. Bodson, and S. S. Sastry. Revisiting the MIT rule for adaptive control. In *IFAC Workshop on Adaptive Systems in Control and Signal Processing*, 1987.

[47] J. R. Medina and A. Billard. Learning stable task sequences from demonstration with linear parameter varying systems and hidden Markov models. In *Conf. on Robot Learning*, 2017.

[48] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. IEEE Conf. on Robotics and Automation*, 2011.

[49] D. Millard, E. Heiden, S. Agrawal, and G. S. Sukhatme. Automatic differentiation and continuous sensitivity analysis of rigid body dynamics. Available at https://arxiv.org/abs/2001.08539, 2020.

[50] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *Int. Conf. on Learning Representations*, 2019.

[51] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Dover Publications, 2005.

[52] K. S. Narendra and L. S. Valavani. Stable adaptive controller design – direct control. *IEEE Transactions on Automatic Control*, 23(4):570–583, 1978.

[53] K. S. Narendra, Y.-H. Lin, and L. S. Valavani. Stable adaptive controller design, part II: Proof of stability. *IEEE Transactions on Automatic Control*, 25(3):440–448, 1980.

[54] G. Niemeyer and J.-J. E. Slotine. Performance in adaptive manipulator control. *Int. Journal of Robotics Research*, 10(2):149–161, 1991.

[55] M. O'Connell, G. Shi, X. Shi, and S.-J. Chung. Meta-learning-based robust adaptive flight control under uncertain wind conditions. Available at https://arxiv.org/abs/2103.01932, 2021.

[56] J. C. Pinheiro and D. M. Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and Computing*, 6(3):289–296, 1996.

[57] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Wiley Interscience, 1962.

[58] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. EPOpt: Learning robust neural network policies using model ensembles. In *Int. Conf. on Learning Representations*, 2017.

[59] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Conf. on Neural*

Information Processing Systems, 2019.

[60] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli. Fully actuated multirotor UAVs. *IEEE Robotics and Automation Magazine*, 27(3):97–107, 2020.

[61] K. J. Åström and B. Wittenmark. Problems of identification and control. *Journal of Mathematical Analysis and Applications*, 34(1):90–113, 1971.

[62] S. M. Richards, F. Berkenkamp, and A. Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conf. on Robot Learning*, 2018.

[63] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Int. Symp. on Robotics Research*, 2013.

[64] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi. 6D physical interaction with a fully actuated aerial robot. In *Proc. IEEE Conf. on Robotics and Automation*, 2017.

[65] R. M. Sanner and J.-J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992.

[66] S. Singh, S. M. Richards, V. Sindhwani, J-J. E. Slotine, and M. Pavone. Learning stabilizable nonlinear dynamics with contraction-based regularization. *Int. Journal of Robotics Research*, 2020. In Press.

[67] R. Sinha, J. Harrison, S. M. Richards, and M. Pavone. Adaptive robust model predictive control with matched and unmatched uncertainty. Available at https://arxiv.org/abs/2104.08261, 2021.

[68] R. E. Skelton. Model error concepts in control design. *Int. Journal of Control*, 49(5):1725–1753, 1989.

[69] J.-J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *Int. Journal of Robotics Research*, 6 (3):49–59, 1987.

[70] J.-J. E. Slotine and W. Li. Composite adaptive control of robot manipulators. *Automatica*, 25(4):509–519, 1989.

[71] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[72] R. Soloperto, J. Köhler, M. A. Müller, and F. Allgöwer. Dual adaptive MPC for output tracking of linear systems. In *Proc. IEEE Conf. on Decision and Control*, 2019.

[73] D. Sun, S. Jha, and C. Fan. Learning certified control using contraction metric. In *Conf. on Robot Learning*, 2020.

[74] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine. Neural stochastic contraction metrics for learning-based control and estimation. *IEEE Control Systems Letters*, 5(5): 1825–1830, 2021.

[75] P. M. Wensing and J.-J. Slotine. Beyond convexity–contraction and global convergence of gradient descent. *PLoS ONE*, 15(12), 2020.

[76] P. M. Wensing, S. Kim, and J.-J. E. Slotine. Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1): 60–67, 2017.

[77] P. Zheng, X. Tan, B. B. Kocer, E. Yang, and M. Kovac. TiltDrone: A fully-actuated tilting quadrotor platform. *IEEE Robotics and Automation Letters*, 5(4):6845–6852, 2020.

[78] J. Zhuang, N. Dvornek, X. Li, S. Tatikonda, X. Papademetris, and J. Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. In *Int. Conf. on Machine Learning*, 2020.

APPENDIX A
TRAINING DETAILS

Our Method: To meta-train $\theta_{\text{ours}} \coloneqq (\theta_y, \Lambda, K, \Gamma)$, we first train an ensemble of M models $\{\hat{f}_j\}_{j=1}^{M}$, one for each trajectory $\mathcal{T}_j$, via gradient descent on the regression objective (12) with $\mu_{\text{ensem}} = 10^{-4}$ and a single fourth-order Runge-Kutta step to approximate the integral over $t_{k+1}^{(j)} - t_k^{(j)} = 0.01$ s. We set each $\hat{f}_j$ as a feed-forward neural network with 2 hidden layers, each containing 32 $\tanh(\cdot)$ neurons. We perform a random $75\%/25\%$ split of the transition tuples in $\mathcal{T}_j$ into a training set $\mathcal{T}_j^{\text{train}}$ and validation set $\mathcal{T}_j^{\text{valid}}$, respectively. We do batch gradient descent via Adam [37] on $\mathcal{T}_j^{\text{train}}$ with a step size of 10^{-2}, over 1000 epochs with a batch size of $\lfloor 0.25|\mathcal{D}_j^{\text{train}}| \rfloor$, while recording the regression loss with $\mu_{\text{ensem}} = 0$ on $\mathcal{T}_j^{\text{valid}}$. We proceed with the parameters for $\hat{f}_j$ corresponding to the lowest recorded validation loss.

With the trained ensemble $\{\hat{f}_j\}_{j=1}^{M}$, we can now meta-train $\theta_{\text{ours}} \coloneqq (\theta_y, \Lambda, K, \Gamma)$. First, we randomly generate $N = 10$ smooth reference trajectories $\{r_i\}_{i=1}^{N}$ in the same manner as above, each with a duration of $T = 5$ s. We then randomly sub-sample $N_{\text{train}} = \lfloor 0.75N \rfloor = 7$ reference trajectories and $M_{\text{train}} = \lfloor 0.75M \rfloor$ models from the ensemble to form the meta-training set $\{(r_i, \hat{f}_j)\}_{i=1,j=1}^{N_{\text{train}}, M_{\text{train}}}$, while the remaining models and reference trajectories form the meta-validation set. We set $Ay(q, \dot{q}; \theta_y)$ as a feed-forward neural network with 2 hidden layers of 32 $\tanh(\cdot)$ neurons each, where the adapted parameters $A(t) \in \mathbb{R}^{n_q \times 32}$ with $n_q = 3$ serve as the output layer. We use an unconstrained log-Cholesky parameterization for each of the positive-definite gains (Λ, K, Γ). We set up the meta-problem (11) using $\{(r_i, \hat{f}_j)\}_{i=1,j=1}^{N_{\text{train}}, M_{\text{train}}}$, $\alpha = 10^{-3}$, $\mu_{\text{meta}} = 10^{-4}$, and the adaptive controller (17). We then perform gradient descent via Adam with a step size of 10^{-2} to train $\theta_{\text{ours}} \coloneqq (\theta_y, \Lambda, K, \Gamma)$. We compute the integral in (11) via a fourth-order Runge-Kutta integration scheme with a fixed time step of 0.01 s. We back-propagate gradients through this computation in a manner similar to Zhuang et al. [78], rather than using the adjoint method for neural ODEs [19], due to our observation that the backward pass is sensitive to any numerical error accumulated along the forward pass during closed-loop control simulations. We perform 500 gradient steps while recording the meta-loss from (11) with $\mu_{\text{meta}} = 0$ on the meta-validation set, and take the best meta-parameters θ_{ours} as those corresponding to the lowest recorded validation loss.

ACMRR Baseline: To meta-train $\theta_{\text{ACMRR}} \coloneqq \theta_y$, we first perform a random $75\%/25\%$ split of the transition tuples in each trajectory $\mathcal{T}_j$ to form a meta-training set

$\{\mathcal{T}_j^{\text{meta-train}}\}_{j=1}^M$ and a meta-validation set $\{\mathcal{T}_j^{\text{meta-valid}}\}_{j=1}^M$. We set $Ay(q, \dot{q}; \theta_y)$ as a feed-forward neural network with 2 hidden layers of 32 $\tanh(\cdot)$ neurons each, where the adapted parameters $A(t) \in \mathbb{R}^{n_q \times 32}$ with $n_q = 3$ serve as the output layer. We construct the meta-problem (4) using the task loss (22), the adaptation mechanism (23), and $\mu_{\text{meta}} = 10^{-4}$; for this, we use transition tuples from $\mathcal{T}_j^{\text{meta-train}}$. We then meta-train θ_{ACMRR} via gradient descent using Adam with a step-size of 10^{-2} for 5000 steps; at each step, we randomly sample a subset of $\lfloor 0.25|\mathcal{T}_j^{\text{meta-train}}| \rfloor$ tuples from $\mathcal{T}_j^{\text{meta-train}}$ to use in the closed-form ridge regression solution. We also record the meta-loss with $\mu_{\text{meta}} = 0$ on the meta-validation set at each step, and take the best meta-parameters θ_{ACMRR} as those corresponding to the lowest recorded validation loss.